FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Application of Explainable AI in Deep Learning Models

Mariana Margalho Alves Calado

MESTRADO EM ENGENHARIA BIOMÉDICA

Supervisor: João Nuno Duarte Fernandes

Co-Supervisor: Jaime dos Santos Cardoso

September 3, 2023

Application of Explainable AI in Deep Learning Models

Mariana Margalho Alves Calado

MESTRADO EM ENGENHARIA BIOMÉDICA

Faculdade de Engenharia da Universidade do Porto

September 3, 2023

Resumo

A adoção crescente de sistemas de Deep Learning transformou a sociedade moderna numa sociedade algorítmica, levando à necessidade de decisões informadas e fiáveis. No entanto, o uso destes sistemas no apoio à decisão permanecem como "caixas negras" em que o seu funcionamento interno está oculto e torna-se complexo de ser compreendido por especialistas. Assim, a sua implementação torna-se difícil em áreas consideradas críticas, em que as decisões que são tomadas levam a consequências graves como é o caso do sector da saúde. Este domínio requer sistemas com elevados níveis de transparência e responsabilidade. A Inteligência Artificial Explicável pretende satisfazer essas preocupações. A comunidade científica tem vindo a desenvolver algoritmos explicáveis, utilizando métodos que analisam o modelo após o seu treino, em vez de métodos que restringem a complexidade do modelo durante o treino.

Esta dissertação tem como foco principal aprofundar o conhecimento nesta área, desenvolvendo um modelo intrínseco explicável de Deep Learning para a deteção de objetos, sem que o desempenho preditivo do modelo seja sacrificado. As explicações produzidas baseiam-se no processo de raciocínio que as pessoas normalmente fazem para identificar um determinado objeto, utilizando o seu conhecimento prévio sobre as características do objeto e fazendo uma análise das semelhanças com as observações passadas.

Esta nova *framework* envolveu um detetor de objetos de dois estágios bastante conhecido que é o Faster-RCNN com o modelo da ProtoPNet tendo sido treinada e avaliada no "GRAZPEDWRI-DX", um conjunto de dados de raio-X de pulso com trauma pediátrico. Posteriormente, foi desenvolvida e avaliada uma outra abordagem que inclui os mesmos modelos integrados que na solução proposta, mas numa configuração diferente. Finalmente, para além desses dois métodos, foram também testados em separado os modelos que compõem as abordagens para um estudo comparativo do nível de desempenho e qualidade de explicações geradas.

Os resultados obtidos mostram que o método intrinsecamente explicável, não afeta significativamente o desempenho ou o tempo de inferência do modelo de deteção usado, atingindo uma Mean Average Precision de 66,1%.

Abstract

The increasing adoption of Deep Learning systems has transformed modern society into an algorithmic society, leading to a need for informed and reliable decisions. However, these systems used in decision support are often "black boxes" in which their inner workings are hidden and become complex for experts to understand. Thus, their implementation becomes difficult in areas considered critical, where the decisions made lead to severe consequences as in the healthcare sector. This domain requires systems with high levels of transparency and accountability. The Explainable Artificial Intelligence field aims to satisfy these concerns. The scientific community has been developing explainable algorithms by using methods that analyse the model after it has been trained, instead of methods that constrain the complexity of the model during training.

This dissertation focuses on furthering the knowledge of this topic by developing an intrinsic explainable Deep Learning model for object detection, without sacrificing its predictive performance. The explanations produced are based on the reasoning process used by persons to identify a certain object, using their previous knowledge of the object's characteristics, and making an analysis of the similarities with past observations.

This new framework involved a two-stage object detector well-known that is the Faster-RCNN model with the ProtoPNet model, having been trained and evaluated on the GRAZPEDWRI-DX dataset, a pediatric trauma wrist X-ray dataset. Later, another approach was developed and evaluated that included the same integrated models as in the proposed solution, but in a different configuration. Finally, in addition to these two frameworks, the models that make up the approaches were also tested separately for a comparative study of the level of performance and quality of explanations generated.

The results demonstrate that the method designed for intrinsic explainability has minimal impact on both the performance and inference time of the detection model. The model achieved a mean Average Precision of 66.1%.

Acknowledgements

Firstly, I would like to thank my supervisor, João Fernandes, and my co-supervisor, Jaime S. Cardoso, for giving me the opportunity to work on such a challenging project in a field that both captivates and encourages my curiosity. Their solid guidance, support, and insightful advice have been instrumental in my progress.

I would also like to express my immense gratitude to Luís Fernandes, who accompanied me throughout the dissertation process, giving his full availability to advise and help me whenever I needed it, even in the smallest details, such as existential doubts.

I would also like to acknowledge Ricardo Cruz and Celso Pereira for their useful and constructive suggestions, which have significantly enhanced the effectiveness and clarity of my work.

A special mention goes to the THEIA project (Automated Perception Driving), in which the dissertation was developed. Their environment provided the perfect setting to delve into this research.

Lastly, I thank my family and friends, who supported me during moments of frustration, preventing me from losing sight of my motivation and goals.

Mariana Calado

"The only journey is the one within."

Rainer Maria Rilke

Contents

List of Figures

List of Tables

Acronyms, Abbreviations and Symbols

AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Average Precision
BCE	Cross-Entropy Loss
CAMs	Class activation maps
CAVs	Concept Activation Vectors
CBR	Case-Based Reasoning
CE	Cross-Entropy Loss
CNN	Convolutional Neural Network
CT	Computed Tomography
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
FC	Fully connected layer
Fast R-CNN	Region-based Convolutional Network
Grad-CAM	Gradient-weighted Class Activation Mapping
iFaster-RCNN	Interpretable-Faster-RCNN
IoU	Intersection-over-Union
KB	Knowledge Base
LIME	Local Interpretable Model-Agnostic Explanations
LRP	Layer-wise Relevance Propagation
MAE	Mean Absolute Error
mAP	Mean Average Precision
ML	Machine Learning
MSE	Mean Squared Error
NMS	Non-Maximum Suppression
NN	Neural Network
PNG	Portable Network Graphics
ProtoPNet	Prototypical Part Network
RCNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
ResNet	Residual neural network
RMSE	Root Mean Square Error
RoI	Region of Interest
RPN	Region Proposal Network
SHAP	Shapley Additive Explanations
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine
xAI	Explainable Artificial Intelligence
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Context

Nowadays, machine learning (ML) systems are everywhere influencing choices made in both industry and research and showing more and more abilities to carry out different tasks. It has been part of our daily lives through examples from mobile music applications that provide personalized audio content recommendations and facial recognition technology used to unlock smartphones to more critical domains whose decisions have a significant impact on society, such as healthcare systems, criminal decision systems or self-driving vehicles [1]. The ML subfield that has been responsible for driving most of these advancements is Deep Learning (DL).

The use of Deep Neural Networks (DNNs) models for medical imaging analysis has led to the development of a wide variety of applications ranging from lesion detection to diagnosis, and prognosis [2–4]. However, if these models are to be deployed in medical institutions they must comply with safety and regulatory concerns. The fact that DL models operate as a black box, where is difficult to interpret what is happening inside it and what led to that decision, makes these entities not trust those systems and therefore not risk a person's life.

Explainable Artificial Intelligence (xAI) is a branch of Artificial Intelligence (AI) that aims to address these concerns by developing methods that can provide clear, easy-to-understand explanations for the decisions made by those systems. Explaining the inner workings related to the prediction of the outputs can be challenging because of the complexity and dimensions of deep learning models. With this, understanding the logic behind such models becomes a crucial step in identifying potential biases or errors in their predictions, providing a better comprehension of their malfunction. This way, trustability, transparency, bias and fairness of AI algorithms will increase, ensuring that they are used ethically and responsibly [5].

Presently, most explainable models developed in the state-of-the-art are based on post-hoc techniques that explain opaque neural networks by perturbing the input data, creating approximations, saliency maps or derivatives. Nevertheless, this type of explanation can lead to misleading results because, for example, using gradient-based saliency maps methods only show the areas of the most relevant images, since they tend to highlight edges and do not demonstrate how the

calculation of pixels is done to arrive at the result [6]. Despite this, few works overcome these post-hoc models, which are inherently interpretable and have a reasoning process designed to be understandable to humans. Examples of models that are developed with transparency in mind so that their predictions are explainable without needing extra properties can be the neural network latent space disentanglement [7] or the use of Case-Based Reasoning, which involves using the knowledge obtained from past situations to understand new cases requiring the network to implement logical conditions in its final layers [8].

Furthermore, most explainable AI approaches focus on using deep learning models for classification tasks, rather than object detection tasks. Object detection is an important computer vision task that allows one to understand the position and dimensions of multiple objects of interest within an image or video frame and classify each into predefined categories. Its application in medicine can aid in the precise location and identification of early signs of abnormalities and improve diagnosis accuracy. Additionally, object detection can automate time-consuming tasks such as cell counting, reducing the risk of human error [4].

1.2 Motivation

The field of Artificial Intelligence has made remarkable progress in modern computer science technology. However, the complex nature of these models often makes it difficult for experts and non-experts alike to understand their decisions and the reasoning process. This lack of transparency and accountability can be a barrier to the wider adoption of those systems in critical domains such as medicine, which is a highly regulated field and every decision-making leads to severe consequences.

XAI aims to make models more transparent and understandable to humans in order to provide general public trust. By studying intrinsic explainable deep learning methods, particularly Case-Based Reasoning approaches, it will be possible to see how the opaque model arrived at a specific conclusion. This will enable healthcare professionals to review the reasoning and assess its accuracy, thus promoting the adoption of these models in clinical settings. The application of these methods to object detection tasks has the potential to significantly benefit the field of xAI research, specifically in the field of medicine, where advancements are presently limited. This study can result in improved diagnosis through the enhanced ability to locate specific objects in addition to classification.

In sum, research on this topic aims to promote advances in the xAI field and help make deep learning models more accessible and trustworthy for clinical professionals. And, additionally, provide valuable insights for future research and development in this area.

1.3 Goals

The overall goal of this dissertation is to research and develop an intrinsic explainable deep learning model for object detection and with this make relevant contributions toward the future inte-

gration of DL algorithms in clinical practice. The developed model should be robust, generate explainability in a transparent manner, and at the same time exhibit high predictive performance.

1.4 Document Structure

The structure of the document begins with an introduction, in which the context, motivation and goals of the dissertation are presented. In Chapter 2, the fundamental principles of Deep Learning are introduced, accompanied by an exploration of their significance in the field of computer vision. Chapter 3 provides a comprehensive review of the current state-of-the-art in object detection, giving an overview of the main concepts of this computer vision task, followed by an analysis of the existing categories of detectors, with a focus on some of the most well-known examples within each category. Ending up with a description of the use of object detection in the medical field, including examples of applications, highlighting the challenge associated with developing algorithms for medical image data. Chapter 4 presents a literature review on explainability in deep learning, starting by describing relevant concepts in xAI and their importance. A taxonomy of xAI techniques is presented, and the most commonly used explainability strategies are discussed. Some examples of the use of xAI on critical applications are mentioned, and the existing literature on explainability in object detection is examined. Chapter 5, consists of the dissertation materials and methods, where is presented a description of the selected dataset for this study and the pre-processing used on the data, followed by a description of the proposed frameworks developed and ends up with some implementation details. Chapter 6 involves a critical discussion of the results of the different experiments carried out in this work, including a comparison with current state-of-the-art methodologies. In addition, the hyperparameters used to train the models that generated the results are detailed. Finally, Chapter 7 presents the study's conclusions and future work.

Chapter 2

Fundamentals: Deep Learning

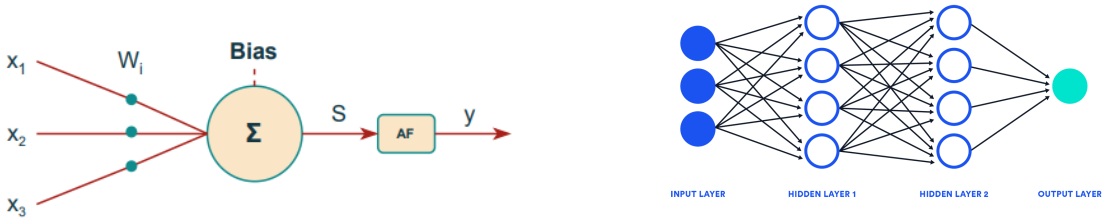
2.1 Basic Concepts

AI research initially focused on issues that could be modelled using mathematical models. Machine learning became a significant field in AI by developing models and optimization algorithms capable of solving complex problems that were challenging to codify using available mathematical tools and human understanding.

Deep learning, a subset of machine learning, differs from traditional ML methods in that it eliminates the need for separate feature extraction. Instead, DL models can automatically determine the relevant features for a given task enabling systems to directly process raw data and produce desired outcomes with effective performance. In essence, deep learning allows models to learn and represent data through multiple levels of abstraction emulating how the brain processes multimodal information, thereby capturing complex patterns of large-scale data [9].

Artificial Neural Networks (ANN) is a machine learning approach that gradually gained popularity, it takes inspiration from the human nervous system and how the communication structure of the brain works. It consists of units and connections between them that attempt to emulate biological neurons and synapses, which is how the nervous impulses are transmitted from one neuron to another. The network processes these units arranged in input, hidden, and output layers (as only the input and output layers are visible) with connections between nodes in adjacent layers (Figure 2.1b). These connections are assigned weight values. By multiplying the inputs with their corresponding weights and summing them at each unit, a transformation occurs based on an appropriate activation function (Figure 2.1a) [10].

These functions are used to help in the model's training process to learn patterns in data as they compute the weighted sum and biases and determine whether a neuron should be activated. It also has the purpose to introduce non-linearity to the output of a neuron, enabling the network to learn more complex tasks.



(a) **(b)**

Figure 2.1: (a) Neural Network active node (Perceptron): input x_i, weights W_i, activation function AF, sum of the weighted input S and output y. (b) An artificial neural network Architecture from [11].

The following presents and describes some examples of widely used non-linear activation functions, where x is the input value [12, 13]:

Sigmoid: Also known as the logistic function, it transforms the input value into a range between 0 and 1 thereby yielding a smooth, non-linear response that can be interpreted as a probability. The derivative of this function remains consistently low, never surpassing 1. This characteristic can pose a challenge during model training due to the occurrence of vanishing gradients, where gradients become extremely small, close to zero, which impedes the optimization of unit weight values. Consequently, this results in slower learning and complicates DL processes. For this reason, this function is typically used as a final output activation function.

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

Hyperbolic Tangent (Tanh): This function, often abbreviated as Tanh, was one of the proposed to solve some of the sigmoid activation function limitations. Unlike the sigmoid function, it maps the input value to a range between -1 and 1 and it is zero-cantered, which can help in optimizing the network and improving convergence. However, the tanh function still suffers the vanishing gradient problem.

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \tag{2.2}$$

Rectified Linear Unit (ReLU): Compared to the previous activation functions, this one introduces sparsity in hidden units and helps overcome the vanishing gradient problem. Here negative input values are set to zero while positive input values keep unchanged, so the neuron units will only be deactivated if the transformed output is less than 0, eliminating the vanishing gradient problem by avoiding saturating issues, and allowing gradients to flow more freely. The disadvantage is that the constant non-activation of negative units can cause "dead ReLU" problem.

$$f(x) = max(0,x) = \begin{cases} x, & if \quad x \geq 0 \\ 0, & if \quad x < 0 \end{cases} \tag{2.3}$$

Leaky Rectified Linear Unit (Leaky ReLU): In order to address the "dead ReLU" problem, it introduces a small slope for negative values, allowing some activation to flow even when the

input value is negative.

$$f(x) = \begin{cases} x, & if \quad x > 0 \\ ax, & if \quad x \leq 0 \end{cases} \tag{2.4}$$

Softmax: Like the sigmoid function, this one is also used to convert a vector of real-valued numbers into a probability distribution, with the resulting output ranging between 0 and 1. It shares similarities with the sigmoid function but is specifically designed for multi-classification tasks.

$$f(x) = \frac{e^x}{\sum_j e^{xj}} \tag{2.5}$$

In Figure 2.2 it is possible to find the graphical representations of some of the activation functions mentioned.

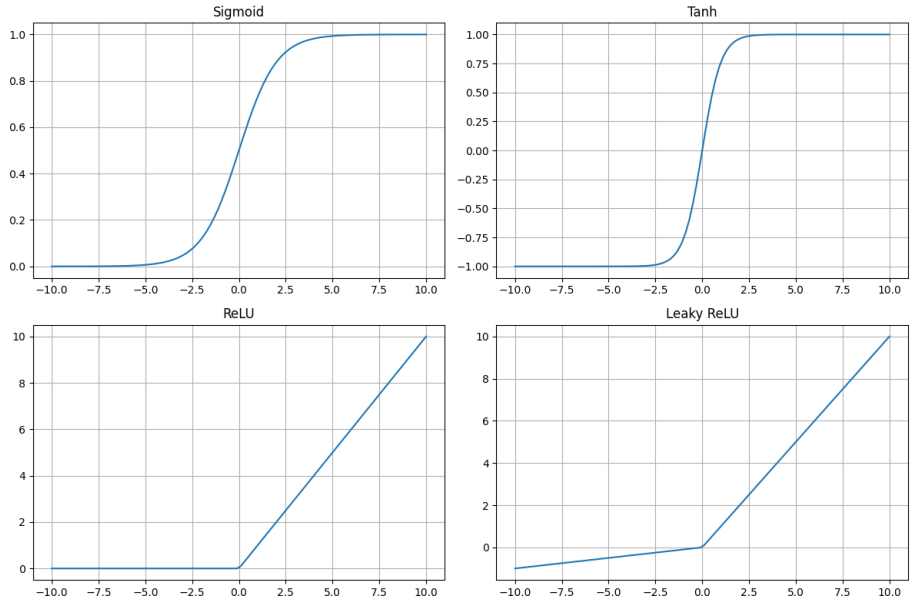


Figure 2.2: Activation functions.

The ANN can learn features by comparing their predictions with known correct outputs thanks to an optimization process known as backpropagation. By calculating the gradient of a loss function with respect to the model's parameters (weights) the model could update itself, improving its accuracy, and hopefully generalize to unseen data, that is allowing it to make predictions about new data with an increased precision.

To effectively use deep learning, it is important to grasp certain concepts and procedures:

Optimizer. As explained before, the training process of a DNN involves an iterative procedure where the model parameters such as weights and learning rates are updated taking into account the gradient of the loss function. An optimizer is an algorithm that defines the way of adjusting these parameters in order to reduce the loss and consequently improve the model performance. Some of

the widely employed optimization strategies include Stochastic Gradient Descent (SGD), Adam and Root Mean Square Prop (RMSProp).

Regularization. Preventing overfitting is a crucial aspect of training deep learning models. This phenomenon occurs when a model becomes excessively fitted to the training data, not being able to perform well against unseen data in the test set and consequently missing the ability to generalize to new data. Here, the model is focusing on capturing data points that do not genuinely reflect the underlying properties of the training data. To address this issue, regularization is usually implemented during training to generate a generalized model, and consequently reduce the test error. L1 and L2 regularization works by adding a penalty term in the error function, this term penalizes variations in the model output when the input is transformed. Dropout, Early Stopping and Data augmentation are also regularization methods. Dropout consists of randomly ignoring a certain number of output layers during the training process, modifying the appearance of the layers and changing the connection to the other layers in this way each update to a layer during the learning process occurs from a distinct perspective of the arranged layer, that is, each node in this layer assumes different levels of responsibility when it receives data, introducing more noise and making the model become more robust and better adapted to the received data [14]. Early Stopping consists in stopping the training process when the performance on the validation set stops improving. Finally, Data Augmentation is a strategy that by generating transformed samples from the original train set increases variability and consequently reduces generalization error.

Data Split. Once a dataset is available, it must have to be divided into three distinct parts: a training set, a validation set, and a testing set. In the training phase, the primary purpose of the model is to learn patterns and features from the data, this process occurs at each epoch, which represents a complete pass through the entire dataset. The validation set is used to assess the model performance during the training, it evaluates the current level of generalization by making predictions on the validation set after a specific number of training iterations. Finally, the testing phase begins only after the training phase is completed. A dataset independent and has never been exposed to the model before is provided for evaluating the model's understanding of patterns using predetermined metrics.

2.2 Computer Vision

Deep Learning has been the primary driver behind the exponential improvement in the performance of Computer Vision (CV) systems thus far, enabling computers to extract relevant information from images, videos, and other visual inputs. This advancement has empowered computers to perceive, observe, and comprehend their surroundings, making informed decisions and providing recommendations.

Deep neural networks (DNNs) have proven effective in representing images at different levels, making them valuable for tasks like object detection, semantic segmentation, motion tracking,

style transfer, action recognition in Computer Vision [9,13]. Consequently, the subsequent subsection will focus on Convolutional Neural Networks (CNNs), a crucial type of deep learning model that has substantial relevance in visual comprehension.

2.2.1 Convolutional Neural Networks

Convolutional neural networks [15] are specifically intended to address the limitations of treating image data as flattened vectors. In contrast to fully connected multilayer perceptron (MLP) that are agnostic to the spatial structure of pixels, CNNs take advantage of the inherent spatial relationships in images. By leveraging the prior knowledge that nearby pixels are usually related to each other, CNNs enable efficient modelling and learning from image data [13].

These networks have become a fundamental tool in the field of CV and have yielded significant advancements in various applications. For example, in the year of 2012, in order to classify images from the ImageNet dataset into 1000 different classes, Krizhevsky et al. [16] used a CNN that achieved remarkable results decreasing the classification error rate compared to other approaches.

Another advantage of CNNs is their computational efficiency as they require fewer parameters compared to fully connected architectures. Additionally, the structure of convolutions lends itself well to parallelization across GPU cores, further enhancing computational efficiency [17].

A CNN is composed of three types of layers: convolutional layers, pooling layers, and fully connected layers (Figure 2.3). In this architecture, an activation function is consistently applied to the output of the convolutional layer before the pooling operation takes place. This application of the activation function modifies the obtained output of the convolutional layer.

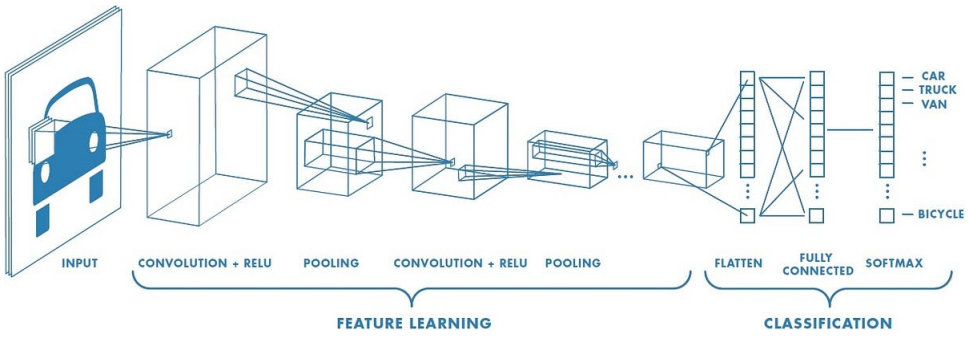


Figure 2.3: A complete convolutional neural network architecture for a classification task from [18].

Convolutional layer: In this layer, the input image suffers a convolution operation and outputs a matrix called "feature map". Convolution serves as the initial stage in feature extraction from an input image. By employing small square sections of input data, known as filters or kernels, convolution effectively captures and retains the inter-pixel relationships in the input image. Functioning as a numerical operation, it involves an image matrix and a spatial filter to learn significant image features such as edges, lines or other more complex shapes. By using different convolution kernels is possible to perform different operations and obtain different feature maps. In addition to the size of convolution kernels and the number of channels, there are other parameters that can be tweaked in a CNN, such as the stride and padding techniques. These parameters play a crucial

role in controlling the spatial dimensions and information flow in the network, affecting the size, resolution, and receptive field of the resulting feature maps.

Pooling layer: Following the convolution step, there is a downsampling process that serves to reduce the spatial dimension (width × height) of each feature map, retaining the most important information. Average pooling and max pooling are the most commonly used methods. This process consists of applying a sliding window along the matrix thus performing operations such as average or maximum (Figure 2.4).

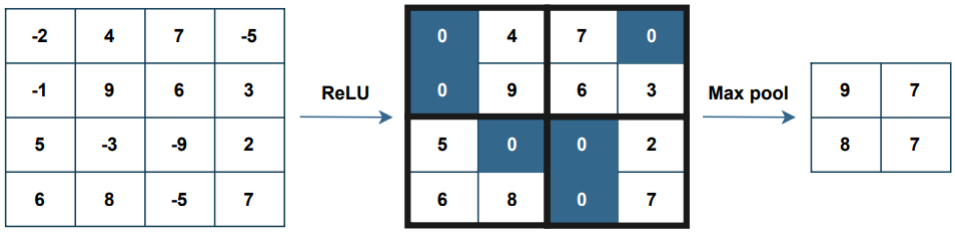


Figure 2.4: Max pooling operation. This demonstration uses a 2x2 filter and a stride of 2.

Fully connected layer (FC): After going through several convolutional and pooling layers, the neural network carries out the higher-level reasoning using fully connected layers. Here, each neuron is connected to all the activations in the previous layer, as indicated by the layer name. To compute the activation of these neurons, a matrix multiplication is performed, followed by the addition of a bias constant. And at the end, these layers transform the two-dimension feature maps into a one-dimension feature vector. This output vector can then either be passed through a set of classes for decision-making (in this case classification) or treated as a feature vector for additional processing [9].

2.3 Object Detection Overview

In image classification, it is assumed that there is only one object in an image and the main goal is how to recognize its category. However, there are frequently several objects of interest in an image and in addition to wanting to know what it is, it is intended to understand their specific locations in that image. This Computer Vision technique is known as object detection [13].

Before describing this task, it is necessary to distinguish between object localization and object detection since there is a difference between these two terms. The first is finding what and where a single object exists in an image and the second is finding what and where multiple objects are in an image.

In many datasets, there is only one class and the image is cropped around the object. In localization, in addition to classifying, the idea is to spatially locate each interest object in the frame using for example a bounding box that is a rectangular region that encloses the actual object.

Thus, object detection is a regression and a classification problem as it estimates the position and dimensions of the bounding boxes and classifies them into predefined categories. This method

performs a significant role in scene understanding and has been used in several areas like robotics, security, autonomous driving, retail, medicine, video surveillance and military.

2.3.1 Region Proposal Methods

The most common approaches to generalize the localization of one object for multiple objects in an image are the Sliding Window and the Selective Search.

The Sliding Window approach is the *natural* extension of localization that involves dividing the input image into a grid of overlapping regions and evaluating each window to determine if it contains an object. This technique can be computationally expensive, as it requires evaluating a large number of windows making it disadvantageous [19].

To address this issue, another method was developed named Selective Search which is a deterministic algorithm that extracts a potential set of regions in an image that is likely to contain objects and then ranks these regions based on their similarity to known object classes [20].

2.3.2 Feature extraction

Extracting robust features for object detection can be challenging, as objects can appear in a wide range of variations in terms of shape, texture, colour, and illumination. Scale-Invariant Feature Transform (SIFT) is a popular feature descriptor that is robust to scale and orientation changes and can capture the local structure of an image [21]. Histogram of Oriented Gradients (HOG) is another popular feature descriptor that is used to capture the structure and shape of objects by representing the distribution of edge orientations in an image [22]. Haar-like features are binary features that are computed using a simple sum of pixel intensities and are commonly used in the Viola-Jones object detection framework [23].

While these feature descriptors have been widely used in object detection, they may not always be sufficient to capture the complex variations that can occur in real images. This is where deep learning-based approaches, such as CNNs, have become increasingly popular. These networks can learn features automatically from the image and be effective at object detection in a wide range of scenarios.

2.3.3 Evaluation Metrics

The most common metric used for measuring the accuracy of object detectors is Average Precision (AP) which computes the average detection precision value under different recall values. Since the formula of this metric is based on other sub-metrics, such as Intersection-over-Union, Precision and Recall it is necessary to understand them first.

Intersection-over-Union (IoU) measures how much the predicted bounding box (A) overlaps with the ground truth bounding box (B). This similarity can be computed using the ratio of these two bounding boxes intersection area by their union area as shown in Equation 2.6. The IoU range is between 0 and 1, where 0 means that two bounding boxes do not overlap at all, while 1 suggests that the two bounding boxes are equivalent [13].

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.6}$$

In order to calculate Precision and Recall, to classify whether the detection of an object is correct or not compared to the ground truth it is defined a threshold value for IoU.

The prediction is classified as True Positive (TP) when the IoU is greater than the threshold which means that the prediction that an object exists and the ground truth object also denotes the same. On the other hand, a False positive (FP) means there is a prediction of an existing object, but the ground truth reveals that the object is not present and the IoU is less than the threshold. The prediction is classified as True Negative (TN) when the model predicts that an object is not present, and the ground truth also shows that the object is not present. Finally, and contrary to the previous, False negative (FN) the ground truth indicates that the object is present, but the model's prediction says the opposite.

Precision measures the fraction of the bounding box predictions that are actually correct from all the predictions:

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

Recall measures the fraction of the bounding box predictions that were correctly detected from all the target objects that are present:

$$Recall = \frac{TP}{TP + FN} \tag{2.8}$$

The F1 Score metric evaluates the balance between recall and precision, reflecting the relationship between the behaviours of these metrics. When the obtained value is lower means that there is a greater imbalance between recall and precision scores and vice versa:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.9}$$

After explaining these terms, the first step in Mean Average Precision is to get all predictions made in all the images and rank them in descending order according to the predicted confidence score. Then the Precision and Recall metrics are calculated as it goes through all outputs and these values are plotted in a graph called Precision-Recall Curve, where precision is on the y-axis and the recall is on the x-axis. It is calculated the area under that graph, which is basically how Average Precision is measured for a specific class. Before calculating this, the resulting plot has some variations in the ranking, so it is often smoothed out as it can be seen in Equation 2.11, the precision value for recall $\hat{r}$ is replaced with the maximum precision for any recall $\geq \hat{r}$.

Average Precision is the weighted mean of precisions at each threshold and the weight is the increase in recall from the previous threshold. The Mean Average Precision(mAP) is calculated by finding the AP for each class and then averaging over the number of classes. The Equations

2.10 and 2.12 of AP and mAP are shown below, where N is the number of classes and AP(i) refers to the used AP for class i [24].

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

where,

$$p_{interp}(r) = max_{\hat{r}\geq r} p(\hat{r}) \tag{2.11}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{2.12}$$

2.3.4 Loss Functions

As described previously, the goal of object detection is to predict both the location of the bounding box and the class to which the object belongs. This is accomplished through the training phase of the model, during which the model learns to perform object detection by gradually minimizing a loss function. This loss function assesses the discrepancy between the predicted results and the actual results. So, the object detection loss function is a combination of regression loss functions and classification loss functions.

Regression Loss Functions. Measure the accuracy of predicted bounding boxes compared to the ground truth bounding boxes.

This evaluation is done by computing the Mean Squared Error (MSE) between the predicted box coordinates P and the ground truth box coordinates T:

$$MSE(T,P) = \frac{1}{N}\sum_{xi,yi,i=1}^{N} (T_{xi} - P_{yi})^2 \tag{2.13}$$

Where N is the number of coordinates (x1, y1, x2, y2) [25].

A similar function that also can be used is the Mean Absolute Error (MAE), also known as L1 Loss which measures the absolute error difference between P and T. MAE is more robust to outliers since it converts all values to positive:

$$MAE(T,P) = \frac{1}{N}\sum_{xi,yi,i=1}^{N} |T_{xi} - P_{yi}| \tag{2.14}$$

Finally, another important function is the Root Mean Square Error (RMSE) that mitigates the problem of MSE high sensitivity to outliers due to its strong punishment of large errors:

$$RMSE(T,P) = \sqrt{MSE(T,P)} = \sqrt{\frac{1}{N}\sum_{xi,yi,i=1}^{N} (T_{xi} - P_{yi})^2} \tag{2.15}$$

Classification Loss Functions. Evaluate the performance of the object detector in correctly assigning the right class to a given input at the location of the predicted bounding boxes.

The Cross-Entropy Loss (CE), also called negative Log Likelihood, is the most common loss for classification tasks. Observing the formula below, this loss is obtained by multiplying the logarithm of the predicted probability p for the ground truth class t. This function strongly penalizes the predictions that are confident but wrong [25].

$$CE(t,p) = -\sum_{i=1} t_i \times \log(p_i) \tag{2.16}$$

Binary Cross-Entropy Loss (BCE) is a variation of Cross-Entropy Loss as the name implies is used for the classification of two classes:

$$BCE(t,p) = -\frac{1}{N}\sum_{i=1}^{N} t_i \times \log(p_i) \tag{2.17}$$

2.3.5 Post-processing

Non-Maximum Suppression (NMS) is an important object detection technique used as a post-processing step to remove the duplicated bounding boxes predictions and get the final detection outcome. The model usually generates multiple bounding boxes predictions for one object of interest and at the end, it is desired to have only one prediction. The idea of this process is, for a set of overlapped bounding boxes predictions with their probability score/confidence (indicating how likely it is an object in that box), the bounding box with the highest confidence is selected while the other boxes are removed according to a predefined IoU threshold, this continues repeatedly until all boxes have been evaluated. If there are multiple classes in an image the NMS algorithm is performed independently for each class [13, 26].

Chapter 3

State-of-the-art Review: Object Detection

3.1 Common Detectors

Two main categories of state-of-the-art object detection methods are one-stage and two-stage detectors [27]. These methods use deep learning to process an input image and identify the presence and location of objects in that image. An object detector's goal is based on two tasks: identify a potentially unlimited number of objects in an image, classify each object and estimate the size of each object using a bounding box.

One-stage object detectors combine these tasks into a single step, which allows them to achieve higher performance at the expense of accuracy, therefore taking less time and can be used in real-time applications. On the other hand, Two-stage object detectors separate the tasks into two stages, which can improve accuracy but may come at a cost in performance, as they are typically slower [28].

3.1.1 Two-Stage Object Detectors

Two-Stage Object Detectors produce, in the first stage, a set of potential bounding boxes (proposals) that may contain objects. In the next stage, they use a CNN model to extract features, classify object categories and predict the presence of an object within each proposal. R-CNN, SPP-Net, Fast R-CNN, Faster R-CNN, R-FCN and FPN are some examples of two-stage object detectors [28].

3.1.1.1 R-CNN and Fast R-CNN

Girshick et al. [29] proposed an object detection algorithm called Regions with CNN features (R-CNN) that extracts around two thousand region proposals at multiple scales with distinct sizes and shapes for each image by Selective Search. Each region proposal is labelled with a ground truth bounding box and a class. These are warped into a bounding box with the required fixed size to

serve as input into a CNN, where the extraction of a fixed number of features happens. After that, the obtained features are fed into a Support Vector Machine (SVM) to classify the presence of the object within that candidate region proposal. At the end of this, the non-maximum suppression method is applied for each class independently. This method presents problems such as the model train time and cost as the training requires multiple phases and disk storage for the features. The detection is slow since it has to process thousands of object proposals per image. Furthermore, the Selective Search method is fixed, thus it is not able to learn what can lead to general bad candidate object proposals. In sum due to a load of repeated computation, it becomes impossible to use this technique for real-world applications.

In order to override these disadvantages, the same authors developed the Fast Region-based Convolutional Network method (Fast R-CNN) [30] which instead of sending all the 2000 features per image (individual region proposals) to the CNN, they generated a convolutional feature map for the entire image making it a trainable network. After all the object proposals have been extracted, they go through the RoI pooling layer. This layer uses max pooling to convert the features of the object proposals into a small feature map with a fixed spatial dimension $H \times W$, what happens here is that each RoI window of shape h x w is divided into a $H \times W$ grid of sub-windows where the size of each is $(h/H) \times (w/W)$. Then the maximum values are used as the output of each sub-window. Thus, this layer can extract features of the same shape even when regions have unique shapes.

Each feature vector serves as input to a sequence of fully connected layers that originates two output layers. One output a probability distribution for each RoI across k object classes plus 1 which is the background. This is computed by a softmax function that converts a vector of real numbers into probabilities. The other layer outputs bounding box regression offsets for each k object class.

For this reason, they have two loss functions for training Fast R-CNN: one for classification, L_{clas}, and the other for regression, L_{reg}, weighted by a balance parameter λ (hyperparameter) as shown in Equation 3.1.

$$L = L_{clas} + \lambda L_{reg} \tag{3.1}$$

The loss for the classifier is calculated using the negative log probability of the true class (t), $L_{clas} = -logP_t$. The logarithm is used to ensure that the loss increases as the probability of the true class decreases. The result is then multiplied by -1 to make the loss a positive value. This means that the loss will be higher when the classifier produces a lower probability for the true class. For regression it was used a robust loss/smooth $L1$ loss (Equation 3.2 and 3.3).

$$L_{reg} = \sum_{i \in x,y,w,h} smooth_{L_1}(T_i, V_i) \tag{3.2}$$

where,

$$smooth_{L_1}(T_i, V_i) = \begin{cases} 0.5(T_i - V_i)^2 & if\ |T_i - V_i| < 1 \\ |T_i - V_i| - 0.5 & otherwise \end{cases} \tag{3.3}$$

Where i means for each class, T is the ground truth bounding box and V is the estimated bounding box. x, y, w, h as (x,y) are centre coordinates and weight and height of the bounding box.

3.1.1.2 Faster R-CNN

The factor that takes more time in the previous detection techniques is the CPU-based region proposal algorithms. To overcome that problem and improve the efficiency of this task, Ren et al. [31] extend the last algorithm to Faster R-CNN, which introduces a Region Proposal Network (RPN) as opposed to the Selective Search algorithm as the method to generate region proposals. This method consists of two components, the first one is a deep fully convolutional network that proposes region proposals and the second one is the Fast R-CNN as a detector network that uses these regions.

When using deep learning, it is difficult to generate a variable-length list of bounding boxes because the last layer is of a fixed size. RPN solves this problem by using "anchors" which are a pre-defined set of bounding boxes with specific aspect ratios and scales that are placed uniformly on the image. For each anchor, it is determined whether it contains an object and how it can be adjusted to better fit the ground truth object.

Then the extracted features and the bounding boxes with relevant objects go through RoI where they are extracted into a new feature map and the Fast R-CNN module uses that information to classify the bounding boxes into the given labels and adjust their coordinates to better fit the object. The convolutional features of the classification network are shared with the RPN component in order to reduce the computational cost.

Anchors. As said before, for each point in the obtained feature map, the network will have to learn whether an object is present in the input image, its location and estimate its size, and this is done by placing a set of anchor boxes in the original image in the output feature map, where this will reference the original image for each location on the output feature map from the CNN backbone. These anchor boxes indicate the possible presence of objects at that location in different sizes and aspect ratios.

The network slides each pixel in the output feature map and determines whether the corresponding anchors covering the input image contain objects. It then refines the coordinates of these anchors to create bounding boxes as a result.

Region Proposal Network. Is a fully convolutional network that takes the full image as input and generates a set of a predefined number of regions proposals accompanied by an objectness score, which indicates the likelihood that the region contains an object. *So how does this work?*

The RPN use the convolutional feature map obtained from the base network, which uses first a convolution layer with a certain number of channels and a certain kernel size and then uses two parallel convolutional layers using another kernel, whose number of channels depends on the number of anchors per point. Where one is a classification layer that outputs two predictions per

region, the estimated score of it being background and the estimated score of it being an actual object. The other one is a regression layer that outputs the coordinates of bounding boxes that are used to readjust the anchors to contain the object.

Li et al. [32], proposed a model called "DeepMitosis" method to detect mitotic cells on histopathological slides, the counting of which is nowadays performed manually by medical professionals. It is a task that is time-consuming and difficult to classify as there is a wide variety of cell appearances. Despite this, it is an essential task since its count is a biomarker of tumour aggressiveness in breast cancer diagnosis. They used Faster R-CNN as a detector and then used a deep verification model that verifies the detections made and removes false positives, in the end, a deep segmentation model was performed to segment the images and obtain the predicted bounding boxes. The detector architecture can be seen in Figure 3.1, where the first module is the RPN and the second is the Fast R-CNN, the region-based classifier. Where the green boxes and yellow boxes displayed in the detected image are respectively true positive and false positive.

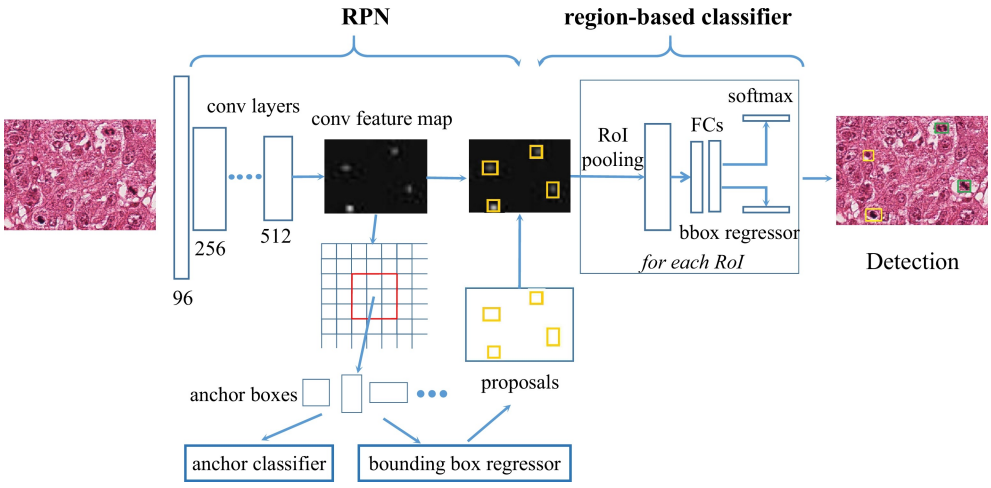


Figure 3.1: Detection model architecture from [32].

Other authors used a model Residual neural networks (ResNet) modified Faster R-CNN whit the goal of diagnosing brain normality and abnormalities, where identifies the object borders and then classifies them into hydrocephalus, hemorrhage and normal in brain CT images [33]. Xie et al. [34], developed a method to detect nodules in the lungs that produced a good performance. The method was based on the Fast R-CNN structure, with some modifications. They added two region proposal networks and a deconvolutional layer to identify potential lung nodules. Then, they trained separate models to detect nodules in different types of slices and merge the results for final classification. To decrease the number of false positives, they use a 2D CNN-based boosting architecture to differentiate target nodules from potential nodules. Any incorrectly classified samples are retained to improve the sensitivity of detecting pulmonary nodules. The final ranking is determined by combining the results from all networks.

3.1.2 One-Stage Object Detectors

One-Stage Object Detectors produce bounding boxes and class probabilities for objects in an image in a single step through the neural network without the region proposal step. YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) and RetinaNet are some of the most known one-stage detectors [28].

3.1.2.1 You Only Look Once (YOLO)

In 2016, Redmon et al. [35], presented the first version of YOLO. This model can directly detect multiple objects in an image with a single forward pass through the network. It can process an entire image and understand the context of the image, including information about the overall scene and specific details about each particular object within the image.

The algorithm starts by splitting the input image into a grid of cells (S x S), where each cell in the grid is responsible for predicting a set of possible bounding boxes with their confidence scores and the associated class probabilities as can be seen in Figure 3.2. The confidence score represents the presence or absence of an object in the bounding box, it can be computed with the multiplication of the probability of the present object and the percentage of the IoU.

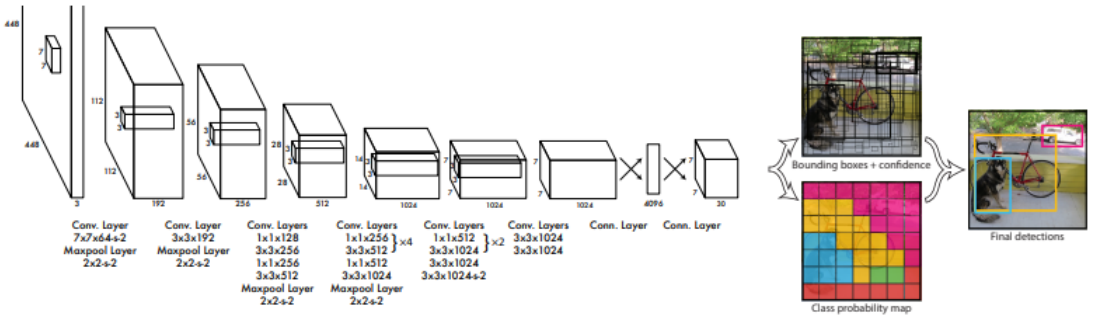


Figure 3.2: A representation of how the YOLO algorithm detects objects from [35].

So, each bounding box has a confidence score prediction that corresponds to the IoU between the predicted box and any ground truth box and five variables: (x, y) that represents the centre coordinates of the box and (w, h) that is the box width and height according to the image proportion. Each cell also estimates multiple conditional class probabilities, $\Pr(Class_i \mid Object)$, how likely a certain class is, given that the cell has an object. The product of this probability with the individual box confidence estimations is a class-specific confidence score for each box as shown in Equation 3.4.

$$P_r(Class_i|Object) \times P_r(Object) \times IoU_{pred}^{truth} = P_r(Class_i) \times IoU_{pred}^{truth} \tag{3.4}$$

The network architecture consists of 24 convolutional layers responsible for the feature extraction and it is followed by 2 fully connected layers responsible for predicting the output box coordinates and class probabilities, it uses 1 × 1 reduction layers followed by 3 × 3 convolutional

layers.

Training Process. The training process of the YOLO model involves optimizing an overall loss function composed of several components as demonstrated in the following:

$$(1) \quad \text{Loss}_1 = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{\text{obj}_{ij}} \left[(x - \hat{x}_i)^2 + (y - \hat{y}_i)^2 \right]$$

$$(2) \quad \text{Loss}_2 = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{\text{obj}_{ij}} \left[(\sqrt{w} - \sqrt{\hat{w}_i})^2 + (\sqrt{h} - \sqrt{\hat{h}_i})^2 \right]$$

$$(3) \quad \text{Loss}_3 = \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{\text{obj}_{ij}} \left(C_i - \hat{C}_i \right)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{\text{noobj}_{ij}} \left(C_i - \hat{C}_i \right)^2$$

$$(4) \quad \text{Loss}_4 = \sum_{i=0}^{S^2} l_{\text{obj}_{ij}} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

$$\text{Total Loss} = \text{Loss}_1 + \text{Loss}_2 + \text{Loss}_3 + \text{Loss}_4$$

Breaking down these components:

(1) Bounding Box Coordinate Predictions: The first two loss components are responsible for predicting the coordinates of the bounding boxes. To calculate this loss, it starts by considering the box coordinates (x, y) as the midpoint of the object in a specific cell grid i. The predicted midpoint x-value $\hat{x}_i$ is subtracted from the actual midpoint x-value, and then the result is squared, happening the same for the y-value by subtracting the predicted y-value $\hat{y}_i$. The l_{ij}^{obj} parameter is an identity function that takes a value of either 0 or 1. It is set to 1 if there is an object in cell i (where it's going to go from 0 to S^2 since it goes through every cell in the image), and if the bounding box j is responsible for outputting the highest IoU with any other predictor in the grid cell. After evaluating each cell to determine the presence of an object and summing up all these values., a λ_{coord}, constant set to 5 is multiplied by this loss part to give higher priority to accurate bounding box predictions.

(2) Bounding Box Size Predictions: The second loss component is similar to the previous one but focuses on predicting the width and height (w, h) of the bounding boxes. The only difference is the inclusion of the square root. This allows the loss to prioritize smaller bounding boxes as much as larger ones, as a very large bounding box would contribute significantly to the squared loss. The square root helps balance the impact of bounding box sizes.

(3) Object Probability Predictions: After the bounding box losses, it is added two more components that are responsible for predicting the probability of an object's presence in each grid cell C. For calculating this loss, if there is no object in a grid cell, the loss is multiplied by a

constant λ_{noobj}, which is set to 0.5. This ensures that the loss for cells without objects is not prioritized as much as those with objects.

(4) Class Probability Predictions: Finally, the last loss function focuses on class probabilities predictions. For each cell, is determined if there is an object present (using the identity parameter l_{ij}^{obj}). Then, it goes through each class $(c \in C)$ and calculates the sum-squared error of the class probabilities.

According to Redmon et al. [35] YOLO was able to achieve the goal of reducing the false positive classifications coming from background areas on an image which was a problem with the Fast-RCNN method. However, it presents some limitations such as the inability to accurately locate the small objects that appear in groups as the grid cell only provides two bounding boxes and limits the number of potential objects in the same cell. Lastly, while the algorithm is trained using a loss function that aims to improve detection performance, the loss function does not consider the size of each box. A slight error in a large bounding box may not significantly impact the overall performance, but a similar error in a small box can have a much greater impact. The main limitation of YOLO is that it struggles with the incorrect localization of objects, which is the main source of errors in the algorithm.

Currently, there are several versions of YOLO, in which the difference between them is mainly the architecture in order to improve the accuracy and speed of the algorithm. The original YOLO version outputted a single 7x7 grid. The earlier versions started using three different grids so that the model can learn what is useful: larger and smaller objects.

Ayan [27] proposed a system in which he used YOLO version 3 [36] and an image segmentation algorithm, GrabCut, to develop an automatic segmentation method of skin lesions in dermoscopic images, in which the precise location of the lesion in the image was performed and then its segmentation from the background. This technique was evaluated in already known datasets such as the PH^2 and the ISBI 2017, outperforming other deep learning-based approaches. YOLO version 4 [37] was used by Koo et al. [38] for detecting superficial fungal infections from microscopic images.

3.1.2.2 Single Shot MultiBox Detector (SSD)

The Single Shot MultiBox Detector [39] is a widely used one-stage object detector that aims to increase speed by eliminating the need for bounding box proposals and the feature resampling step which are parameters that lead to high computation and make object detection systems slow for real-time applications. It achieves this by using different convolutional filters for different aspect ratio detections to predict object classes and, at the same time, offset bounding box locations. These filters are then applied to multiple feature maps to execute detection at multiple scales. With these adjustments, SSD can achieve similar accuracy as Faster R-CNN while using lower-resolution images, making it faster. In comparison to other one-stage object detection methods such as YOLO version 1, it has been found to have real-time processing speed and even surpasses them in accuracy as measured by the mAP metric on the PASCAL VOC dataset [40].

This model uses a unified two-part network (Figure 3.3), a standard base network that is truncated before the last classification layer to acquire high-level features, then the fully connected layers are converted into convolutional layers. Next, this network is extended by adding an auxiliary structure of convolution feature layers of progressively decreasing resolutions which enable predictions of detections at multiple scales [13].

SSD uses a set of pre-selected boxes, known as default boxes, to match the ground truth bounding boxes across each feature map cell of different scales and aspect ratios in a convolutional way. These default boxes are distributed across different scales and aspect ratios, allowing the model to detect objects of different shapes and sizes. Then a filter is applied over the feature map, and for each default box, this filter predicts the bounding box coordinates, by providing offsets to the default box, to better fit the ground truth box. Additionally, it outputs confidence scores for all object classes. At training time, it is selected only the best default boxes that overlap most with each ground truth box, that is the ones that have the highest IoU, this strategy is called Matching.

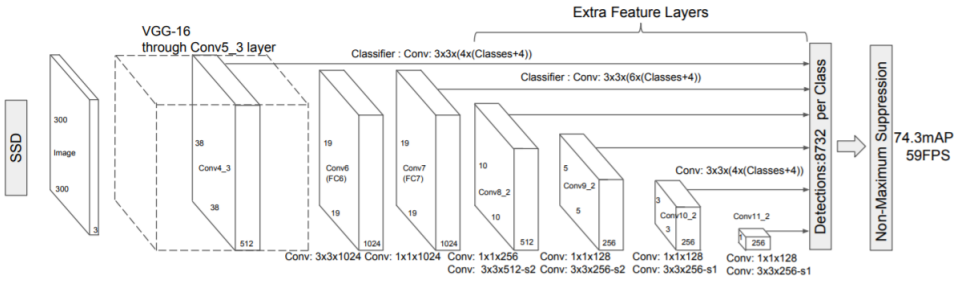


Figure 3.3: SSD model architecture from [39].

Hirasawa et al. [41] applied SSD to detect gastric cancer in endoscopic images which quickly processes a large number of images with a high level of diagnostic accuracy for clinical use. The non-detected lesions were shallow and difficult to distinguish from gastritis even for experienced health professionals, they were differentiated-type intramucosal cancers. A significant proportion of the false positive detections were gastritis with variations in colour or an irregular mucosal surface.

3.2 Object Detection in Medical Image Analysis

The object detection task is normally used in the medical image analysis field to identify early signs of abnormalities in patients [42]. A precise localization of these abnormal tissues is essential for the diagnosis. Some examples involve detecting breast lesions in ultrasound images [43], detecting lung nodules in Computed Tomography (CT) images [34] and detecting cancer cells in microscopic images [4].

According to the study developed by Li et al. [44] several object detection models are created for general images, such as detecting cars and pedestrians. However, medical images have unique characteristics that can lead to poor performance when using these general models. Nevertheless,

many objects of interest in medical images share similar features, such as similar shapes, sizes, blurry borders and single targets. Meanwhile, there is a huge variety of different features that the network has to learn to do efficient detection which can make it more difficult and challenging. For this reason, it requires a large amount of training and annotated data, which is not always possible because it is often expensive and time-consuming. This makes the training data hard to cover all situations and therefore it can be challenging for DNNs which needs a large amount of data, to achieve good performance on validation data. Furthermore, images taken with different medical equipment or in distinct environments may have varying characteristics, which can make it difficult to use the same model for all images. While there are methods to decrease these differences between data they can be complex to implement in practice. An ideal algorithm should be able to perform well despite these variations in data distribution.

This task can also be incorporated into other tasks such as semantic segmentation and classification tasks as a way of adding information about the shape of the anatomic object of interest to the loss function so that it can be applied to the original image and improve the performance of the model. The work of Chegraoui et al. [2] yielded good results in assessing the impact of the object detection task on a brain tumours segmentation model, when the models are applied to different domains, that is, validated in images with rare tumours and common tumours.

Chapter 4

State-of-the-art Review: Explainability in Deep Learning

There has been a significant increase in the use of Machine Learning algorithms to automate a diverse range of fields, including science, social work, and business. Among these algorithms, Deep Learning methods are their most popular offering in terms of accuracy. These models have a large number of parameters that contain the information learned by the training data but these parameters are difficult to isolate, making it challenging to explain how a specific decision was made to human users.

Explanations may not be necessary for certain AI applications but for other fields where a failure could lead to critical consequences like in medicine, law and finance, explanations are essential for users to understand what happens and therefore trust these systems [45]. This has led to the development of the area of Explainable Artificial Intelligence and with that, led to discussions in the literature about the distinctions between two linked terms: *interpretability* and *explainability*. According to Burkart and Huber [46] interpretability alone is not sufficient for understanding the inner workings of black box models, as it does not address all the issues that can arise when trying to understand these types of models. To build trust and gain meaningful insights into the decisions made by black box approaches, it is necessary to have explainability, which goes beyond simple interpretability. While explainable models are inherently interpretable, the reverse is not always the case.

Interpretability is defined as the ability to provide interpretations in terms that are understandable to a human. It is closely connected with the idea of understanding the reasoning behind the output of a model, according to Gilpin et al. [47] *"for a system to be interpretable, it must produce descriptions that are simple enough for a person to understand using a vocabulary that is meaningful to the user"*.

Explainability is concerned with how humans can understand and interact with an AI system.

This includes the capacity to provide additional information about how the system processes internal functionalities and makes decisions [5]. It is a way of verifying the output decision made by the algorithm.

A model considered the opposite of a black box is transparent and therefore interpretable if by itself is expressive enough to be human-understandable [48].

4.1 Reasons for Explainability

One of the main reasons that humans do not usually accept automated decision-making systems in critical domains is that they want to understand the reasoning behind a decision or at least receive an explanation for certain decisions. This is because humans are not prone to blindly trust these systems. For that reason, trust is a key factor in motivating explainability, along with other factors such as transferability, causality, informativeness, Fairness, accountability and the ability to make adjustments [46].

- **Trust**: This is a measure of confidence that is necessary for the prediction model's deployment. Human trust also requires an understanding of the model's strengths and weaknesses. Logic reasoning for a decision is safer than an extremely confident decision without any explanations.
- **Transferability**: For the predicted model be used with new data model must carry an understanding of future behaviour and must show that it generalizes well in certain contexts.
- **Causality**: Explainability, provides a sense of causality to the system's target group by revealing the underlying input-output association.
- **Informativeness**: To be implemented, it must be known whether the system serves its intended real-world purposes rather than just the purposes it was trained for.
- **Fairness**: The right to explanation requires decision-makers to present results clearly and conform to ethical standards.
- **Accountability**: This is *"the ability to determine whether a decision was made in accordance with ethical standards and to hold someone responsible if those standards are not met"* [49]. Explainability makes algorithms accountable for their actions and addresses the data-shift problem by making interpretable systems more responsible.
- **Adjustments**: Explainability enables adjustment of the prediction model by including domain knowledge and identifying failure modes.

It can be used to produce clear explanations and improve the expressiveness of the correlations between features in different areas of the data distribution in order to better comprehend AI fairness. It is possible to determine which features are correlated with a certain class, by using these techniques to trace the output predictions back to the input [50].

Additionally, incorporating xAI into the design of AI systems can provide an evolving model that takes into account previous parameters, explanations, problems, and improvements, reducing the potential for human bias. Still, it is crucial to be cautious when selecting the appropriate methods for explanation and to consider incorporating interpretability into machine learning models [51].

4.2 Taxonomy

There are many different ways to categorize and distinguish xAI techniques. The survey of Adadi and Berrada [1] classified xAI techniques based on Table 4.1.

Table 4.1: Adapted taxonomy and description.

Taxonomy	Description
Scope	Scope of explanations can be either local or global. Local methods focus on the explanation of individual data instances, while global methods aim to give an overall understanding of the entire model. [52, 53].
Methodology	The algorithmic approach can focus on the modifications input data instances and the ones which focus on the model architecture and parameters, perturbation-based and backpropagation-based respectively [53, 54].
Usage	The explainable method can be either integrated into the NN model itself (Intrinsic) or used as an external algorithm to provide explanations (Post-hoc) [54, 55].

The classification of certain methods as explainable or not is not so linear and unambiguous, since it can be based on one or more criteria and can be recognized as explainable methods for some and non-explainable for others.

- **Global Methods and Local Methods**: Local methods are intended to express the individual feature attributions of a single instance of input data, while global methods aim to understand the overall behaviour of the black box model by evaluating multiple inputs, this offers general relationships learned by the NN for example *"how much do features contribute to the output across the entire dataset"* [56].

- **Perturbation-based and Backpropagation-based Methods**: Explanation algorithms that are based on Backpropagation or Gradient-Based involve the use of information flow extracted by the backpropagation stage to understand neuronal relevance input-output. In contrast, Perturbation-based explanation methods only require a single forward pass through the network to produce attribution representations and do this by "perturbing" (making variations) on the feature set of an input instance. Both local and global explanations can be considered as either backpropagation-based or perturbation-based methods [5].

- **Intrinsic and Post-hoc Explanation**: Intrinsic model methods is a model that can easily explain its own decision-making process, as it contains interpretable elements. These models

can be easily understood by looking at the rules and logic used to make decisions (rule-based explanations), mathematical principles and through examination of specific examples that illustrate the decision-making process (example-based explanations). Post-hoc model methods refer to approaches where the output of a model, that has already been trained and performed, can be explained using a separate algorithm designed to interpret the behaviour of the original model, without sacrificing its accuracy, unlike intrinsic methods which, being interpretable, do not normally generate accurate predictors [5].

- **Model-specific and Model-agnostic Explanation**: These two concepts are associated with intrinsic and post-hoc explanation methods. Most model-Intrinsic methods are model-specific which are limited to certain classes of model architectures. The disadvantage is that, when it requires a specific type of explanation, the selections of models that lead to this are less flexible, excluding models that could better fit the output to the input data. For that reason, there has been significant research interest in developing model-agnostic explainable methods as they don't need to depend on a specific type of neural network, they can be applied to any model. This technique separates prediction and explanation. Model-agnostic explanations are usually post-hoc [1].

Currently, there is no universally accepted xAI taxonomy for classifying explainability techniques. Several published articles proposed their own categorization schemes, which can vary in terms of the level of specificity [57]. This can make it difficult to compare different approaches or to determine which technique may be best suited for a certain case. However, there is some widely accepted classification, that are local and global, intrinsic and post-hoc and model-specific and model-agnostic explanations that are covered in almost every work related to xAI.

4.3 Explainability Strategies

Currently, most works in the literature present post-hoc methods to provide explanations in DL models that are based on model simplification, feature relevance and visualization techniques [58].

The Local interpretable model-agnostic explanations (LIME) [59] method follows the model simplification technique. It explains the predictions of the complex model by building simpler models. This is done by perturbing the input data and training an interpretable surrogate model to learn the local behaviour of a global black box model's predictions. Then it uses the similarity between the original input and the changed input as a way to weigh the relevance of the simple model's explanations.

Shapley Additive explanations (SHAP) [60] follows the feature relevance technique where it seeks to enhance interpretability by computing additive importance values for each feature for individual predictions [61].

Class activation maps (CAMs) [62] are a method to visualize the regions of an image that are most important in the decision made by a CNN model. This is accomplished by applying global average pooling to the final convolutional feature map, then using the pooled feature maps as input

to a fully connected layer to produce a heatmap representation per class of the areas in the input image that had the most influence over the model decision. Gradient-weighted Class Activation Mapping (Grad-CAM) [63] and Grad-CAM++) [64] are generalizations of CAM that attempts to increase the explainability of networks by creating a localization map using the gradients of the target concept that flow into the last convolutional layer which highlights the regions in the image that were most important in making the prediction.

Simonyan et al. [65] introduce gradient-based saliency maps method to produce a saliency map for convolution networks that shows the absolute value of the gradient of the majority predicted class. The image pixels with the highest activation are highlighted, which correspond to the areas that are most important in the decision. These methods provide visualization of activations of individual neurons with high influence or overall feature attributions reshaped to the input dimensions but they are unable to explain why the model came to a particular decision [5].

Layer-wise relevance propagation (LRP) method [66] allowed to understand how a complex deep neural network is making its predictions by tracing back the predictions it made and by providing a heatmap that shows which pixels of the input image contributed to the output prediction. It applies a propagation rule that preserves the magnitude of the output as it is passed backwards through the layers of the network [67]. DeconvNet [68] is a similar method that uses a semantic segmentation algorithm to learn a deconvolution network and provides an understanding of the contribution of features during the classification task.

Kim et al. [69] introduce Concept Activation Vectors (CAVs) as a way to provide explanations of decisions made by NN in terms of high-level concepts that humans can easily understand. It allows to understand how sensitive an estimation is to a user-defined concept and how significant the concept is to the classification itself [67]. Saliency maps can't extract concepts like "car" that are not represented by pixels. This method attempts to solve this by presenting a translation between the input vector space and the high-level concept space. [56]

In summary, the most interpretability methods for explaining deep learning models are focused on image classification tasks, by using saliency maps, Grad-CAM, or DeconvNet as the main approach and LIME and SHAP are the most widely used methods in the state-of-the-art literature for visualizing feature relevance and feature interactions.

4.3.1 Intrinsically Explainable Methods

Intrinsic interpretable highly performing models are much more challenging to develop, however, some research proves that this task is possible by developing hybrid transparent and black box approaches. According to Arrieta et al. [58], incorporating previous knowledge in the form of constraints or logical statements in Knowledge Bases (KBs) can improve both the explainability and performance of DL models, which can make the learning process more resilient to errors in the training. Other strategies led to logical-probabilistic reasoning through learning and reasoning simultaneously with inferences and symbolic representations. In this way, the interpretability of a complex and opaque model can be enhanced by including knowledge from other domain origins. Examples of this are Deep Kalman filters, Deep Variational Bayes Filters and Structural

Variational Autoencoders. Another suggested methodology, different from hybrid models, was to combine the knowledge of transparent models with that of black box models. This can be done by using a semantic KB to guide the neural network or stacking ensembles of transparent and opaque models. Finally, the more recent technique developed is to create an interpretable version of an uninterpretable black box system.

Kean et al. [70] proposed a combination of deep neural networks with a transparent Case-Based Reasoning (CBR) module that involves using the knowledge and experiences obtained from past situations to understand and solve new cases. In this study, the idea is to explain the DNN model predictions of house prices through the use of examples. This is performed by studying the feature weights extracted from the network which are then applied to the CBR to retrieve nearest-neighbour cases to the given query case that is the data entry with some elements descriptions of a house [71].

Wachter et al. [72] introduced counterfactual explanations that involve identifying the factors that would be needed to change for a different prediction to occur. Counterfactual prototypes allow the understanding of the capabilities and limitations of a model, which can help to increase trust in the system and provide a more informed perspective for analysis [58].

Chen et al. [7] recently conducted a study that introduced the Concept Whitening (CW) mechanism that modifies a specified layer of the network by forcing predefined input information to converge at one unit, thereby inducing disentanglement of the latent space. When this module is integrated into a CNN model by replacing the batch normalization layer, the latent space is whitened, resulting in decorrelation and normalization of the latent space, and alignment of the latent space axes with the interest concepts. The CW mechanism provides a clearer understanding of how the network gradually learns concepts over different layers and gives insight into the extent to which a particular concept contributes to a specific prediction. Importantly, implementing the CW mechanism does not compromise the model's accuracy. Compared to post-hoc Concept Activation Vectors described before, the CW method is a superior approach as it performs actual concept disentanglement, placing the concept information along the latent space axes, rather than making assumptions that the different points in the latent space represent those concepts. This intrinsic aspect of the CW mechanism, which constrains the network instead of assuming extrinsic properties, makes it a promising approach for disentangling the latent space of neural networks.

Recent research has shown that prototypes can be used to provide explanations. Chen et al. [8] presented a network called prototypical part network (ProtoPNet) that performs the reasoning that usually a person do - *"this looks like that"*. Uses prototypical image parts to classify images by grouping training inputs into classes and selecting a representative example, or prototype, for each class. During testing, the network compares parts of the test image to these prototypes and produces a classification based on the similarities. This approach is strictly better than saliency maps because offers information about what the network is doing with those pixels, not just which pixels are used.

4.3.1.1 Prototypical Part Network

As mentioned earlier the ProtoPNet model is a method where explanations are generated during classification rather than created post-hoc, the network examines the input image in order to find prototypical parts using those to make its classification. This type of reasoning is used for example by radiologists when they want to identify whether what they are observing in an X-ray scan is a malignant or benign tumour. One can thus consider that this explainability strategy is very close to human reasoning and that it becomes interpretable because it is a transparent process when the system makes predictions.

In Figure 4.1, it is possible to observe the architecture of the model that was trained and evaluated with the dataset of bird species images as the first study case by the Chen et al. paper [8]. This architecture has three components, the first f consists of layers from convolutional neural network models such as VGG-16 [73], ResNet-34, ResNet-152 [74], DenseNet-161 or DenseNet-121 [75], the second g_P consists of a prototype layer and finally follows a fully connected layer h. For the first two types of convolutional layers is used ReLU as the activation function and for the last layer is used the sigmoid activation function.

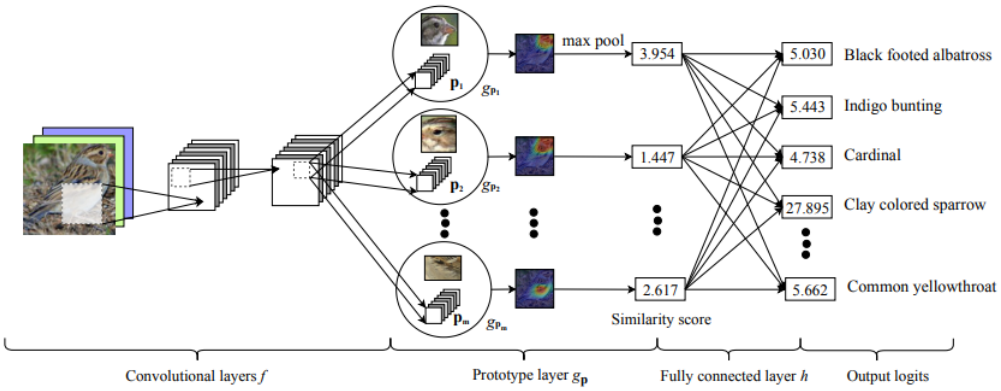


Figure 4.1: ProtoPNet architecture from [8].

Thus, in the first phase, the extraction of features $f(x)$ is performed, and then the network learns m prototypes $P = \{p_j\}_{j=1}^{m}$ whose size has to be smaller than the original image. Each one will represent a distinctive activation pattern in a convolutional output patch, which, in turn, corresponds to a prototypical image patch in the original pixel space. Consequently, each prototype, denoted as p_j, can be interpreted as the latent representation of some prototypical part of the image.

In the second phase, in the prototype layer g_P, it is computed the squared L2 distances between the *j-th* prototype, and all patches of the convolutional output that possess the same shape as p_j, this function as the name implies, is used to determine the separation between the input images' latent patches and prototypes. These distances are then transformed into similarity scores that result in an activation map that indicates the degree of presence of a prototypical part in the image and that can be expanded to match the input image's dimensions generating a heat map that highlights the region most similar to the learned prototype. Additionally, the activation map, consisting of

similarity scores from each prototype unit g_{p_j}, is consolidated into a single similarity score using global max pooling (Equation 4.1).

$$g_{p_j}(z) = max_{\tilde{z} \in patches(z)} log((\|\tilde{z} - p_j\|_2^2 + 1)/(\|\tilde{z} - p_j\|_2^2 + \varepsilon)) \tag{4.1}$$

In order to represent each class $k \in 1, ..., K$, it is necessary to establish the number of prototypes m_k required for each class. These prototypes should cover the crucial characteristics that are valuable in recognizing images belonging to class k.

Lastly, in the fully connected layer h, the weight matrix is multiplied by the prototype layer, m similarity scores to create the output logits product which is then normalized using the softmax function to obtain the projected probability for a certain image belonging to several classes.

In short, what can be understood when analysing the reasoning process of the network in deciding the class in Figure 4.1, is that the similarity score between the first prototype p_1, a Clay colored sparrow head and the most activated patch of the input image of a Clay colored sparrow is 3,954. On the other hand, the similarity score between the second prototype p_2, a Brewer's sparrow head and the corresponding most activated patch of the input image is only 1.447, this means that the ProtoPNet model finds that the head of a Clay colored sparrow has a stronger presence than that of Brewer's sparrow in the input image. After reviewing all the similarity scores, it is possible to verify that the first one mentioned will increase the probability related to the class Clay colored sparrow in the final prediction.

Training process. Goes through three steps: stochastic gradient descent (SGD) of layers before the last layer, projection of prototypes and convex optimization of the last layer.

The SGD is used to optimize the convolutional layer's parameters and the prototypes in the prototype layer while maintaining the weight matrix of the final layer, which is the fully connected layer, fixed. This is performed to make the neural network learn a meaningful latent feature space. This latent space is a lower-dimensional representation of the input data that captures important features for classification. In this feature space, important image patches that are essential for classifying images are grouped based on their $L2$ Euclidean distance. These patches are clustered around semantically similar prototypes representing the true classes of the images. Additionally, clusters that are centred at prototypes from distinct classes are kept separate from each other [8].

Assuming that $X = \{x1...xn\}$ represents the training images and $Y = \{y_1...y_n\}$ represents their corresponding class labels, the set D consists of pairs (x_i, y_i), where xi belongs to X and yi belongs to Y, the goal function for the initial optimization process can be expressed as follows:

$$\min_{P, w_{conv}} \frac{1}{n} \sum_{i=1}^{n} CrsEnt(h \circ g_p \circ f(x_i), y_i) + \lambda_1 Clst + \lambda_2 Sep \tag{4.2}$$

, where

$$
\begin{aligned}
Clst &= \frac{1}{n}\sum_{i=1}^{n} \min_{j:p_j \in P_{y_i}} \quad \min_{z \in patches(f(x_i))} \left\|z - p_j\right\|_2^2 \quad and \\
Sep &= -\frac{1}{n}\sum_{i=1}^{n} \min_{j:p_j \notin P_{y_i}} \quad \min_{z \in patches(f(x_i))} \left\|z - p_j\right\|_2^2
\end{aligned}
\tag{4.3}
$$

Where λ_1 and λ_2 are regularization hyperparameters. In addition to the minimization of cross entropy loss which ensures that the network learns to classify the images correctly, there are two terms that are intended to minimize the cluster cost (Clst) and the separation cost (Sep). The cluster cost promotes each training image to have at least one latent patch that is close to a prototype representing its class. In other words, it helps the network cluster similar patches for each class. On the other hand, minimizing the separation cost keeps prototypes far away from their incorrect classes [8].

In this stage, the (k, j)-th entry in the weight matrix of the last layer that is fixed corresponds to the weight connections between prototype units and class logits. These connections are adjusted based on whether a prototype belongs to the class being considered or not. Positive connections (weight values set to 1) are established between prototypes of the class being considered and the class's logits. Negative connections (weight values set to -0.5) are established between prototypes not belonging to the class and the class's logits. This adjustment of connections serves to influence the predicted probabilities of the model. Positive connections increase the probability that an image belongs to the corresponding class if it's similar to that class's prototype. Negative connections decrease the probability if the image's similarity to non-class prototypes is high.

By setting the last layer h with these defined values, the network is forced to learn a meaningful latent space because if a latent representation (latent patch) of an image from a specific class k is too close to a prototype of a different class (a non-class k prototype), it can have two consequences: decreased predicted probability, in another words the network might mistakenly predict the image as not belonging to class k due to its similarity to the non-class k prototype and an increased cross entropy loss reflecting the present misclassification.

As described before, separation cost is a term that encourages prototypes to be well-separated in the latent space, making them distinct from each other. The negative connection between a non-class prototype and the logits (output layer) of a specific class further enforces this separation. It means that the presence of a non-class prototype should not overly influence the prediction for the corresponding class. The design aims for class prototypes to capture specific semantic concepts that are characteristic of their respective classes. However, these prototypes should not be activated by images from other classes. If a prototype meant for class k represents a concept that is also found in a non-class k image, that non-class k image would mistakenly activate the class k prototype. This activation of a class prototype by a non-class image is penalized in two ways, the separation cost becomes less negative, indicating that the prototypes are not as well-separated as desired and cross entropy loss increases due to the negative connection. Since the non-class prototype is activating the class k prototype, it interferes with the prediction for class k and results in a higher loss.

The second training step is a projection of prototypes ("push operation"), which involves updating the prototypes using patches from the output of an image x that has the smallest distance to the prototypes. The update is performed as follows: for each prototype p_j belonging to class k, the model selects the patch Z from the patches of image x that belongs to class k and has the minimum distance:

$$p_j \leftarrow arg\ \min_{z \in Z_j} \left\| z - p_j \right\|_2, Z_j = \{\tilde{z} : \tilde{z} \in patches(f(x_i)) \forall is.t.y_i = k\} \tag{4.4}$$

This update ensures that the prototype layer is updated with prototypical parts that are closer to their respective classes. The patch that is most similar to the prototype is used for visualization purposes. Additionally, the activation value of the prototype must be at least at the 94th percentile of the activation values of the prototypical patches, as described in [8].

Finally, it is performed a convex optimization on the weight matrix of the last layer h. Here the purpose is to modify the last layer connections $w_h^{(k,j)}$ between class k and prototype unit j, such that when p_j does not belong to class k, the weight values $w_h^{(k,j)}$ approximate zero sparsity (initially set to -0.5). So the goal function can be expressed as:

$$min_{w_h} \frac{1}{n} \sum_{i=1}^{n} CrsEnt(h \circ g_p \circ f(x_i), y_i) + \lambda \sum_{k=1}^{K} \sum_{j:p_j \notin P_k} \left| w_h^{(k,j)} \right| \tag{4.5}$$

Where the regularization term λ, enforces sparsity by penalizing non-zero values of $w_h^{(k,j)}$ for j such that p_j is not in P_k. This optimization problem is convex because the parameters of the convolutional and prototype layers are fixed.

4.4 Critical Applications

Explainable Artificial Intelligence has gained significant interest in recent years due to its increasing frequency and important decision-making in several applications, in this part it is presented some of them. Regulatory authorities have already defined policies that provide stronger accountability for algorithmic decision-making [76]. AI systems in critical tasks must comply with safety and security requirements and have to prevent inequality and bias in their decisions. As already discussed, xAI has been applied in several critical areas, like healthcare decision systems, autonomous driving, criminal justice, security, natural language processing, anomaly and fraud detection.

4.4.1 Medical Applications

Looking at an example in the medical field, Barnett et al. [6] developed an interpretable DL model for classification of mass lesions in digital mammography. This approach was based on the CBR process, where the aim was to mimic the reasoning process of an actual radiologist who observes specific features of the image that are known to be relevant in the growth of breast lesions tissue, such as characteristics of the margins of masses in mammograms. They tested variations of the

ProtoPNet network with different baseline architectures (VGG-16 pre-trained on ImageNet, VGG-16 with Grad-CAM and VGG-16 with Grad-CAM++) incorporating detailed labels, changes to its training method, and the implementation of a multi-step reasoning process that begins with determining the feature related to mass margin and then uses that information to identify malignancy. They used AUROC (area under receiver operator characteristic curve) as a model performance metric and designed an activation precision metric to evaluate the model's interpretability. The ProtoPNet model with the VGG-16 pre-trained on ImageNet baseline (named IAIA-BL) obtained the best predictive performance and interpretability. This private dataset comprises 1136 cropped mammogram images of breast masses from 484 patients. They collected fine annotations from radiologists for 30 images to add more information to the dataset. The total of 1136 images consisted of five classes of mass margins: 125 spiculated, 220 indistinct, 171 circumscribed, 41 microlobulated, and 579 obscured. Although the last two classes were excluded due to the low amount of data and the fact that it is not a good indicator of classifying a malignant or benign lesion respectively.

Singh et al [77], proposed a negative-positive prototypical part network (NP-ProtoPNet) to classify chest X-ray images of Covid-19, pneumonia and normal patients. NP-ProtoPNet considers the positive and negative reasoning of humans ("this looks like that" and "this does not look like that") to perform the predictions, whereas ProtoPNet focuses only on positive reasoning. They did a comparison with different base models like VGG-16, VGG-19, ResNet-34, ResNet-152, DenseNet-121 and DenseNet-161 and with the original ProtoPNet model. It was verified that this model has superior performance in the measured metrics than ProtoPNet with all the baseline models but compared to the other baseline models it achieves better performance in accuracy only in VGG-16 and VGG-19. The CORD-19 database [78] is composed of 3875 and 1341 training images and 390 and 234 test images of pneumonia and normal patients respectively. The other database [79] consists of 930 images of COVID-19 patients including a variety of image modalities like CT scans, side X-rays and other types of images.

Table 4.2 shows a summary of the datasets used in some studies that implemented explainable deep learning models for classification and detection problems. Some of these are not publicly available, which makes it more difficult to train a DNN that requires a large amount of data to perform well.

Several studies have been conducted to develop explanatory algorithms for the classification and detection of various stages of Diabetic Retinopathy (DR) which is a condition associated with the diabetes disease and which causes vision loss. The conventional method of diagnosing DR involves manual analysis of retinal fundus images by ophthalmologists, which is a time-consuming process. Studies by Alghamdi [80], Chetoui et al. [81], and Shorfuzzaman et al. [82] applied explainability algorithms based on Grad-CAM approach to different datasets of retinal fundus images such as APTOS, Messindor, IDRiD, and others. The first study showed that the evaluation of the developed models in terms of their explainability was weak, as the models failed to detect the lesions related to DR. The study concluded that the VGG-16 model achieved significantly higher performance compared to the other models and provided more justified decisions. The second

Table 4.2: Datasets of published research in explainability and interpretability of deep learning algorithm in the medical field.

Article Reference	Dataset	Approach	Usage	Availability
[6]	Duke University	ProtoPNet	Classification	Unavailable
[77]	CORD-19, covid-chestxray	NP-ProtoPNet	Classification	Available
[80]	APTOS	Grad-CAM	Classification	Available
[81]	EyePACS, MESSIDOR, MESSIDOR-2, DIARETDB0, DIARETDB1, STARE, IDRID, E-ophtha, UoA-DR	Grad-CAM	Classification	Available
[82]	APTOS, MESSIDOR, IDRiD	Grad-CAM, SHAP	Classification	Available
[83]	ISIC	SHAP	Classification	Available
[84]	DRIVE, Retinal-Lesions, FGADR, IDRiD	Patho-GAN	Detection	Available
[85]	UK Biobank cardiac MRI	TCAV	Classification	Unavailable
[86]	Red Lesion Endoscopy	LIME	Classification	Available
[87]	DeepLesion	EigenCAM, Gradient SHAP, Layer Gradient SHAP, Occlusion, LIME	Detection	Available

study showed that in terms of explainability, it achieved different signs of DR using the DCNN Inception-Resnet-v2 model as a classifier. Niu et al. [84] proposed using Patho-GAN to separate the activation of neurons from the DR detector and present them visually, to better understand the signs and symptoms considered by the detector when making predictions.

Other applications are the detection of melanoma skin cancer [83], detection of lesions in CT images [87], classification of in-vivo Gastral Images [86] and classification on cardiac Magnetic resonance imaging (MRI) [85].

4.4.2 Other Relevant Applications

Autonomous driving systems are also a growing area in relation to xAI, for example in self-driving vehicles where the goal is to be able to drive in any unknown place [88]. Introducing explainability into these systems will help to monitor the perception of any potential errors that might occur. This, in turn, will enable the optimization of the AI system itself and make it more transparent to humans. Other prototype-based approaches have been published for scene understanding [89] and for the situation awareness of a self-driving car on a highway through the so-called vector of affordance indicators.

This technique has also been applied in the criminal justice system, according to Dressel and Farid [90], automated algorithms are used to predict the likelihood of crimes occurring in certain areas, as well as the likelihood that individuals will commit violent crimes and fail to appear for court hearings.

In the area of face recognition systems, Bansal et al. [91], introduce a method that helps identify the causes of system failures, such as the blurriness of a test image. This presents the

causes in a way that is easy for humans to understand and can be used to improve the design of features or gather more targeted training data during the development of the system.

4.5 Explainability in Object Detection

As seen previously, it is concluded that major explainable AI approaches are developed for deep learning models concerning classification tasks compared to the object detection task. Currently, hybrid object detection systems have been developed for applications such as autonomous driving, which is a continuous evolution field [92].

In the Medical field, there is a smaller amount of published work on explainability in object detection, but it is a domain that will benefit a lot from the implementation of these AI systems as an aid to the diagnosis made by health professionals. Sahatova and Balabaeva [87], present several explainable methods for the models that address the detection of objects on CT images. Explainable methods that fall within the domains of xAI such as feature visualization, Attribution maps and Latent space interpretation, including class-agnostic EigenCAM, Gradient-based SHAP, Occlusion, Layer Activation and LIME. This study yielded distinct results but demonstrated that the perturbation strategy is important but using grayscale images is ineffective for understanding CT scans. The gradient SHAP method provides information about how the system makes decisions, but the predicted bounding boxes depend on a final processing step of NMS threshold. However, some regions were not detected and others were false positive detections.

The IAIA-BL paper discussed in Section 4.4.1 despite being used for classification is the first work that applies Case-Based Reasoning using interpretable deep learning techniques to analyse medical images and is one of the main drivers for the study of inherently interpretable techniques but this time applying it to the object detection task in medical images.

Chapter 5

Materials and Methods

This section begins by presenting a description of the dataset selected for this study and the pre-processing steps applied to it. Following that, it offers an explanation of the proposed solution methods, along with supplementary experiments conducted as an extension of the proposal.

The proposed framework involves the integration of the ProtoPNet model to explain the classification decision in the object detection task. This activity focuses on creating an intrinsically explainable deep learning model involving a two-stage object detector.

Moving forward, the additional experiments were intended to establish a basis for comparison with the aforementioned framework where the first one similarly integrates a two-stage object detector and the ProtoPNet model but with a different configuration and the other one was the evaluation of the ProtoPNet baseline model on the adopted dataset so that it is possible to understand the impacts of these different approaches on the quality of explanations provided and the overall performance of their respective tasks.

Concluding the chapter, some details of the implementations are mentioned.

5.1 Dataset and Pre-processing

The selected dataset must respect the main requirement, which is to have all the image annotations necessary to carry out the object detection task, that is, to have the ground truth coordinates of the bounding boxes and the respective labels to which each object corresponds. It should be noted that the task of finding a medical dataset that met all these standards was quite difficult. Before selecting the ideal dataset, some alternative datasets were tested, but they didn't perform as successfully in the detection task.

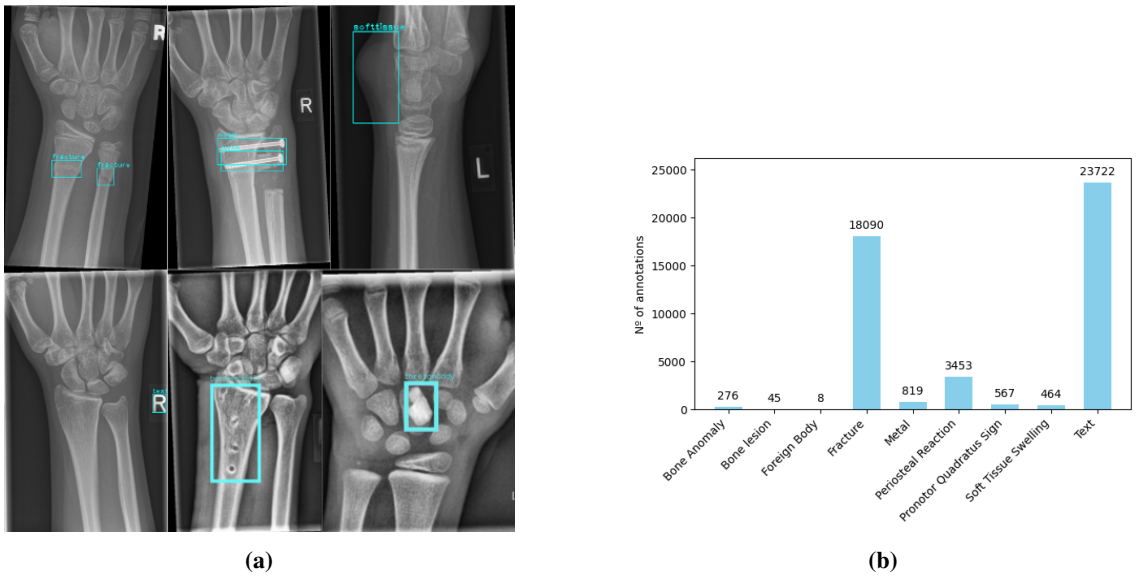
That said, the GRAZPEDWRI-DX dataset [93] was chosen. This dataset was collected from the Department of Paediatric Surgery of the University Hospital Graz between 2008 and 2018 consisting of paediatric trauma wrist radiographs. Involving a total number of 10,643 studies of 6,091 unique paediatric patients (2,688 females, 3,402 males, 1 unknown), 20,327 image samples,

covering lateral and posteroanterior projections in a 16-Bit grayscale PNG (Portable Network Graphics) format.

The dataset was annotated, by a group of paediatric radiologists, with 74,459 image tags and 67,771 labelled objects. They annotated the objects that represent pathologies like fractures and reactions or the presence of metal and foreign bodies by placing bounding boxes and lines.

There are nine different types of annotated objects, and each image can be associated with multiple objects. These objects include fractures, periosteal reactions, pronator quadratus sign (a specific radiological sign indicating swelling in the musculus pronator quadratus region [94]), soft tissue swelling (normally caused by trauma), foreign bodies, bone anomalies (such as drill holes or Madelung's deformities), metals (representing internal or external metal implants), bone lesions (referring to bone tumours like osteomas), text objects (representing any visible letter or text in the image, commonly used as laterality indicators like "L" or "R") [93].

Figure 5.1 shows some image examples with labelled bounding boxes and the distribution of annotations for each class.



(a) **(b)**

Figure 5.1: (a) Visualization of wrist X-ray examples with labelled bounding boxes. (b) Class distribution in the dataset.

They provide the data in PASCAL Visual Object Classes (VOC) [40] format and in YOLOv5 [95] format. Additionally, they provide a CSV file with all the filenames, image tags and basic patient information (sex, age) and a folder that contains some auxiliary Jupyter notebooks for annotations visualization and image post-processing.

Regarding the pre-processing, it was necessary to organize and structure the dataset to be used in the selected object detection model. To achieve this, certain labels were removed based on specific criteria. The labels 'boneanomaly', 'bonelesion', and 'foreignbody' were eliminated due to the limited number of annotations available, as can be seen in (Figure 5.1b), and also their poor performance when evaluated using the AP metric based on the reference paper for this dataset [93]. The label 'text' was also excluded since it holds no significance for this particular study focusing

on the detection and explainability of bone abnormalities. Additionally, the labels 'periosteal reaction' and 'pronator quadratus sign' were removed due to the challenges they presented in terms of explanation. Consequently, the study will only focus on three classes: 'fracture', 'metal', and 'soft tissue swelling'.

Analysing the distribution of classes in the dataset (Figure 5.2), it is evident that there is a significant class imbalance between the classes that will be used, in which there is a larger number of fracture samples (18090 samples) compared to the other classes. Consequently, to address this problem of class imbalance in the dataset, which may negatively impact model performance since there would not be a fair representation of classes in the training set, it was decided to employ the weighted random sampler strategy. This method ensures that, during training, each batch "sees" each class proportionally. In other words, it makes the probability of using a sample of a particular class proportional to the fraction of that class times the weight that is assigned to that class.

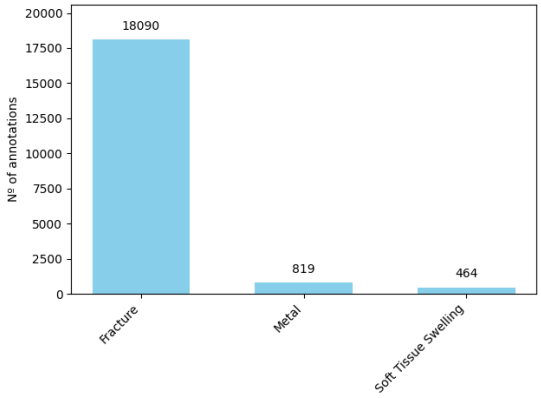


Figure 5.2: Number of annotations for the fracture, metal and soft tissue swelling class in the dataset.

Data augmentation techniques were applied to the images to mitigate the risk of overfitting. These techniques involved geometric transformations such as random vertical or horizontal flips, resizing with a minimum size of 600 and a maximum size of 1000, and affine transformations. In addition to these transformations, colour transformations were applied to the images such as random changes of contrast, brightness and random change of hue, saturation and value to highlight the most relevant features. The bounding boxes were also faced with the same transformations as the images, using the Albumentations package [96].

The data was divided into two sets: a training set and a test set. The distribution ratio between these sets was 80% for the training set and 20% for the test set.

5.2 Proposed Framework: iFaster-RCNN

This integrated solution, uses a two-stage object detector, the Faster R-CNN as the feature extraction backbone and the ProtoPNet model as the classification head that inherently provides explanations.

The developed strategy, named "interpretable-Faster-RCNN (iFaster-RCNN)" consists of using the Region Proposal Network component of the Faster R-CNN model to identify relevant features that would lead to new detections. These identified features are then passed through the ProtoPNet model which is only able to classify images with one detection (one object), as this explainability model was designed for image classification tasks. Using this approach, it can classify several regions of interest (RoIs) from a single image. The RoI pooling operation, which resizes the RoIs into the desired dimensions H×W, simplifies the handling of each RoI feature map by the explainability model. The RoI size was set to 7x7 so that it matches the convolution layer output dimensions employed in the original implementation of the ProtoPNet model. The complete model architecture can be seen in Figure 5.3.

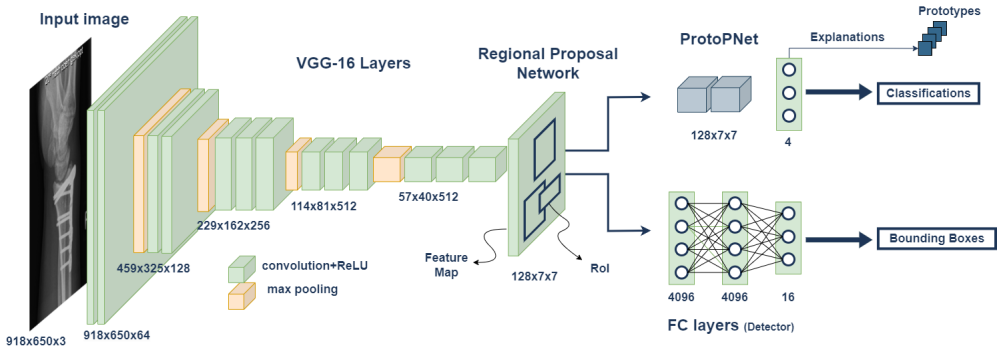


Figure 5.3: iFaster-RCNN architecture.

In terms of the prototype visualization, to confirm the correspondence between the features of RoIs with the predicted detection, the predicted bounding box from the iFaster-RCNN is responsible to crop the input image giving rise to the detected object. Furthermore, the cropped output is resized to a dimension of 112x112, so that it can be possible to visualize tiny detected objects and simultaneously ensure that it is aligned with its feature map.

After this process, the activation map is projected into the cropped image for subsequent visualisation of the prototype and its representation. All the patches of the cropped image input that are most activated by a particular prototype are saved in a PNG file. This procedure is demonstrated in Figure 5.4.

Training: In the training process, the network receives two inputs: Wrist X-ray images, and the ground truth bounding boxes with the respective label.

During this process, in order to train the integrated model that is training both object detection and the explainability task, the loss values from the Faster R-CNN model and ProtoPNet model were added together. Regarding, the regularisation hyperparameters values, λ_1 and λ_2, used in the ProtoPNet loss function computation (Equation 4.2), the decision was made to adopt the same values (0.8 and -0.08 respectively) as those employed by the authors of the original paper. This choice was influenced by the rationale provided in the existing state-of-the-art literature presented before.

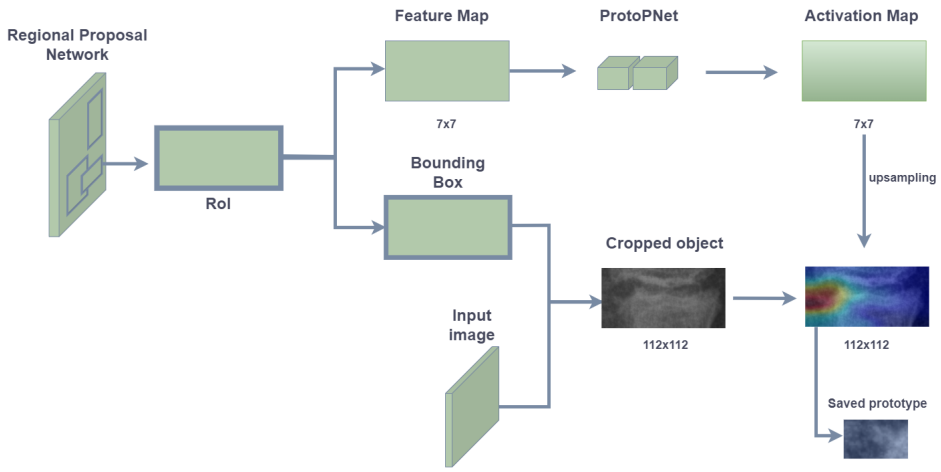


Figure 5.4: Scheme of the prototype visualisation procedure.

In this implementation, the background RoI was excluded. The fact that the RoI background features have a random nature, makes it more difficult to learn a meaningful latent representation, which negatively influences the calculation of the Separation Cost and Cluster Cost loss.

The push operation (Equation 4.4) for the visualisation of the prototypes and the update of the prototype vectors was initiated in specific epochs. At the end of each of these specific epochs, convex optimization (Equation 4.5) of the last layer step occurs, so that, in this way, the model performance is improved without modifying the latent space. Nevertheless, in addition to the selected epochs used for convex training, the final layer's weights should remain fixed throughout the remainder of the training procedure.

Test: During the testing process, the network is provided with wrist X-ray images without access to ground truth detections. In this scenario, the IoU threshold value was defined as 0.7.

Evaluation: This developed model was evaluated using the GRAZPEDWRI-DX dataset. The metrics used in this study were Average Precision and Mean Average Precision.

5.3 Additional Experiments in Support of the Proposal

5.3.1 Explainable Sequential Classifier Framework

In order to establish a basis for comparison, another experiment was performed. In this framework, the two networks were connected as illustrated in Figure 5.5. The Faster R-CNN receives the input image, returning the predicted bounding boxes involving objects of interest and these are used to crop the detected objects from the input images. The extracted cropped images are then resized to a dimension of 224 x 224 and fed to the ProtoPNet model.

Concerning, the training process, it can be unfolded in two sequential stages: first, the detector network is trained to generate accurate bounding boxes, then the ProtoPNet model is trained using the images cropped based on the predicted bounding boxes.

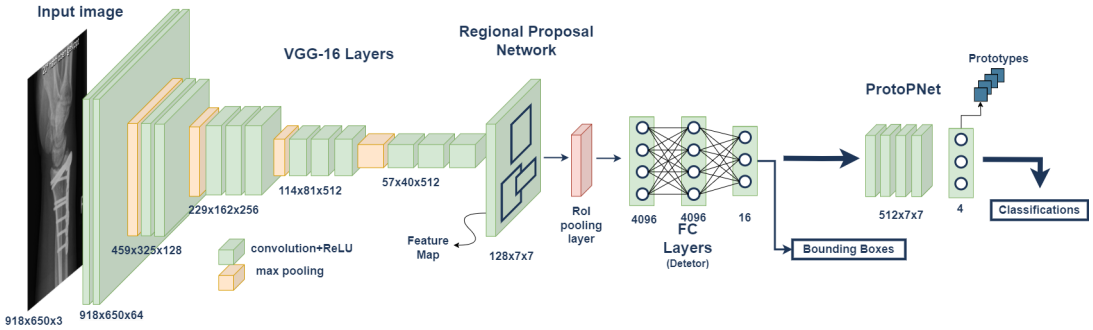


Figure 5.5: Explainable Sequential Classifier framework architecture.

5.3.2 ProtoPNet Baseline Model

The ProtoPNet model was replicated using the ground truth bounding boxes from the selected dataset, following the exact steps as specified in the original ProtoPNet model paper. This required cropping each image to extract the ground truth bounding boxes corresponding to the objects in the images as illustrated in Figure 5.6. These cropped regions were then associated with their respective class labels. After this step, the cropped images were fed into the model and resized to a dimension of 224 x 224. The base architecture used is the VGG-19.

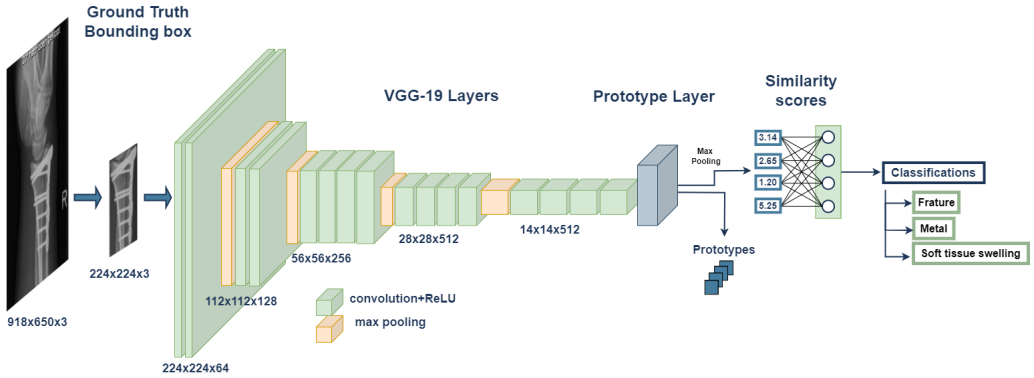


Figure 5.6: ProtoPNet architecture.

Addressing the dataset's class imbalance and the risk of overfitting already mentioned before, an oversampling technique together with data augmentation was employed in this experiment. In this way, is possible to achieve a balanced representation of classes in the training set and it can prevent the happening of overfitting by introducing variability in the training set. The Albumentations package facilitated this process by applying a range of transformations to the images. Geometric transformations, including random horizontal or vertical flips, and affine transformations with scale factors of 1.1 and 1.25 (allowing controlled stretching or compression) were implemented

while preserving the initial aspect ratio. Furthermore, colour-level transformations like brightness, contrast, and saturation adjustments were conducted.

The regularization hyperparameters values, λ_1 and λ_2, used in the loss function, are respectively 0.8 and -0.08 as used in the reference paper.

5.4 Implementation Details

i) **Faster R-CNN, iFaster-RCNN and Sequential models**

The Faster R-CNN model baseline, used to do the integration experiments, was based on the simplified implementation of a project available on GitHub [97] because it matches the performance reported in the original paper of Faster R-CNN. In addition to this, it achieves speed comparable with other implementations and it is memory efficient. The training procedure of the frameworks (iFaster-RCNN, Sequential and Faster R-CNN) included a meticulous fine-tuning of the hyperparameters, such as the learning rate (lr), choice of optimization algorithm, batch size and others. In Table 5.1 is possible to observe the specific range of values for these hyperparameters.

Table 5.1: Range values of hyperparameters used in iFaster-RCNN, Sequential and Faster R-CNN models.

Hyperparameters	Range of values
Learning Rate	0.001-0.00001
Learning Rate Decay	0.1
Weight Decay	0.0001-0.0005
Batch Size	1*
Optimizer	SGD/Adam

*The simplified implementation used of the Faster R-CNN model only allows a batch size of 1.

Concerning the visualization of the prototypes, it was used an IoU of 0.7 for object detection to make sure that there is an actual object inside the bounding box.

ii) **ProtoPnet model**

Just like with the other models, hyperparameter fine-tuning was also carried out when training the ProtoPNet model. In Table 5.2 are presented the range of values for these hyperparameters.

Table 5.2: Range values of hyperparameters used in ProtoPNet model.

Hyperparameters	**Range of values**
Learning Rate	0.001 - 0.0001
Learning Rate Decay	0.001 - 0.0001
Weight Decay	0.001 - 0.0001
Batch Size	1 - 100
Optimizer	SGD/Adam

It is important to mention what happens to ProtoPNet's layers during its training process, which, as mentioned before, consists of three stages. In the first training phase, only the last layer parameters are frozen, in order to make the backbone model learn meaningful latent space and prototype-related features without affecting the last layer's behaviour which is responsible for the final prediction.

In the step where the prototype push operation takes place, both parameters of the last layer and the layers before are frozen so that there is no modification of the learnt latent space when it reaches that point. What is important in this context is the act of updating the learned prototype vectors.

Finally, the convex optimisation of the last layer takes place in the epochs immediately following the push operation in which only the parameters of the last layer are unfrozen in order to reinforce the connection between this layer and the rest of the layers and by fixing the parameters from the convolutional and prototype layers, the learned latent space is not modified and an accurate prediction is made based on what has been learned.

Chapter 6

Results and Discussion

This chapter presents the results from the executed experiments and a critical discussion about them, including the comparison with current state-of-the-art methodologies.

It begins with a discussion of the results in terms of object detection performance, comparing the results obtained from the proposed framework with the alternative model and with the detector that serves as a baseline. As well as a comparison with the performance results that the baseline detector achieved with the adopted dataset in the original work.

This is followed by an evaluation of the results in terms of generating explanations for each one and finally a discussion of the classification performance and explainability of the intrinsic classification model used as a baseline in the frameworks developed.

This chapter also includes a section presenting the hyperparameters used in these experiments.

6.1 Hyperparameters Details

i) **iFaster-RCNN and Sequential models**

In Table 6.1 are presented the values of the hyperparameters used in the iFaster-RCNN and Sequential models frameworks. The Explainable Sequential Classifier Framework used the Faster R-CNN model pre-trained on the selected dataset with the same hyperparameters used on the proposed solution.

Table 6.1: Values of hyperparameters used by iFaster-RCNN and Sequential models.

Hyperparameters	Values
Learning Rate	0.001
Learning Rate Decay	0.1
Weight Decay	0.0001
Batch Size	1
Optimizer	SGD

ii) **ProtoPNet model**

Regarding the Hyperparameters utilized in training the ProtoPNet model, is possible to see the range of values employed in Table 6.2.

Table 6.2: ProtoPNet hyperparameters.

Hyperparameters	Values
Learning Rate	0.0001
Learning Rate Decay	0.001
Weight Decay	0.001
Batch Size	100/1
Optimizer	Adam

The developed models were carried out over 85 epochs. This number of epochs was essential to ensure that the training loss reached its fully converged state. Regarding the intervals of epochs selected to carry out the operation of updating the prototypes according to the projection of each prototype onto the nearest latent training patch from the same class, in the original ProtoPnet work, the authors used a range interval of 10, not specifying the reason for this choice. In this work, it was decided to perform the push operation at epochs 42 and 82, in the middle and at the end of the training. This choice was based on strategic considerations: updating prototypes in the middle of training allows the prototypes to adapt to the changing data distribution as the network's internal representations evolve, which ensures that the prototypes capture meaningful features of the data at a critical stage of learning. Updating the prototypes near the end of training enhances the network's ability to learn prototypes of the highest possible quality, given that the loss value is minimized at this stage.

6.2 Object Detection Performance

Regarding the results, Table 6.3 presents them considering the evaluated metrics and the inference time between the iFaster-RCNN, Sequential and Faster R-CNN (non-interpretable) models. In this table is possible to observe that the Faster R-CNN baseline model, tested on the selected medical dataset, was able to deliver a performance level equivalent to the reported in the original research of Faster R-CNN tested on PASCAL VOC 2007 (73.2% mAP), achieving a mAP of 70.5%.

The iFaster-RCNN model achieved a worse performance than the non-interpretable model, with a mAP decay of approximately 4.4%. This difference is expected due to the change in the architecture and the way the training process is performed. Alongside classification and regression tasks, an additional task involves generating explanations for the classification. This extra task has an adverse impact on the training process, thus affecting the overall performance of object detection.

Examining the obtained results of the Sequential model reveals a noticeable underperformance in comparison to the Faster R-CNN model, with a mAP value of 31.4%. This demonstrates that in addition to the negative impact of the aforementioned factors of changes in architecture and the

addition of the explainability task, there was also a propagation of the error associated with the prediction of the bounding boxes throughout the network that negatively influenced the overall detection performance.

Upon comparing the results of the two developed frameworks, there was a mAP decay of approximately 34.7%. This considerable difference is also unsurprising given the different training methodologies of the two models. The iFaster-RCNN model is trained together with the ProtoPNet model in which the loss functions are combined, on the other hand, the alternative approach employs a sequential procedure, in which the detector is trained first and then the ProtoPNet model is trained. Another contributing factor is that, in the iFaster-RCNN approach, the ProtoPNet model receives the feature map as input that gives rise to the classification, allowing it to acquire a representative latent space earlier than the other approach that receives input from the cropped projected bounding box.

Analysing the results of each class AP in the three developed models it is possible to notice that there is a significant difference between the results of the classes: 'Fracture' and 'Metal', and the 'Soft Tissue Swelling' class. This outcome may be due to the class imbalance, since, as previously mentioned, there are fewer samples of the 'Soft Tissue Swelling' class.

Concerning the duration required for inference in the models, it is possible to verify that there is a small, almost insignificant difference of milliseconds (ms) between the models. With the interpretable model achieving a lower inference time. The reason for the slightly higher inference duration in the developed approaches can be the number of parameters used by them and also the allocated memory.

Table 6.3: Performance results of the iFaster-RCNN model, Sequential model and Faster R-CNN baseline model and respective inference time.

			AP		
	Inference Time (ms)	mAP	Fracture	Metal	Soft Tissue Swelling
iFaster-RCNN	232.676	0.6606	0.8714	0.8103	0.2999
Faster R-CNN (non-interpretable)	243.498	0.7047	0.8775	0.8139	0.4228
Sequential	288.512	0.3137	0.8388	0.0909	0.0114

6.3 Explainability Results: iFaster-RCNN

Following the evaluation of the iFaster-RCNN model's performance, the generation of explanations with prototypes is tested. The illustrations depicted in Figures 6.1, 6.2 and 6.3, schematically represent the process of the explanations generated by the intrinsically explainable deep learning model that led to the final classification prediction of the detected object.

In each figure, is shown an object detection inference for each class, along with the explanation process that led to the classification of the detected object. This process allows the observation of the most highly activated patches in the input image by the first 3 prototypes. It also includes a heat map of the input image resulting from upsampling the similarity scores activation maps to the input image size. This heat map identifies the region (shown in red) of the input image that is most

similar to the learned prototype (part of the image that is activated by the prototype). Additionally, there is a self-activation map of the prototype and the resulting values of similarity scores. As previously mentioned, the similarity scores represent the distance between the presented prototype and the latent space representation of that class. From this value, it is possible to assess how close the presented prototype is to some patch in the input image.

An example of the explanation for the metal class can be found in Figure 6.1. The first and third prototypes correspond to metallic pins in the training image, while the second prototype corresponds to a screw. With this, it seems that for the generation of prototypes, the model used characteristic features such as the colour contrast between metals and the background as well as object contours. In X-ray images, metallic objects typically exhibit a bright white colour with high contrast against a less opaque background. Out of the three recognized prototypes, the first one has the highest similarity score of 4.626. This high score suggests that the first prototype is visually closer to the specified patch in the input image, however, upon closer examination, it is evident that other prototypes also closely resemble specific patches in their respective input images, even though they have lower similarity scores. In the second case, the prototype in the input image corresponds to the thickest part of the metal, similar to the most activated prototype in the training image. In the third case, both prototypes focus on the tip of the metal and share the same shape. Through this analysis, it can be stated that the iFasterRCNN model focuses on finding the most relevant parts of metals such as screws, pins and wires to generate its prototypes.

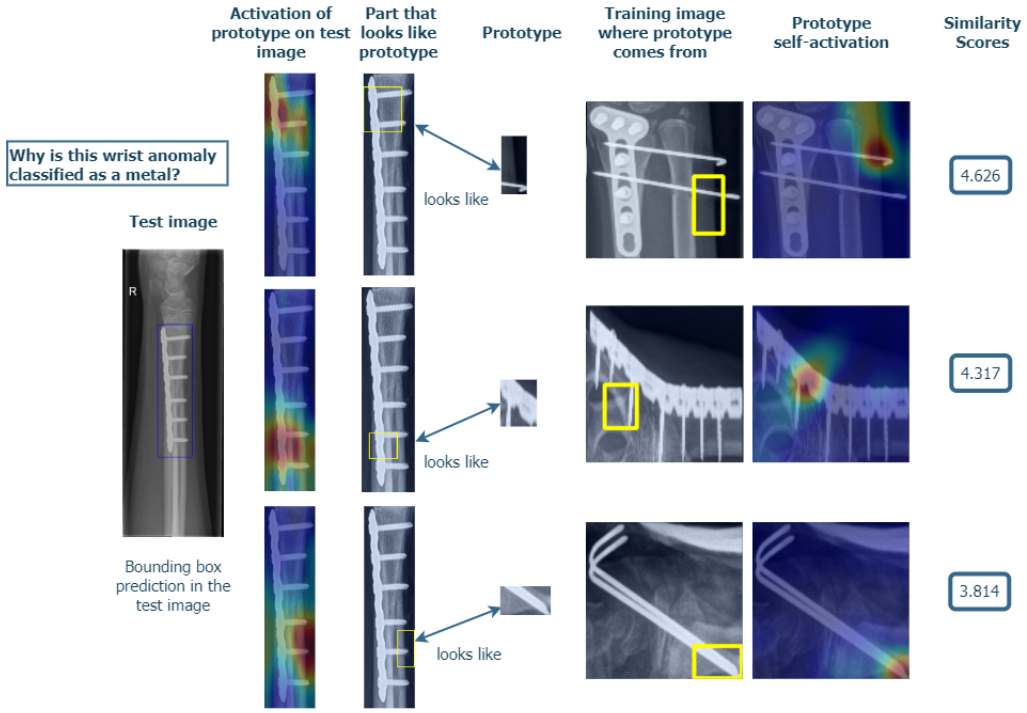


Figure 6.1: Wrist anomaly correctly classified as a metal.

Figure 6.2 shows an example of an explanation for the fracture class. In the first two prototypes generated, the deep learning model is considered as the most relevant features belonging to the

fracture class, the notable difference in colour in the location of the fracture, with no continuation of the bone, and also the contour/delineation of the fracture itself. The first and second prototypes have 4.987 and 3.274 respectively, however, it is possible to visualise better the second prototype in the corresponding patch activated in the test image than the first case, not respecting the rule that the higher the similarity score, the closer the prototype presented in the respective patch of the input image will be. Regarding the third prototype generated, is possible to understand the similarities between the prototypes because they both correspond to the delineation of the bone and that specific zone is brighter and contrasts with the background colour.

Finally, in Figure 6.3, the classification for the Soft tissue swelling class is explained. The model focused on specific features to create prototypes, mainly the contours of the swelling. Initially, these contours may appear indistinguishable without any processing. However, after appropriate processing, it becomes possible to observe a slight difference between normal tissue and swelling. The model also considered the edge between the background and the soft tissue as a relevant area for creating general prototypes. The first prototype, with a similarity score of 4.557, corresponds to the edge between the background and the tissue, similar to the test image. The remaining prototypes also accurately reflect the corresponding patches of the image.

In addition, Figure 6.4 shows other self-activation prototype images, which are highlighted by the red regions. Typically, the prototypes produced are compact in size, so they are presented alongside their corresponding original images to enhance comprehension.

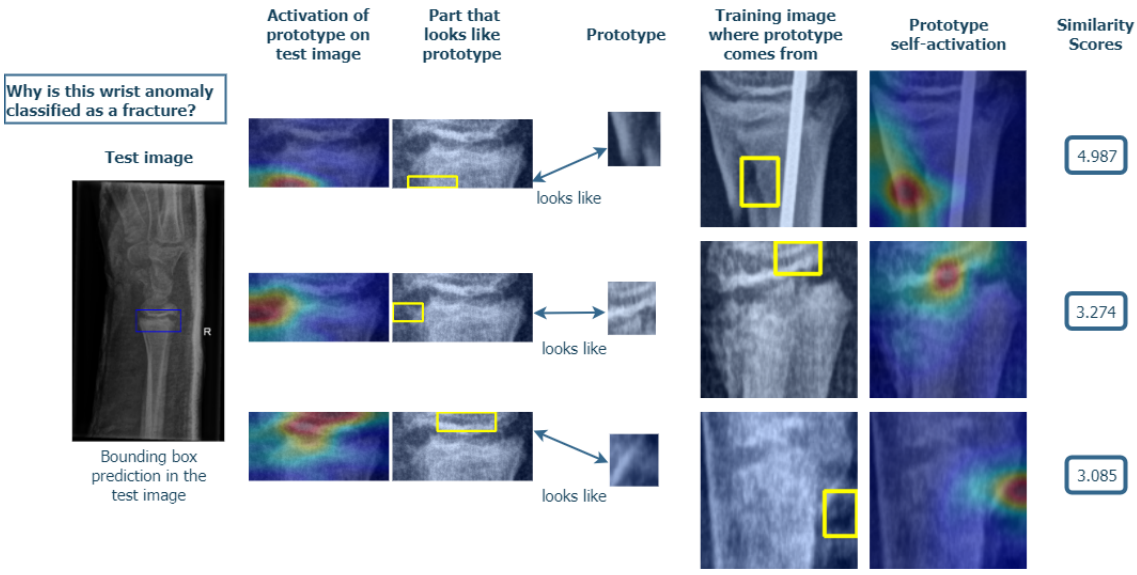


Figure 6.2: Wrist anomaly correctly classified as a fracture.

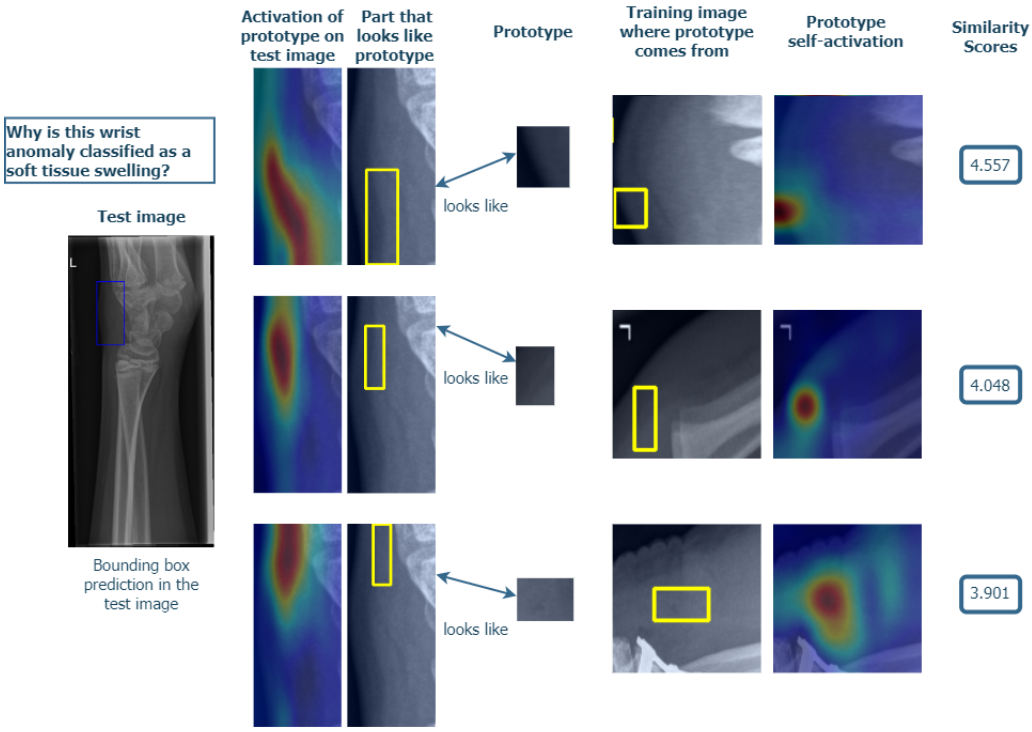


Figure 6.3: Wrist anomaly correctly classified as a soft tissue swelling.

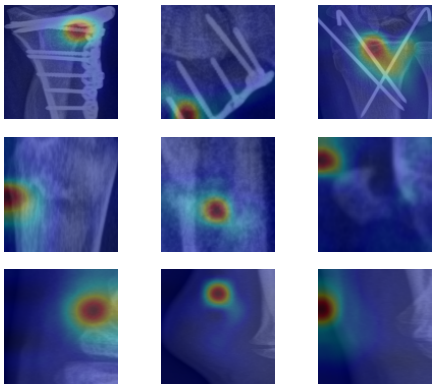


Figure 6.4: Examples of self-activation of the prototypes for the classes: metal (first row), fracture (second row) and soft tissue swelling (last row). The Red region indicates the highest relevance part.

6.4 Explainability Results: Explainable Sequential Classifier Framework

In the same way that the generation of explanations was evaluated in the iFaster-RCNN model, a similar evaluation was conducted for the sequential model. New illustrative panels (Figure 6.5) were generated using the same test images as those used in the proposed solution to allow a more effective comparison of the quality of the explanations. Unfortunately, due to the unsatisfactory performance of this model, there is a noticeable decline in the quality of the generated explanations.

The term "Class Connection" presented in the following illustrations refers to the weight connections between prototype units and class logits in the last layer as already described in Section 4.3.1.1. Recalling the content previously elucidated, these connections are defined initially in the first training phase of the ProtoPNet model ("Stochastic gradient descent (SGD) of layers before last layer") based on whether a prototype belongs to the class being considered or not. Positive connections (weight values set to 1) are established between prototypes of the class being considered and the class's logits. Negative connections (weight values set to -0.5) are established between prototypes not belonging to the class and the class's logits. These defined values were followed according to the original work of the ProtoPNet model. This adjustment of connections serves to influence the predicted probabilities of the model. Positive connections increase the probability that an image belongs to the corresponding class if it is similar to that class's prototype. Negative connections decrease the probability if the image's similarity to non-class prototypes is high.

Analysing Figure 6.5a, the first prototype of the soft tissue swelling class activates most strongly on a specific part of a soft tissue object and the most activated image patch of the given input image for this prototype, that is the patch that the model considers to look like the corresponding prototype, is a specific part of a metallic structure, with a similarity score of 3.361.

Given that the model accurately predicted that the object presented belongs to the metal class, it means that its inherent features align well with the characteristics of that class. However, the prediction is partially driven by the activation of a prototypical part from a different class, which exhibits high similarity to the given input image. Nevertheless, this prototype has a negative class connection of -0.514 to the predicted class, meaning that it will not contribute to the prediction for the correct class (-1.728 points contributed to the classification decision). In sum, it can be stated that, despite the prototype belonging to a different class, the model finds that the input sample shares enough features or characteristics with that prototype to activate it significantly but due to the value of class connection, this situation negatively impacts the predicted probability for the correct class.

In opposition to what happens in the second prototype, which is a prototypical part belonging to the metal class that activates most strongly on a specific patch of the test image achieving a similarity score of 3.234 with a positive class connection value of 0.968, thereby making a favourable contribution to the accurate prediction for the correct class.

In Figure 6.5c, a similar occurrence to the first example is observed in the second prototype: An activation of a non-class prototype by a class image with a similarity score of 2.293. This prototype comes from a training image belonging to the metal class but the model prediction for the displayed object was that it falls under the classification of soft tissue swelling.

Through these results, it is possible to understand that the model's behaviour is not desirable, as there are some examples of activation of a non-class prototype by a class image. This indicates that, at certain times, there was no reduction in the penalty for increasing the separation cost. In other words, in some cases the separation cost becomes less negative, meaning that the prototypes are not as well-separated as intended. Additionally, this behaviour negatively impacts the cross-entropy loss because it increases when there is a negative connection, this interferes with the

prediction for class k and results in a higher loss.

Thus, the model was unable to extract a meaningful latent space for each class, so that it would be able to distinguish which features belong to one class and which features belong to another, showing confusion when generating prototypes for a given class.

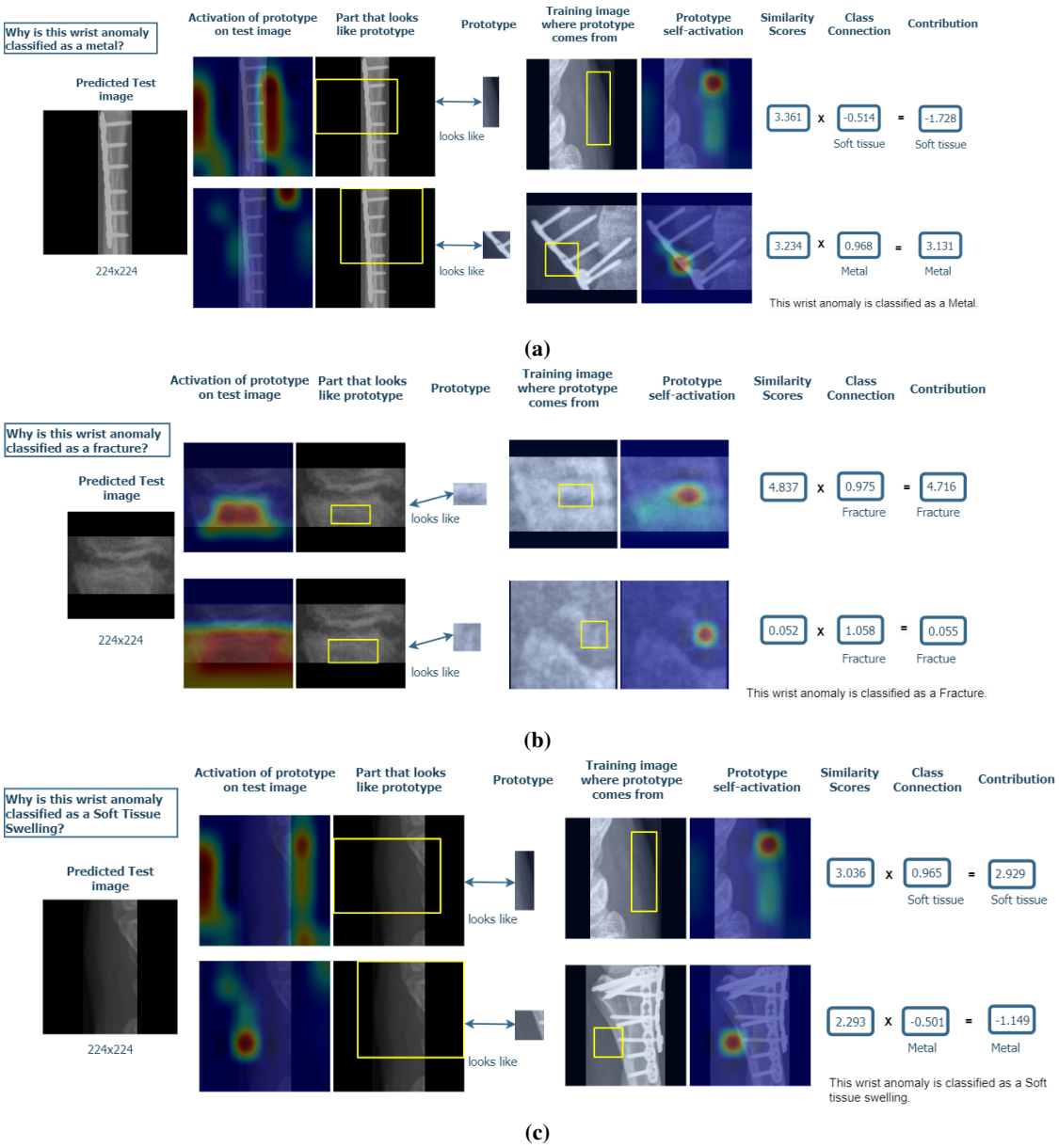


Figure 6.5: Sequential framework: (a) Wrist anomaly correctly classified as metal. (b) Wrist anomaly correctly classified as fracture. (c) Wrist anomaly correctly classified as soft tissue swelling.

6.5 Classification and Explainability Performance: ProtoPNet Baseline Model

Concerning the ProtoPNet performance as a Baseline model, it achieved an accuracy of 98.0% on both the training and test sets. However, despite the high accuracy of the model, and after further investigation into the model's training and testing, it was found that the value of separation cost exhibited a constant increase. This behaviour suggests that the network was not able to make all the latent patches of the training images distant from prototypes of different classes, consequently leading to poor learning of a meaningful latent space.

Explanations were generated with this performance to see the impact on the quality of the prototypes and, by analyzing the newly presented illustrative panel (Figure 6.6), it can be stated that, the network generated viable explanations. However, the presence of a high separation cost in both training and testing sets negatively influenced the effective calculation of similarity scores as can be seen in the unusual set of similarity scores in the three examples (Figure 6.6a, 6.6b and 6.6c), with the top 1 activated prototypes having a similarity score of 9.210 and the top 2 activated prototypes having a very low similarity score of 0.366, 0.375 and 0.366 respectively.

In an effort to improve the explainability performance of the model, and improve the separation cost behaviour, various experiments were conducted by adjusting the hyperparameters. One such experiment involved using a batch size of 1. Unfortunately, this adjustment did not yield better results. In this particular experiment, it was observed that the model reached an optimal solution during training with an accuracy of 33.7%. However, the accuracy during testing was inconsistent, fluctuating between 94.1% and 2.7%, indicating that sometimes the network can make correct predictions and other times not, which may be due to the weak representation at the class level in the test set, managing to have higher accuracy in the majority class. On the other hand, a significant improvement was noted in the behaviour of the separation cost, which decreased both in training and testing. However, looking at the prototype-based explanation panels (Figure 6.7), it was not enough to improve explainability results due to unstable performance.

As can be seen in the figures, the model classified these three test images as soft tissue swelling, and only the last one, truly belonged to that class. Besides these misclassifications, the network identified resemblances between each of these test images and prototypes from the metal class, indicating a score value of 9.210. This outcome underscores the model's inability to capture essential characteristics for distinguishing between different classes, thereby hindering its ability to establish reliable similarities between prototypes and specific image patches.

Concluding these observations, while the ProtoPNet model achieved an impressive accuracy performance of 78.0% and at the same time managed to generate reliable explainability results when tested with the original work dataset, the same does not occur using the dataset studied in this work. This discrepancy could be related to the difference in the number of images or the level of complexity between the two datasets.

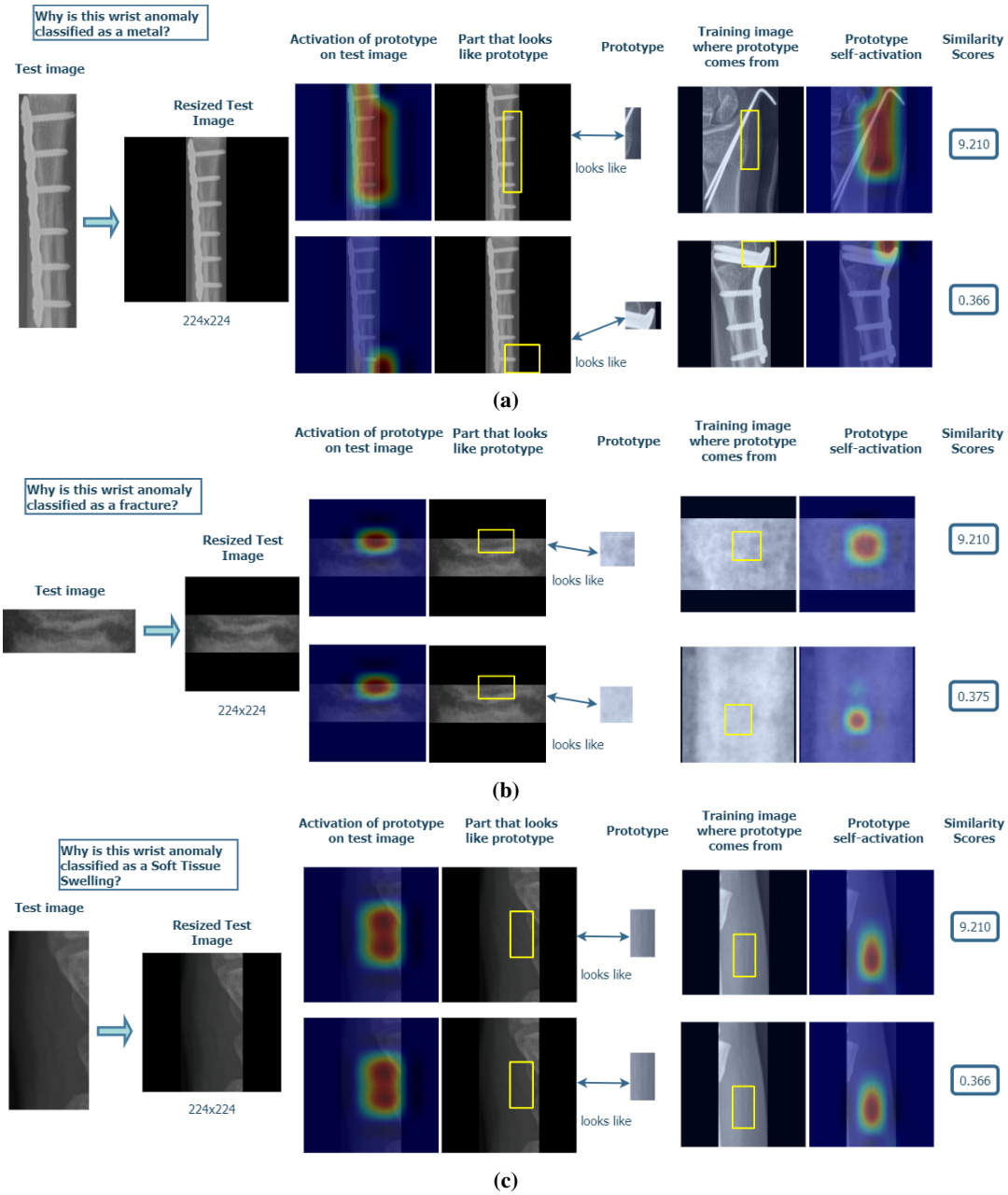
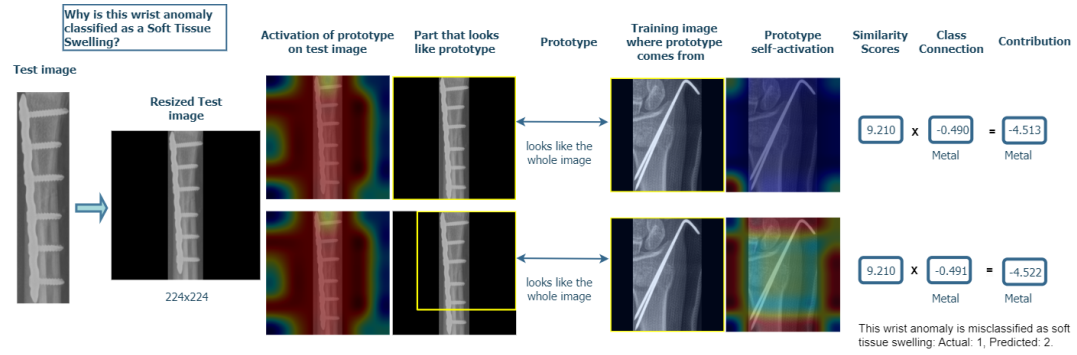
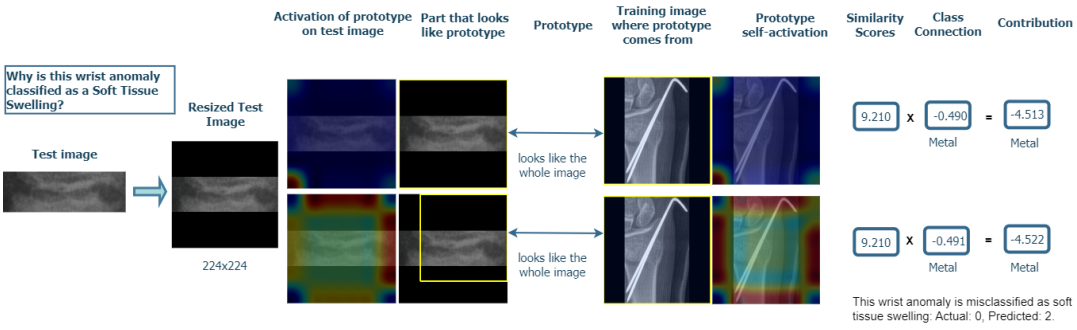


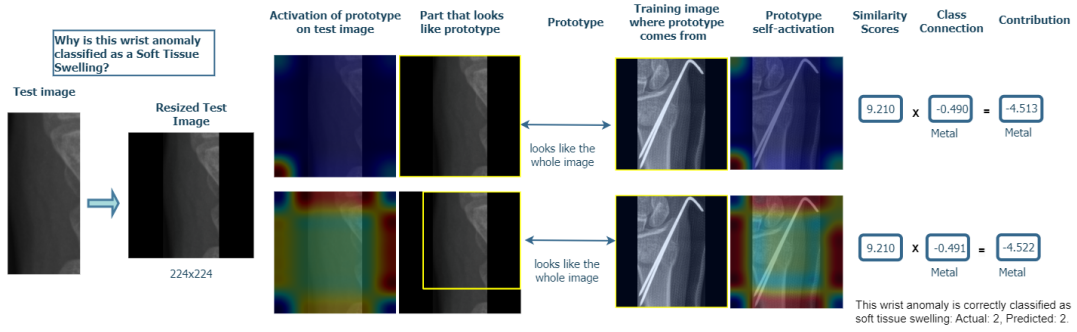
Figure 6.6: ProtoPNet model (batch size of 100): (a) Wrist anomaly correctly classified as metal. (b) Wrist anomaly correctly classified as fracture. (c) Wrist anomaly correctly classified as soft tissue swelling.



(a)



(b)



(c)

Figure 6.7: ProtoPNet model (batch size of 1): (a) Wrist anomaly misclassified as soft tissue swelling. (b) Wrist anomaly misclassified as soft tissue swelling. (c) Wrist anomaly correctly classified as soft tissue swelling.

Chapter 7

Conclusions and Future Work

The intrinsic explainable Deep Learning model developed in this dissertation project successfully demonstrates the ability to generate explainability during training without compromising the performance of the object detection task. This model achieves object classification by leveraging pertinent characteristics shared by objects of the same class that it has encountered previously, thereby emulating human reasoning rooted in past observations. Furthermore, this innovative framework aids in comprehending the reasons behind classification failures by providing visual explanations. Additionally, an interesting discovery emerges: the model employs distinct features specific to each class for the classification of Regions of Interest. This phenomenon is evident from prototype visualizations that illustrate changes in colour, contrast, and shape for individual objects.

Based on the proposed solution analysis of explanations, it can be stated that although the ProtoPNet model works correctly and effectively in explaining the classification of birds, cars, and persons, where the features used to extract the prototypical parts of the images are more perceptible and distinctive as in the case of the part of a bird's beak or a car's wheel. However, when dealing with medical data, particularly X-ray images, the situation is more complex. X-ray images are abstract in nature, with similar extracted features and a limited range of pixel values between black. This lack of variety makes the task of explainability more challenging in the medical domain.

Furthermore, the quality of prototypes generated by the integrated model is compromised due to the substantial heterogeneity of objects in this medical dataset. As already seen from the presented examples, this dataset contains diverse metal objects, such as wires, pins, plates, screws, and other variations, along with fractures and tissue anomalies, each unique in their presentation. In contrast, the bird dataset, which served as a case study in the original ProtoPNet report, does not share this diversity. The bird species exhibit a high level of uniformity in terms of their characteristics, shape, and posture. This homogeneity contributes to the superior quality of prototypes generated for this dataset.

A comparative assessment between the iFaster R-CNN approach, the alternative method, as well as the separated intrinsic classification model, reveals that the first one yields superior results

both in performance and explainability via prototypes. A potential reason for this outcome is the multitasked nature of the iFaster R-CNN model, which appears to operate as a regularization mechanism, mitigating the risk of poor model generalization.

Certain limitations are intrinsic to this new approach and could be considered in future research. The method's inference time is too slow, potentially making it unacceptable for real-time applications like autonomous driving, where immediate response is imperative. This constraint is tied to the detection model utilized, which lacks compatibility with real-world applications. However, given the present focus on healthcare applications in this study, the time factor is less crucial since the approach serves as an auxiliary diagnostic tool rather than requiring instantaneous outcomes. Another interesting future research involves exploring alternative backbone models, such as one-stage detectors like YOLO or SSD, which will decrease the inference time in the detections. Additionally, a deeper study should be carried out to evaluate the performance of this solution to optimize these results through experimentation with different backbone models such as ResNet and DenseNet.

Finally, the research conducted in this dissertation was important to make a significant addition to the realm of xAI in object detection with a particular focus on how it can be put into practical use in the domain of healthcare services. Increasing the explainability of deep learning models holds the potential to identify any latent flaws, thus facilitating accuracy enhancements.

References

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018. doi:10.1109/ACCESS.2018.2870052.

[2] Hamza Chegraoui, Cathy Philippe, Volodia Dangouloff-Ros, Antoine Grigis, Raphael Calmon, Nathalie Boddaert, Frédérique Frouin, Jacques Grill, and Vincent Frouin. Object detection improves tumour segmentation in mr images of rare brain tumours. *Cancers*, 13(23), 2021. doi:10.3390/cancers13236113.

[3] Klaus Toennies, Marko Rak, and Karin Engel. Deformable part models for object detection in medical images. *Biomedical Engineering Online*, 13(1):1–25, 2014. doi:10.1186/1475-925X-13-S1-S1.

[4] Carina Albuquerque, Leonardo Vanneschi, Roberto Henriques, Mauro Castelli, Vanda Póvoa, Rita Fior, and Nickolas Papanikolaou. Object detection for automatic cancer cell counting in zebrafish xenografts. *Plos one*, 16(11):e0260609, 2021. doi:10.1371/journal.pone.0260609.

[5] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020. doi:10.48550/arXiv.2006.11371.

[6] Alina Jade Barnett, Fides Regina Schwartz, Chaofan Tao, Chaofan Chen, Yinhao Ren, Joseph Y Lo, and Cynthia Rudin. A case-based interpretable deep learning model for classification of mass lesions in digital mammography. *Nature Machine Intelligence*, 3(12):1061–1070, 2021. doi:/10.1038/s42256-021-00423-x.

[7] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020. doi:10.1038/s42256-020-00265-z.

[8] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf.

[9] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018. doi:10.1155/2018/7068349.

[10] Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019. doi:10.1109/ACCESS.2019.2912200.

[11] Utkarsh Priyam. The evolution of parallel distributed processing. Accessed: 2023-05-23. URL: https://www.linkedin.com/pulse/evolution-parallel-distributed-processing-utkarsh-priyam.

[12] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning, 2018. arXiv:1811.03378.

[13] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021. doi:10.48550/arXiv.2106.11342.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[15] Yann LeCun, Larry Jackel, Leon Bottou, A Brunot, Corinna Cortes, John Denker, Harris Drucker, Isabelle Guyon, UA Muller, Eduard Sackinger, et al. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60. Perth, Australia, 1995.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017. URL: https://doi.org/10.1145/3065386, doi:10.1145/3065386.

[17] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning, 2014. arXiv:1410.0759.

[18] Johanna Pingel and Shyamal Patel. Introduction to deep learning: What are convolutional neural networks?, 2017. Accessed: 2023-05-29.

[19] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *CoRR*, abs/1807.05511, 2018. URL: http://arxiv.org/abs/1807.05511, arXiv:1807.05511.

[20] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. URL: https://doi.org/10.1007/s11263-013-0620-5.

[21] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. doi:10.1023/B:VISI.0000029664.99615.94.

[22] 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. doi:10.1109/CVPR.2005.177.

[23] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings. International Conference on Image Processing*, volume 1, pages I–I, 2002. doi:10.1109/ICIP.2002.1038171.

[24] Mark Everingham, Luc Van Gool, Christopher K I Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2010. doi:10.1007/s11263-009-0275-4.

[25] S.B. Jensen. How object detectors learn - what is object detection?, 2022. Accessed: 2022-12-01. URL: https://ambolt.io/how-object-detectors-learn/.

[26] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855, 2006. doi:10.1109/ICPR.2006.479.

[27] Halil Murat Ünver and Enes Ayan. Skin lesion segmentation in dermoscopic images with combination of yolo and grabcut algorithm. *Diagnostics*, 9(3), 2019. doi:10.3390/diagnostics9030072.

[28] Xin Lu, Quanquan Li, Buyu Li, and Junjie Yan. Mimicdet: Bridging the gap between one-stage and two-stage object detection. In *European Conference on Computer Vision*, pages 541–557. Springer, 2020. doi:10.48550/arXiv.2009.11528.

[29] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. doi:10.1109/CVPR.2014.81.

[30] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. doi:10.1109/ICCV.2015.169.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. doi:10.1109/TPAMI.2016.2577031.

[32] Chao Li, Xinggang Wang, Wenyu Liu, and Longin Jan Latecki. Deepmitosis: Mitosis detection via deep detection, verification and segmentation networks. *Medical Image Analysis*, 45:121–133, 2018. doi:https://doi.org/10.1016/j.media.2017.12.002.

[33] Kübra Uyar, Şakir Taşdemir, Erkan Ülker, Mehmet Öztürk, and Hüseyin Kasap. Multi-class brain normality and abnormality diagnosis using modified faster r-cnn. *International Journal of Medical Informatics*, 155:104576, 2021. doi:https://doi.org/10.1016/j.ijmedinf.2021.104576.

[34] Hongtao Xie, Dongbao Yang, Nannan Sun, Zhineng Chen, and Yongdong Zhang. Automated pulmonary nodule detection in ct images using deep convolutional neural networks. *Pattern Recognition*, 85:109 – 119, 2019. Cited by: 237. doi:10.1016/j.patcog.2018.07.031.

[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi:10.1109/CVPR.2016.91.

[36] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. doi:10.48550/arXiv.1804.02767.

[37] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. doi:10.48550/arXiv.2004.10934.

[38] Taehan Koo, Moon Hwan Kim, and Mihn-Sook Jue. Automated detection of superficial fungal infections from microscopic images through a regional convolutional neural network. *PLoS ONE*, 16(8 August), 2021. Cited by: 2; All Open Access, Gold Open Access, Green Open Access. doi:10.1371/journal.pone.0256290.

[39] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL: http://arxiv.org/abs/1512.02325, arXiv:1512.02325.

[40] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010. doi:10.1007/s11263-009-0275-4.

[41] Toshiaki Hirasawa, Kazuharu Aoyama, Tetsuya Tanimoto, Soichiro Ishihara, Satoki Shichijo, Tsuyoshi Ozawa, Tatsuya Ohnishi, Mitsuhiro Fujishiro, Keigo Matsuo, Junko Fujisaki, and Tomohiro Tada. Application of artificial intelligence using a convolutional neural network for detecting gastric cancer in endoscopic images. *Gastric Cancer*, 21(4):653 – 660, 2018. doi:10.1007/s10120-018-0793-2.

[42] Elyan, Eyad, Pattaramon Vuttipittayamongkol, Pamela Johnston, Kyle Martin, Kyle McPherson, Carlos Francisco Moreno-García, Chrisina Jayne, and Md. Mostafa Kamal Sarker. Computer vision and machine learning for medical image analysis: recent advances, challenges, and way forward. *Artificial Intelligence Surgery*, 2022. doi:10.20517/ais.2021.15.

[43] Moi Hoon Yap, Manu Goyal, Fatima Osman, Robert Martí, Erika Denton, Arne Juette, and Reyer Zwiggelaar. Breast ultrasound region of interest detection and lesion localisation. *Artificial Intelligence in Medicine*, 107:101880, 2020. doi:https://doi.org/10.1016/j.artmed.2020.101880.

[44] Zhuoling Li, Minghui Dong, Shiping Wen, Xiang Hu, Pan Zhou, and Zhigang Zeng. Clu-cnns: Object detection for medical images. *Neurocomputing*, 350:53–59, 2019. doi:https://doi.org/10.1016/j.neucom.2019.04.028.

[45] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. Xai—explainable artificial intelligence. *Science robotics*, 4(37):eaay7120, 2019.

[46] Nadia Burkart and Marco F. Huber. A survey on the explainability of supervised machine learning. *CoRR*, abs/2011.07876, 2020. doi:10.48550/arXiv.2011.07876.

[47] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. pages 80–89. Institute of Electrical and Electronics Engineers Inc., 2019. doi:10.1109/DSAA.2018.00018.

[48] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018. doi:10.1145/3236386.3241340.

[49] Finale Doshi-Velez, Mason Kortz, Ryan Budish, Chris Bavitz, Sam Gershman, David O'Brien, Kate Scott, Stuart Schieber, James Waldo, David Weinberger, et al. Accountability of ai under the law: The role of explanation. *arXiv preprint arXiv:1711.01134*, 2017. doi:10.48550/arXiv.1711.01134.

[50] Mengnan Du, Fan Yang, Na Zou, and Xia Hu. Fairness in deep learning: A computational perspective. *IEEE Intelligent Systems*, 36(4):25–34, 2021. doi:10.1109/MIS.2020.3000681.

[51] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 2019. URL: https://doi.org/10.1038/s42256-019-0048-x, doi:10.1038/s42256-019-0048-x.

[52] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015. doi:10.1371/journal.pone.0130140.

[53] Hugh Chen, Scott Lundberg, and Su-In Lee. Explaining models by propagating shapley values of local components. In *Explainable AI in Healthcare and Medicine*, pages 261–270. Springer, 2021.

[54] Jürgen Dieber and Sabrina Kirrane. Why model why? assessing the strengths and limitations of lime. *arXiv preprint arXiv:2012.00093*, 2020. doi:10.48550/arXiv.2012.00093.

[55] Julian Tritscher, Markus Ring, Daniel Schlr, Lena Hettinger, and Andreas Hotho. Evaluation of post-hoc xai approaches through synthetic tabular data. In *International symposium on methodologies for intelligent systems*, pages 422–430. Springer, 2020.

[56] Bas H.M. van der Velden, Hugo J. Kuijf, Kenneth G.A. Gilhuijs, and Max A. Viergever. Explainable artificial intelligence (xai) in deep learning-based medical image analysis. *Medical Image Analysis*, 79:102470, 2022. doi:https://doi.org/10.1016/j.media.2022.102470.

[57] Timo Speith. A review of taxonomies of explainable artificial intelligence (xai) methods. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 2239–2250, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531146.3534639.

[58] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. doi:https://doi.org/10.1016/j.inffus.2019.12.012.

[59] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2939672.2939778.

[60] 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. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[61] Plamen P. Angelov, Eduardo A. Soares, Richard Jiang, Nicholas I. Arnold, and Peter M. Atkinson. Explainable artificial intelligence: an analytical review. *WIREs Data Mining and Knowledge Discovery*, 11(5):e1424, 2021. doi:https://doi.org/10.1002/widm.1424.

[62] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[63] 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. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[64] Aditya Chattopadhay, 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. doi:10.1109/WACV.2018.00097.

[65] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. doi:10.48550/arXiv.1312.6034.

[66] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015. doi:10.1371/journal.pone.0130140.

[67] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1), 2021. doi:10.3390/e23010018.

[68] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[69] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677. PMLR, 10–15 Jul 2018. URL: https://proceedings.mlr.press/v80/kim18d.html.

[70] Mark T Keane and Eoin M Kenny. The twin-system approach as one generic solution for xai: An overview of ann-cbr twins for explaining deep learning. *arXiv preprint arXiv:1905.08069*, 2019. doi:10.48550/arXiv.1905.08069.

[71] Mark T. Keane and Barry Smyth. Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In Ian Watson and Rosina Weber, editors, *Case-Based Reasoning Research and Development*, pages 163–178, Cham, 2020. Springer International Publishing.

[72] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr, 2018. doi:10.48550/arXiv.1711.00399.

[73] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. doi:10.48550/arXiv.1409.1556.

[74] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi:10.1109/CVPR.2016.90.

[75] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018. doi:10.48550/arXiv.1608.06993.

[76] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017. doi:10.1609/aimag.v38i3.2741.

[77] 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. doi:10.1109/ACCESS.2021.3064838.

[78] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Nancy Xin Ru Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. CORD-19: The COVID-19 open research dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online, July 2020. Association for Computational Linguistics. URL: https://www.aclweb.org/anthology/2020.nlpcovid19-acl.1.

[79] Joseph Paul Cohen, Paul Morrison, and Lan Dao. Covid-19 image data collection. *arXiv 2003.11597*, 2020. URL: https://github.com/ieee8023/covid-chestxray-dataset.

[80] Hanan Saleh Alghamdi. Towards explainable deep neural networks for the automatic detection of diabetic retinopathy. *Applied Sciences*, 12(19), 2022. URL: https://www.mdpi.com/2076-3417/12/19/9435, doi:10.3390/app12199435.

[81] Mohamed Chetoui and Moulay A Akhloufi. Explainable end-to-end deep learning for diabetic retinopathy detection across multiple datasets. *Journal of Medical Imaging*, 7(4):044503, 2020. doi:10.1117/1.JMI.7.4.044503.

[82] Mohammad Shorfuzzaman, M. Shamim Hossain, and Abdulmotaleb El Saddik. An explainable deep learning ensemble model for robust diagnosis of diabetic retinopathy grading. *ACM Trans. Multimedia Comput. Commun. Appl.*, 17(3s), oct 2021. doi:10.1145/3469841.

[83] Mohammad Shorfuzzaman. An explainable stacked ensemble of deep learning models for improved melanoma skin cancer detection. *Multimedia Systems*, 28(4):1309–1323, 2022. doi:10.1007/s00530-021-00787-5.

[84] Yuhao Niu, Lin Gu, Yitian Zhao, and Feng Lu. Explainable diabetic retinopathy detection and retinal image generation. *IEEE Journal of Biomedical and Health Informatics*, 26(1):44–55, 2022. doi:10.1109/JBHI.2021.3110593.

[85] James R. Clough, Ilkay Oksuz, Esther Puyol-Antón, Bram Ruijsink, Andrew P. King, and Julia A. Schnabel. Global and local interpretability for cardiac mri classification. In Dinggang Shen, Tianming Liu, Terry M. Peters, Lawrence H. Staib, Caroline Essert, Sean Zhou, Pew-Thian Yap, and Ali Khan, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, pages 656–664, Cham, 2019. Springer International Publishing.

[86] Avleen Malhi, Timotheus Kampik, Husanbir Pannu, Manik Madhikermi, and Kary Främling. Explaining machine learning-based classifications of in-vivo gastral images. In *2019 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7, 2019. doi:10.1109/DICTA47822.2019.8945986.

[87] 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.

[88] Éloi Zablocki, Hédi Ben-Younes, Patrick Pérez, and Matthieu Cord. Explainability of deep vision-based autonomous driving systems: Review and challenges. *International Journal of Computer Vision*, 130:2425–2452, 10 2022. doi:10.1007/s11263-022-01657-x.

[89] Eduardo Soares, Plamen Angelov, Bruno Costa, and Marcos Castro. Actively semi-supervised deep rule-based classifier applied to adverse driving scenarios. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019. doi:10.1109/IJCNN.2019.8851842.

[90] Julia Dressel and Hany Farid. The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 4(1):eaao5580, 2018.

[91] Aayush Bansal, Ali Farhadi, and Devi Parikh. Towards transparent systems: Semantic characterization of failure modes. In *European Conference on Computer Vision*, pages 366–381. Springer, 2014.

[92] Yanfen Li, Hanxiang Wang, L. Minh Dang, Tan N. Nguyen, Dongil Han, Ahyun Lee, Insung Jang, and Hyeonjoon Moon. A deep learning-based hybrid framework for object detection and recognition in autonomous driving. *IEEE Access*, 8:194228–194239, 2020. doi:10.1109/ACCESS.2020.3033289.

[93] Eszter Nagy, Michael Janisch, Franko Hržić, Erich Sorantin, and Sebastian Tschauner. A pediatric wrist trauma x-ray dataset (grazpedwri-dx) for machine learning. *Scientific Data*, 9(1):222, 2022. doi:10.1038/s41597-022-01328-z.

[94] Julia Loesaus, Isabel Wobbe, Erik Stahlberg, Joerg Barkhausen, and Jan Peter Goltz. Reliability of the pronator quadratus fat pad sign to predict the severity of distal radius fractures. *World Journal of Radiology*, 9(9):359, 2017. doi:10.4329/wjr.v9.i9.359.

[95] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu , changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. ultralytics/yolov5: v3.1, Bug Fixes and Performance Improvements, 2020. doi:10.5281/zenodo.4154370.

[96] E. Khvedchenya V. I. Iglovikov A. Buslaev, A. Parinov and A. A. Kalinin. Albumentations: fast and flexible image augmentations. *ArXiv e-prints*, 2018. arXiv:1809.06839.

[97] Chenyuntc. A simplified implementation of faster r-cnn that replicate performance from origin paper. https://github.com/chenyuntc/simple-faster-rcnn-pytorch, 2018.