FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Producing Decisions and Explanations: A Joint Approach Towards Explainable CNNs

Isabel Cristina Rio-Torto de Oliveira

MASTER'S DEGREE IN ELECTRICAL AND COMPUTERS ENGINEERING

Supervisor: Prof. Doutor Luís Filipe Pinto de Almeida Teixeira

Co-Supervisor: Doutor Kelwin Alexander Fernandes Correia

October 1, 2019

Resumo

As Redes Neuronais Convolucionais, bem como outros modelos de *Deep Learning*, têm conseguido atingir resultados notáveis em tarefas como classificação e deteção de objetos. Contudo, estes modelos permanecem, em grande parte, caixas-negras. Com o uso generalizado de tais redes em cenários reais e com a crescente exigência do direito à explicação, já incluída nas novas políticas do Regulamento Geral de Proteção de Dados, especialmente em áreas altamente reguladas como medicina e justiça, gerar apenas decisões tornou-se insuficiente. Para ser possível operar neste novo enquadramento legal, os modelos de *Machine Learning* têm de ser explicáveis, isto é, compreensíveis pelos seres humanos, o que implica serem capazes de apresentar as razões por detrás das suas decisões.

Os dados são o ingrediente principal em todos os *pipelines* de *Deep Learning*. No entanto, é incrivelmente difícil encontrar conjuntos de dados anotados e é muito caro recolher e anotar um. Além disso, a explicabilidade de um modelo ou de uma instância específica depende do utilizador e do contexto de operação. Como tal, é imperativo que novas soluções sejam capazes de gerar explicações sem precisar de dados anotados.

Enquanto a maior parte da literatura se foca em métodos pós-modelo, este trabalho consiste numa nova arquitetura intra-modelo baseada numa rede neuronal convolucional, composta por um explicador e um classificador. Esta arquitetura gera não só uma *class label*, mas também uma explicação visual de tal decisão, sem a necessidade adicional de dados anotados para treinar o explicador. O modelo é treinado *end-to-end*, tendo o classificador como entrada não só as imagens do conjunto de dados, mas também a explicação resultante do explicador, permitindo, assim, que o classificador se foque apenas nas áreas relevantes de tal explicação. Um processo de treino decomposto em três fases, é, também, proposto, bem como funções de perda específicas, que regularizam as explicações e promovem propriedades desejáveis, como esparsidade ou contiguidade espacial. Duas abordagens alternativas são desenvolvidas, uma não supervisionada e outra fracamente supervisionada, em que ambas permitem ao explicador produzir explicações sem precisar de anotações adicionais dos dados.

Para testar a arquitetura proposta, foi desenvolvida uma ferramenta de geração de dados sintéticos, que permite anotação automática e a criação de imagens de fácil compreensão que não exigem o conhecimento de um especialista para as explicar, acelerando, assim, o processo de avaliação humana qualitativa das explicações produzidas.

A arquitetura foi, também, validada num conjunto de dados médicos real e os resultados obtidos mostram que, de facto, esta é capaz de produzir explicações visuais para as decisões do classificador sem supervisão. Estes também comprovam que esta pode ser utilizada com qualquer classificador, desde que sejam feitas as conexões necessárias ao explicador. Além disso, este método constitui um melhoramento face a métodos do estado da arte, especialmente no que se refere à explicação de instâncias negativas em casos sintéticos. Por fim, as explicações produzidas num conjunto de dados médico de cancro do colo do útero encontram-se alinhadas com conhecidas caraterísticas visuais associadas a casos cancerosos, como mudanças na morfologia, cor e contornos dos tecidos do colo do útero, que são, assim, indicativas de áreas lesionadas.

Abstract

Convolutional Neural Networks (CNNs), as well as other Deep Learning (DL) models, have shown remarkable performance on tasks like classification and object detection. However, these models largely remain black-boxes. With the widespread use of such networks in real-world scenarios and with the growing demand of the right to explanation already included in the new General Data Protection Regulation (GDPR) policies, especially in highly-regulated areas like medicine and criminal justice, generating accurate predictions is no longer enough. In order to operate within this new legal framework, Machine Learning (ML) models have to be explainable, i.e., understandable to humans, which entails being able to present the reasons behind their decisions.

Data is the core ingredient in every DL pipeline. However, it is incredibly difficult to come by labelled datasets and it is highly expensive to collect and annotate one. Moreover, the explainability of a model or a particular instance is user- and domain-dependent. As such, it is imperative that new solutions are able to generate explanations without needing labelled data.

While most of the literature on explainable ML focuses on post-model methods, this work comprises a novel in-model CNN architecture, composed by an explainer and a classifier. This architecture outputs not only a class label, but also a visual explanation of such decision, without the need for additional labelled data to train the explainer. The model is trained end-to-end, with the classifier taking as input not only images from the dataset but also the explainer's resulting explanation, thus allowing for the classifier to focus on the relevant areas of such explanation. We also propose a three phase training process with custom loss functions that regularise the produced explanations and encourage desired properties, such as sparsity and spatial contiguity. Two alternative approaches are proposed, an unsupervised and a weakly supervised approach, both allowing the explainer to produce explanations without the need for additional annotations of the data.

To test the proposed architecture, a synthetic data generation tool was developed, that allows for automatic annotation and creation of easy-to-understand images that do not require the knowledge of an expert to be explained, thus expediting the human qualitative evaluation process of the produced explanations.

Our joint approach was also validated on a real medical dataset and the obtained results show that, in fact, it is able to produce visual explanations for the network's decisions without supervision. They also show that this architecture can be employed with any classifier, provided that the necessary connections to the explainer are made. Furthermore, this approach improved on state of the art methods, especially when trying to explain negative instances on synthetic cases. Finally, the produced explanations on the medical cervical cancer dataset were aligned with known visual features associated with cancerous cases, such as morphology, colour and contour changes of the cervix tissue, and thus are indicative of injured areas.

Agradecimentos

Aos meus orientadores, Prof. Doutor Luís Teixeira e Doutor Kelwin Fernandes, por todo o apoio, orientação e oportunidades que me proporcionaram ao longo deste percurso.

A todos os meus amigos que sempre me acompanharam e tiveram paciência para me ouvir. Aos amigos de sempre, Mariana, Nuno, Sara e Daniel, obrigada pelo apoio. Ao Nuno, à Rita e ao Pedro, muito obrigada e desculpem as secas gigantes que vos dei a falar sobre este trabalho.

Aos meus avós, que embora já não fisicamente comigo, continuam a ser uma enorme fonte de inspiração, orgulho e ajuda.

Aos meus pais por serem um excelente exemplo de profissionalismo e humildade, pelo apoio incondicional e inabalável paciência que tiveram, não só durante a elaboração deste trabalho, mas ao longo destes árduos 5 anos: o meu mais sincero agradecimento.

Por fim, à Minie.

Isabel Rio-Torto

"Before the vision, comes the question."

InspiroBot[1] - I am an artificial intelligence dedicated to generating unlimited amounts of unique inspirational quotes for endless enrichment of pointless human existence.

[1] https://inspirobot.me

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AI	Artificial Intelligence
ALE	Accumulated Local Effects
ANN	Artificial Neural Network
AUC	Area Under the Curve
CAV	Concept Activation Vector
CIN	Cervical Intraepithelial Neoplasia
CNN	Convolutional Neural Network
COMPAS	Correctional Offender Management Profiling for Alternative Sanctions
CV	Computer Vision
DARPA	Defense Advanced Research Projects Agency
DL	Deep Learning
DNN	Deep Neural Network
DTD	Deep Taylor Decomposition
EU	European Union
FDA	Food and Drug Administration
FN	False Negative
FP	False Positive
GAM	Generalised Additive Model
GD	Gradient Descent
GDPR	General Data Protection Regulation
GLM	Generalised Linear Model
GPU	Graphics Processing Unit
IAPR	International Association for Pattern Recognition
IbPRIA	Iberian Conference on Pattern Recognition and Image Analysis
ICE	Individual Conditional Expectation
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
K-NN	K-Nearest Neighbours
LIME	Local Interpretable Model-agnostic Explanations
LRP	Layer-Wise Relevance Propagation
LTU	Linear Threshold Unit
ML	Machine Learning
MLP	Multi-Layer Perceptron
MMD	Maximum Mean Discrepancy
MSE	Mean Squared Error
NCI	National Cancer Institute
NIH	National Institute of Health
PDP	Partial Dependence Plot
ReLU	Rectified Linear Unit

RGB	Red, Green and Blue (additive colour model)
ROC	Receiver Operating Characteristic
RQ	Research Question
SGD	Stochastic Gradient Descent
SHAP	SHapley Additive exPlanations
SVM	Support Vector Machine
TCAV	Testing with Concept Activation Vectors
TN	True Negative
TP	True Positive
US	United States
VGG	Visual Geometry Group
XAI	Explainable Artificial Intelligence

Chapter 1

Introduction

1.1 Context and Motivation

The emergence of Deep Learning (DL) changed the Machine Learning (ML) paradigm in recent years, especially in Computer Vision (CV). Due to their significant performance improvement in tasks like classification, these models became dominant. They have since been applied to tackle different CV problems - e.g. detection, segmentation, depth estimation - and also to many other domains, such as speech recognition, text analysis and fraud detection.

Despite this overwhelming ubiquitousness, DL models and in particular Convolutional Neural Networks (CNNs), are for the most part considered black-box models, for which it is hard to understand the reasons for the labels that they generate. These models perform millions of non-linear operations, rendering it impossible for humans to follow this enormous amount of computations. Furthermore, contrarily to what happened with traditional ML methods, such as Decision Trees or Support Vector Machines (SVMs), the whole process is done end-to-end, from raw input to output, implicitly transforming the raw input into the most convenient feature space for the model's calculations, eliminating the need for feature engineering. This automatic feature extraction, although highly advantageous, adds an extra layer of complexity and opaqueness to these models. In fact, there is an important trade-off between predictive performance and opaqueness, as depicted in figure 1.1.

Therefore, understanding the decisions of DL models in general, and CNNs in particular, is of the utmost importance to allow the deployment of robust, transparent and trustworthy DL-based systems in real-world scenarios. This is particularly relevant in critical high-stakes areas, where the consequences of a wrong decision could greatly affect its users, or even lead to the loss of human lives. Examples of such highly regulated areas include medicine, criminal justice, finance and autonomous driving.

For these reasons, there is an increasing interest in finding solutions to tackle, or, at least, mitigate this limitation, and that can comply with recent legislation introduced in the European Union (EU) about the right to algorithmic explanation. In fact, these challenges are nowadays aggregated under what now is known as Explainable Artificial Intelligence (XAI). The term first

surfaced in 2004 in the work of Van *et al.* [1] as a description of their AI training system developed for the United States (US) Army, capable of producing explanations and answering questions about the platoon's ammunition status, for example.

However, this is still a relatively new research area, that lacks unified definitions of core concepts, such as interpretability and explainability, as well as a universally accepted taxonomy and quantitative evaluation frameworks.

Finally, on a slightly different note, data is the core ingredient in every DL pipeline. In fact, DL models are as good as the data with which they are trained. However, it is incredibly difficult to come by labelled datasets and it is highly expensive to collect and annotate one, either time- or labour-wise. Moreover, producing meaningful explanations is an user- and domain-dependent task. As such, it is imperative that new solutions are able to generate explanations without needing additionally labelled data.

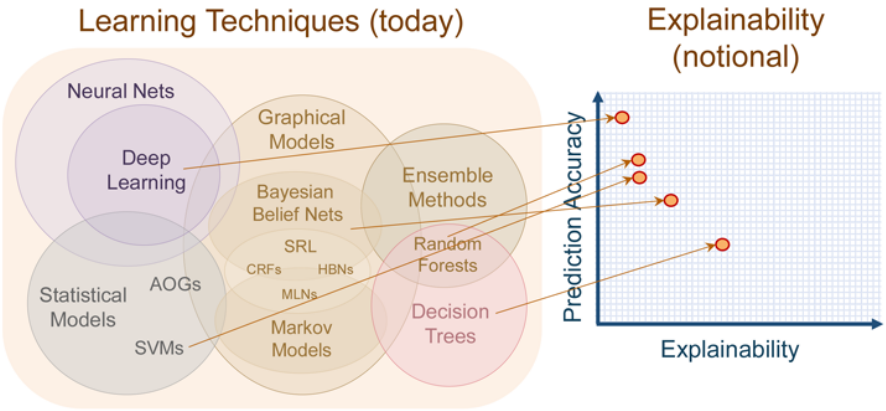


Figure 1.1: Learning techniques and the Accuracy-Explainability trade-off. Extracted from [2].

1.2 Problem Description and Objectives

Considering this growing importance in designing systems capable of producing explanations, on a macro level, this work's objective is to design and implement a solution to try and tackle the opaqueness problem inherent to CNN architectures.

Therefore, the objectives and technical challenges of this work can be formulated as a series of research questions (RQ) as follows:

- RQ1 - Would it be possible to generate visual explanations without supervision, eliminating the need for additional annotations of the data?
- RQ2 - Would it be possible to design an in-model explainability method capable of generating visual explanations as an implicit part of the training process?
- RQ3 - Would it be possible to fulfil the aforementioned aspects with an approach capable of being applied to different CNN architectures?

- RQ4 - Would it be possible to validate this architecture not only on synthetic data, but also on a real medical application?

More specifically, this work contemplates the development and study of an architecture composed by an explainer and a classifier, capable of generating not only class predictions, but also explanations for such decisions, in an unsupervised fashion. This work also involves the development of a synthetic data generation tool with automatic annotation, to speed up the evaluation process, by eliminating the need for expert knowledge in preliminary test scenarios.

1.3 Contributions and Publications

The main contributions of this dissertation are:

- A versatile and highly customisable synthetic data generation tool with automatic annotation.

- A comprehensive and critical survey on definitions, taxonomies and methods found in the reviewed literature, as well as the denomination of two previously unnamed taxonomies (Locative and Methodology criteria), according to their scope. Also, an in-depth study of the differences and overlap between the reviewed taxonomies, and a classification of the surveyed methods according to the different taxonomies.

- A novel in-model joint approach that produces visual explanations for the decisions of different CNN-based classifiers, along with a custom training procedure and loss functions, and its validation on a real world medical scenario.

- Custom loss functions (unsupervised and weakly supervised), that allow generating explanations without the need for supervision and no further annotations of the already existing data.

From the development of this work originated the following scientific paper:

- Isabel Rio-Torto, Kelwin Fernandes, and Luís F. Teixeira, "Towards a Joint Approach to Produce Decisions and Explanations Using CNNs", In 9th Iberian Conference on Pattern Recognition and Image Analysis, Springer International Publishing, pages 3-15, 2019.

This paper was selected for oral presentation as one of the best ranked papers in the ML category at the 9th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA), that took place in July 2019 at Universidad Autonoma de Madrid, Spain. It was the **winner of an honourable mention and was also selected for an extended version on Pattern Recognition Letters, the main archival journal of the International Association for Pattern Recognition (IAPR)**.

1.4 Document Structure

The remainder of this document is divided into 6 chapters, described as follows. Chapter 2 describes some fundamental concepts related to the ML paradigm, tasks and metrics (section 2.1), followed by a brief historical background on DL and some important aspects regarding this particular topic (section 2.2). The chapter ends with basic CNN concepts and the most widely used CNN architectures relevant for this work.

Chapter 3 reviews the literature, starting by detailing several definitions related to XAI, interpretability and explainability in section 3.1. Afterwards, aspects related to XAI systems are presented, such as its relevance nowadays (section 3.2), application scenarios and stakeholders (section 3.3), and feasibility and corresponding challenges (section 3.4). The literature review continues by surveying several proposed taxonomies in section 3.5 and evaluation strategies in section 3.6. Then, an extensive review of existing explainability methods covering model-agnostic, model-specific and intrinsically interpretable models is done in sections 3.7, 3.8 and 3.9, respectively. To end the chapter, a summary and discussion of the reviewed methods is presented in section 3.10.

Chapter 4 describes the proposed joint approach, starting with related work that inspired the proposed solution in section 4.1, as well as a brief overview (section 4.2), followed by a detailed characterisation of both classifier (4.3) and explainer (4.4). Finally, the proposed training process and loss functions are presented in section 4.5.

The next chapter, chapter 5, is dedicated to the experimental methodology, including the synthetic and real datasets used to test the proposed architecture (section 5.1). Afterwards, the chapter focuses on hyperparameter tuning strategies (section 5.2) and a brief study of the connection between explainer and classifier (section 5.3).

Chapter 6 presents the obtained results on both synthetic and real datasets, and regarding the two approaches addressed: unsupervised (section 6.1) and weakly supervised (section 6.2) approaches.

Finally, chapter 7 concludes this work and addresses future research directions.

Chapter 2

Fundamentals

This chapter focuses on fundamental concepts that are the underlying building blocks of this work and any ML system. First, the chapter starts with an overview of what ML is, as well as the three main learning paradigms: Supervised, Unsupervised and Reinforcement Learning. Then, the focus is shifted to DL in particular, starting with some historical background, and then proceeding to Deep Neural Networks (DNNs), their principal components and how they are trained. Finally, the chapter ends with CNNs, what distinguishes them from ”normal” DNNs, and some common CNN architectures relevant for the remainder of this work. For a more in-depth analysis and explanation, the reader can find detailed information in [3] and [4].

2.1 Machine Learning

2.1.1 Overview

ML is a subset of AI that studies models capable of learning without explicit instructions, relying only on data and its patterns. The term dates back to 1959 and is attributed to Arthur Samuel, which defined it as:

”[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.” - Arthur Samuel, 1959

The traditional programming paradigm, which is shown in figure 2.1 a), takes some data as input, applies an algorithm written by a programmer to that data and outputs the results of the calculations included in the computer program. Conversely, ML systems, figure 2.1 b), take as input the data and the desired output, the so-called ground-truth, and learn a model, that represents the learned information from that data.

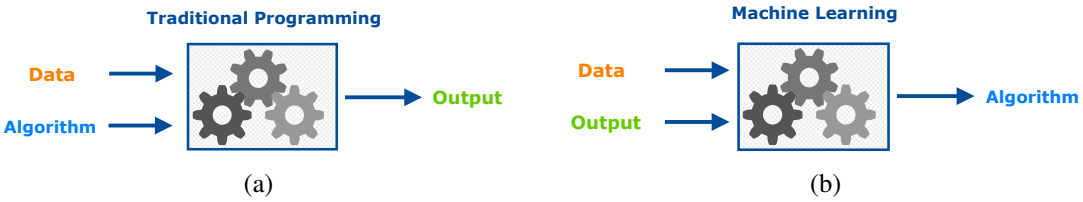


Figure 2.1: Traditional Programming versus ML.

There are different types of ML systems, but most of them follow one of these three paradigms: supervised, unsupervised, or reinforcement learning. Figure 2.2 represents these three paradigms and their main differences.

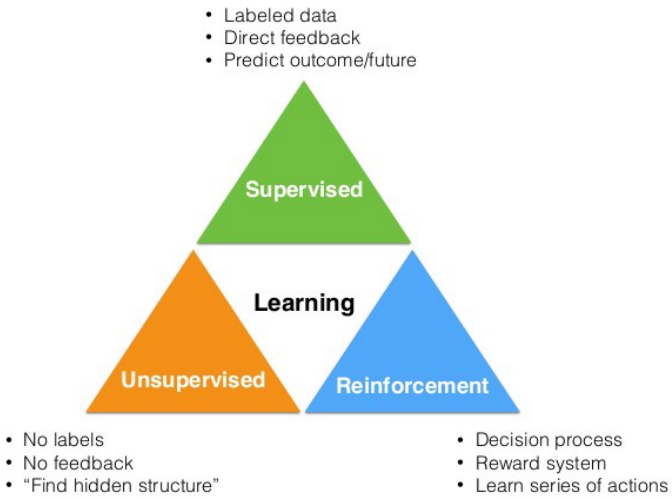


Figure 2.2: Learning Paradigms in ML. Extracted from [5].

On the one hand, in supervised learning, the ML model has human supervision, much like a teacher: the model is fed not only input data, but also labels for each data instance, i.e., the desired/known output for each instance. For example, a set of images of dogs and cats are given to the model, as well as the information for each image if it contains a dog or a cat. Having training data, X, and its respective labels, Y, the goal of the ML model is to learn the function that best approximates the unknown mapping function of X into Y, $Y = f(X)$. This approximated function is obtained iteratively: the algorithm predicts the output of an instance and is corrected if the prediction does not correspond to the label; learning goes on until a certain degree of performance has been achieved. Having such an approximation, one is able to predict Y from X. So, here the goal is to minimise the difference between the predicted and the actual output.

On the other hand, in unsupervised learning scenarios the data is unlabelled, so there is no "teacher" to supervise the learning process. In this approach, the goal is to uncover underlying patterns in the data, such as clusters of similar animals, for example.

The main problem with supervised learning resides in the need for enormous amounts of labelled data, which is costly to obtain. To mitigate this need, but still maintain some degree of supervision, between the supervised and unsupervised paradigms lies weakly supervised learning, which is an approach where the system is not fully supervised. Weak supervision can be done in different ways [6]:

- incomplete supervision (semi-supervised learning): only part of the training data is labelled
- inexact supervision: the training data only has coarse-grained labels or the existing labels are from another task, usually of higher level
- inaccurate supervision: the given labels are not always ground-truth

The third paradigm, reinforcement learning, is a bit different from the other two approaches. Here the learning system is called an ”agent”, that observes the environment and performs actions, getting rewards or penalties in return.

In this work, we will approach supervised, weakly supervised and unsupervised learning. When using weak supervision, we are only talking about inexact supervision, to which we will broadly refer as our weakly supervised approach.

2.1.2 Tasks

ML can be used to tackle different problems, which will be briefly mentioned in this section. In fact, ML is very useful when solving problems that are too complex to be solved by traditional programming techniques, either because they require a lot of hand-tuning or have no known algorithm [3].

A typical supervised learning task is classification, in which one is simply interested in predicting the class of each data instance from a known discrete set of classes, for example, predicting types of dog breeds. Another common problem is regression. Here the output is continuous, so the goal is to predict a numerical value, for example estimate the price of a house.

Unsupervised techniques are most commonly used to perform clustering, i.e., find natural groups in the data and assign a label to each group, or dimensionality reduction, reducing the data feature space.

These techniques are usually independent of the type of input data, that can be tabular (numerical, categorical, etc), sequential or visual. In fact, ML has been applied successfully in many CV tasks. CV is an interdisciplinary field that aims at extracting meaningful information from visual data, such as images and videos. Typical CV tasks include image classification, object detection, semantic and instance segmentation and image captioning. Image classification only addresses the problem of identifying which objects appear in an image. Object detection localises those objects inside the image, usually through bounding boxes. Semantic segmentation consists in splitting an image into segments and labelling each of these segments according to the object they contain. Instance segmentation extends semantic segmentation by distinguishing objects within the same class. Finally, image captioning focuses on generating descriptions of the contents of an image.

These tasks, except image captioning, are depicted in figure 2.3, in ascending order of difficulty and complexity, from simple classification and object detection to segmentation.

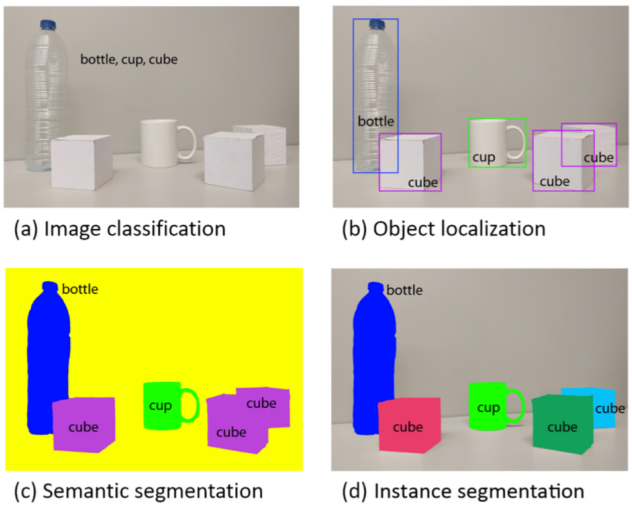


Figure 2.3: CV tasks ranging from image classification to instance segmentation, in ascending order of difficulty. Extracted from [7].

This work focuses on image classification and, consequently, the next sections will deal only with classification related aspects.

2.1.3 Metrics

Measuring the performance of a classifier can be tricky, conversely to measuring the performance of a regressor, which usually just entails calculating some variant of the Mean Squared Error (MSE), the average squared difference between the predicted and the actual values.

Classification tasks are usually evaluated in terms of their accuracy, the ratio between the number of correctly predicted instances and the total number of instances:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predicted instances}} \tag{2.1}$$

Accuracy works well if the data is balanced, i.e., if each class is equally frequent. This rarely is the case, especially in medical contexts, where there are usually many examples of healthy patients and a few examples of patients with a certain disease.

Another way to measure classification performance is the Confusion Matrix, which is a matrix that incorporates information about the number of correctly and incorrectly predicted instances, as depicted in figure 2.4. This analysis can be used to evaluate binary classifiers, in which only two classes are predicted, the positive and negative classes, and can also be extended for multiclass problems, with one column/row per class. In a dog classification system the positive class would be ”dog” and the negative class would be ”not dog”. This matrix is composed by the number

of True Positives (TP), instances that are correctly predicted as positive, True Negatives (TN), instances that are correctly predicted as negative, False Positives (FP), instances predicted as positive although they were actually negative and False Negatives (FN), instances incorrectly predicted as negative.

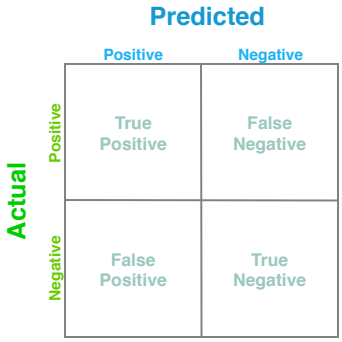


Figure 2.4: Confusion Matrix.

From this matrix we can rewrite accuracy and define other metrics, such as precision, recall and the F1-score, presented in the following equations.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{2.2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.4}$$

$$\text{F1} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{2.5}$$

Finally, for binary classification it is also common to plot the Receiver Operating Characteristic (ROC) curve and calculate the area under it (AUC). The ROC curve is a plot of the True Positive Rate, or recall, against the False Positive Rate, which is the ratio of negative instances that are correctly classified. The ROC AUC can be used to compare classifiers and the closer it gets to 1 the better the classifier. An AUC of 0.5 corresponds to a purely random classifier.

2.2 Deep Learning

DL is one type of ML methods based on Artificial Neural Networks (ANNs). These distinguish themselves from traditional ML methods, such as Linear Models, Decision Trees or SVMs, essentially by performing automatic feature extraction. As shown in figure 2.5, traditional ML methods take as input a set of features, which were previously extracted from the raw data and transformed

to fit the ML method input feature space. For example, in a traditional method designed to help diagnose cervical cancer, specialists would have to extract from cervigrams features like the number of visible lesions and their sizes, and only those features would be fed to the method. However, this process is complex, costly, time consuming and requires expert knowledge. DL techniques replace this engineering step by learning the best features, a process that is called feature learning or representation learning. Therefore, with DL, the whole process is done end-to-end, from raw input to output, and the method transforms that raw input into the feature space most convenient for its calculations. In the cervical cancer example, only the cervigrams would be needed.

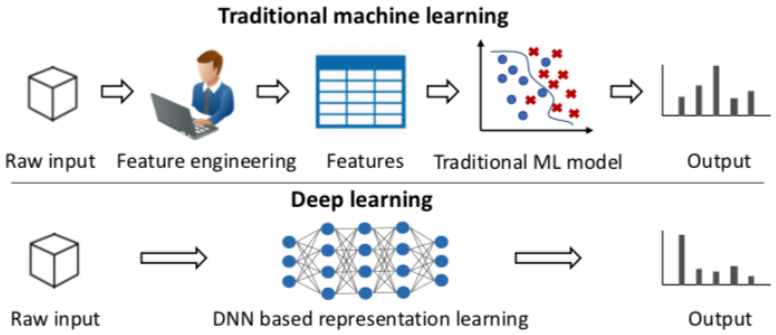


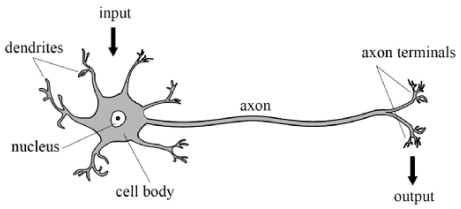
Figure 2.5: DL versus the traditional ML paradigm. DL-based systems are able to perform feature learning and extraction, eliminating the need for feature engineering. Extracted from [8].

2.2.1 Historical Background

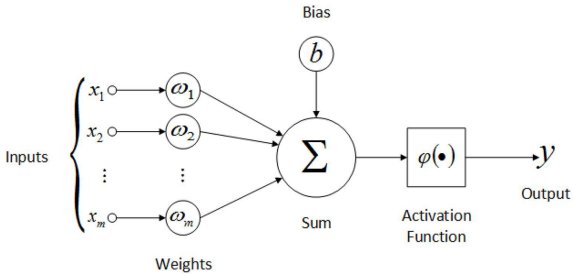
ANNs were first introduced in 1943 by neurophysiologist Warren McCulloch and mathematician Walter Pitts [9]. The authors proposed a simple model of the biological neuron: it had one or more binary inputs and one binary output that was activated when more than a certain number of its inputs were activated [3].

In 1957, Frank Rosenblatt introduced the *Perceptron*, one of the simplest ANNs with just a single layer, based on a slightly different artificial neuron, the Linear Threshold Unit (LTU): its inputs and outputs are now numbers and each input connection has an associated weight. The LTU computes the weighted sum of the inputs and then applies an activation function, originally a step function, which produces the final result. If the weighted sum exceeds the threshold, the neuron is said to be activated and outputs one class, else it outputs the other class [3]. A diagram of an LTU can be found in figure 2.6 b).

These artificial neuron models were inspired by biological neurons, as can also be seen in figure 2.6. Briefly, biological neurons are composed by a cell body, branching extensions called dendrites, and one long extension called the axon. Analogously, the weights of the artificial neuron are the dendrites, the weighted sum is the cell body and the activation function is the axon.



(a) Biological Neuron. Extracted from [10].



(b) Artificial Neuron Model. Extracted from [11].

Figure 2.6: Biological versus Artificial Neurons.

The early successes of ANNs in the 1960s seemed promising, but soon research in this area stagnated, due to the lack of an adequate training algorithm, which was just proposed in 1985 by Rumelhart *et al.* in their groundbreaking work [12]. This algorithm, backpropagation[1], allowed training Multi-Layer Perceptrons (MLPs), a network of stacked Perceptrons, which was able to solve the famous XOR problem [13], previously unsolved by single Perceptrons. Still, ANNs would not gain traction until recent decades, mainly due to powerful alternatives like SVMs, which offered better results, accompanied by stronger theoretical foundations [3]. Recent advances in computing, with the development of the Graphics Processing Unit (GPU), and the growing quantity of available large scale datasets, has made DL possible, which led to the achievement of outstanding predictive performances, even having outperformed humans. Figure 2.7 shows a timeline of DL from 1943 until recent years.

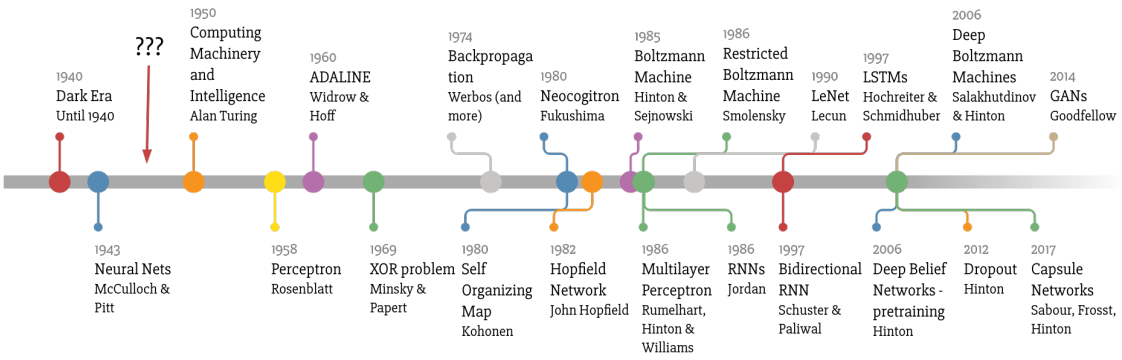


Figure 2.7: DL Timeline. Extracted from [14].

[1]Backpropagation was invented several times, starting with P. Werbos in 1974.

2.2.2 Deep Neural Networks

As mentioned in the previous section, MLPs result from stacking Perceptrons together. They are composed by an input layer and one or more LTU layers, being the last one the output layer. When such an ANN presents two or more hidden layers it is called a DNN [3]. An example of such structure can be found in figure 2.8, where a neural network with two hidden layers is presented. These layers, where each neuron takes as input the outputs from every neuron from the previous layer, are called *dense* or *fully connected layers*.

DNNs are trained using the *backpropagation* algorithm [12], as already mentioned. Briefly, the process starts by feeding each training instance to the network, layer by layer until it reaches the output layer - feed forward pass (similar to what is done when the learned network is used to predict the outputs of unseen instances during training, also called inference). Then, the difference between the predicted and the actual output is calculated. Afterwards, it goes through each layer in reverse to measure the error contribution of each connection - reverse/backward pass[2]. Finally, the connection weights are tweaked to reduce the error according to an hyperparameter called the learning rate or step size - Gradient Descent (GD) step [3].

A few of the basic concepts regarding DNNs and their training are outlined below, mainly weight initialisation, activation and loss functions, optimisers and regularisers, ending with batch normalisation.

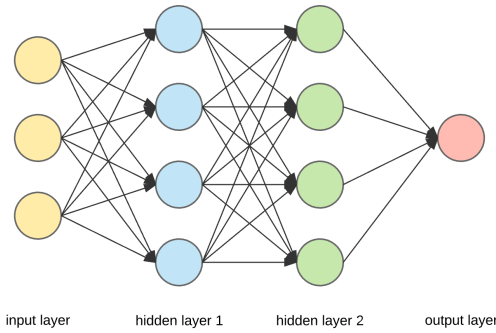


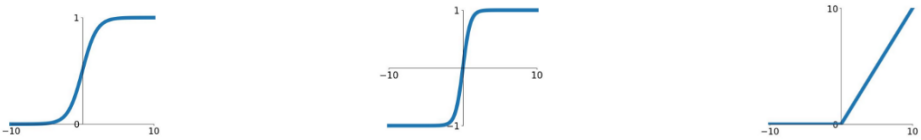
Figure 2.8: Typical neural network structure with an input layer, two hidden layers and an output layer. Extracted from [15].

Weight initialisation The backpropagation algorithm works by propagating the error gradient from output to input, computing the gradient of the cost function with respect to each parameter in the network and using it to update each parameter. These gradients can get gradually smaller, leaving the layers closer to the input practically unchanged and preventing GD convergence to a good solution - vanishing gradients problem. Conversely, gradients can also get bigger and bigger, leading to a divergence of the algorithm - exploding gradients problem. In their work [16], Glorot and Bengio proposed what is now called the Xavier or Glorot initialisation. To achieve proper signal flow, the authors argue that the variance of the outputs of the layer has to be equal to that of the layer's inputs and the gradients also need to have equal variance before and after flowing

[2]The name backpropagation comes from this processing step.

through one layer while in the reverse pass of backpropagation [3]. So, this method consists in randomly initialising the weights of each neuron according to a distribution that takes into account the fan-in and fan-out of the layer (number of inputs and outputs of the layer, respectively). This distribution can be a normal distribution, where this dependence on the fan-in and fan-out is reflected in the distribution's standard deviation, or a uniform distribution, where this dependence is reflected in the lower and upper limits of the distribution.

Activation Function Defines the output of a neuron given its inputs, deciding if the neuron should fire or not. Activation functions define a threshold, above which the neuron is activated. The Perceptron originally used the step function as activation, which was problematic for GD and, consequently, backpropagation, because the step function only has flat segments, which have no gradient. Nowadays, activation functions with nonzero derivatives are used, such as the Sigmoid function, $S(z) = \frac{1}{1+e^{-z}}$, the hyperbolic tangent function, $tanh(z) = 2\sigma(2z) - 1$, the Rectified Linear Unit (ReLU) function, $ReLU(z) = max(0, z)$, or one of its variants, like LeakyReLU, which avoids the problem of dying ReLUs [3]. Examples of these functions can be found in figure 2.9. These activation functions are typically used in intermediate layers of the network. The output layer usually employs the Softmax activation function, in multiclass problems, which normalises the outputs for each class between 0 and 1 and divides them by their sum, resulting in the probability of each instance belonging to each class. The class with the highest probability can then be chosen as the predicted class. It is worth noting that the choice of the most adequate function is application specific, but usually non-saturating activation functions are preferred, because these are less prone to the vanishing/exploding gradients problem.



(a) Sigmoid (b) Hyperbolic tangent (c) ReLU

Figure 2.9: Typical activation functions used in DNNs[3].

Loss Function An objective function is, in the context of optimisation, the function used to evaluate a candidate solution. In ML in particular, it is a measure of how a model performs in predicting the expected outputs. Usually, the goal is to minimise this function, thus it is referred to as a cost or loss function. Commonly used loss functions for classification include the cross entropy or log loss, presented in the following equation:

$$L_{class} = -\sum_{i=1}^{N} y_i^{\top} \times log(\hat{y}_i) \tag{2.6}$$

where N is the number of instances in the training dataset, y_i contains the target class labels for instance i and $\hat{y}_i$ are the classifier's predictions for instance i.

[3] Adapted from https://www.quora.com/Why-is-ReLU-the-most-common-activation-function-used-in-neural-networks

Optimisers When training a DNN it is also important to choose the correct optimisation strategy, i.e., the iterative optimisation process to reduce the loss function. Vanilla GD is rarely used when training DNNs, since it involves using the whole training set at once, which is usually not possible as it cannot fit into the GPU's available memory. Multiple alternatives do exist, being Stochastic GD (SGD) the preferred one. These optimisers have a static learning rate, but other optimisers have adaptive learning rates. Generally, adaptive optimisers reach the minimum faster, because the resulting updates are pointed more directly towards the global minimum, and require less tuning of the learning rate hyperparameter. Examples of such optimisers include Adam [17], Adadelta [18] and, most recently, Adabound [19]. Much like the choice of activation function, choosing the optimiser and its learning rate is dependent on the network architecture and the data.

Regularisation A DNN has thousands of parameters and with it the possibility of fitting huge complex datasets. However, this can lead to what is known as overfitting. Overfitting occurs when the network is too well adapted to the training data and, thus, generalises poorly on unseen data. Avoiding overfitting can be done through regularisation, i.e., by constraining the model and limiting its many degrees of freedom [3]. One way of doing this is by constraining the network's weights, for example by adding the *penalised* $l1$ *or* $l2$ *norms* of the weights to the loss function. Another regularisation technique is *Early Stopping* and, as the name implies, it simply consists in stopping training when the validation loss stops decreasing. This technique usually works well in practice when combined with other regularisation methods, such as *Dropout*, for example [3]. Dropout was first proposed by Hinton *et al.* [20]: at every training step each neuron has a probability, the dropout rate, of being temporarily ignored during that training step. Using Dropout, neurons become less sensitive to small changes in the input, because they cannot co-adapt with their neighbours, nor rely excessively on some of their input neurons, which leads to a more robust network that generalises better. Dropout can also be thought of as creating an averaging ensemble of smaller neural networks, as a unique network is generated at every training step [3]. Other regularisation technique is *Data Augmentation*, which simply involves generating new training instances from existing ones, by performing transformations such as rotations and translations, artificially increasing the size of the training set and, thus, reducing the possibility of overfitting. Finally, *Batch Normalisation*, proposed by Ioffe *et al.* [21], besides being a regularisation technique, also improves the speed, stability and performance of DNNs. This technique addresses the Internal Covariate Shift problem, the fact that the distribution of each layer's inputs changes during training, and the vanishing/exploding gradients problem. Briefly, it consists in adding an operation just before the activation function of each layer, zero-centering and normalising the inputs, and then scaling and shifting the result, i.e., it allows the model to learn the optimal scale and mean of the inputs for each layer. To do so, the algorithm computes the mean and standard deviation of the inputs over each batch of the training data. During inference, it uses the mean and standard deviation of the whole training set. This technique reduces the vanishing gradients problem, lessens the dependence on weight initialisation, allows using higher learning rates and, consequently, speeds up the learning process, and acts as a regulariser, reducing the need for other

regularisation techniques. However, due to the additional calculations it introduces, the network makes slower predictions [3].

2.2.3 Convolutional Neural Networks

CNNs are a type of DNNs that take as input image data. These networks are mainly used for visual applications, such as image analysis and classification, image and video recognition, object detection and localisation, or image segmentation.

These networks emerged from the study of the brain's visual cortex by Nobel laureates David H. Hubel and Torsten Wiesel [22, 23, 24], having inspired the neocognitron [25] in 1980, which evolved into current CNNs. The authors showed that neurons in the visual cortex have small receptive fields that react to visual stimuli located in a sub region of the visual field. Furthermore, their work showed that some neurons react only to horizontal lines, while others react to lines with different orientations, and that neurons with larger receptive fields reacted to more complex patterns that constituted a combination of lower-level patterns [3].

Figure 2.10 depicts a typical CNN architecture, which is composed of some convolutional and pooling layers, which are described in section 2.2.3.1, followed by some fully connected layers, ending with a Softmax layer.

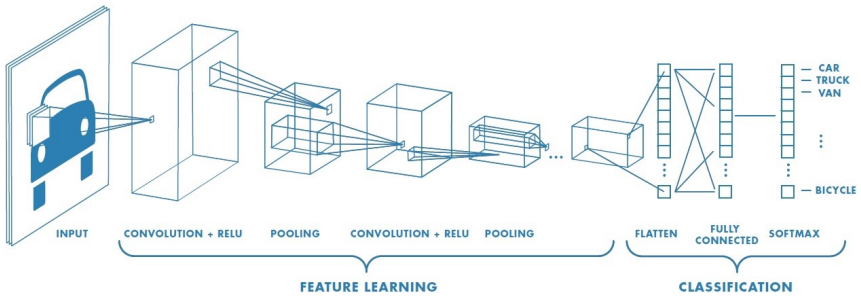


Figure 2.10: Typical CNN architecture. Extracted from [26].

2.2.3.1 Layers

CNNs use the same layers as regular DNNs, but introduce two new types of layers: convolutional and pooling layers.

Convolutional Layer The convolutional layer is the core building block of CNNs. Instead of being connected to every single pixel of the input image, which would happen in regular DNNs, leading to an exponential increase in the number of parameters, neurons on the first convolutional layer are solely connected to pixels in their respective fields, similarly to what happens in the neurons of the brain's visual cortex. This is repeated throughout the different convolutional layers, making this a hierarchical structure that focuses on learning low-level features in the first layers and then joining them into higher-level features as the layers stack-up.

As depicted in figure 2.10, the output of each layer is a set of stacked *feature maps* of the same size. A feature map results from a layer full of neurons using the same filter[4], by means of the convolution operation. This operation consists in computing the dot product between each feature map of the previous layer and the respective filter. During training, the CNN learns the most suited filters for the task and combines them into more complex patterns [3]. In other words, a convolutional layer simultaneously applies multiple filters to its inputs, thus detecting multiple features. Convolutional layers, as described so far, preserve the input's height and width, changing only its depth. However, height and width can be changed by using what is called a stride, greater than 1. The stride defines the distance between consecutive receptive fields. For example, a stride of 2 indicates that the filter moves 2 pixels in the input for every pixel in the output, so it defines the ratio between input and output, thus originating a feature map with half the width and the height of the original input image. Conversely, the reverse operation is also possible, in an operation called transpose convolution or fractionally strided convolution, as it equals a convolution with a stride lower than 1.

Pooling Layer The pooling layer sub samples one or more dimensions of its input image, so as to reduce the computational and memory requirements needed. Similarly to convolutional layers, the pooling layers are connected to a limited number of neurons in the previous layer, the ones in their receptive field. However, a pooling layer presents no weights and simply aggregates the inputs using a function like *max*, in which only the maximum input value is propagated to the next layer, or *average*, in which the mean of the input values is propagated to the next layer, as depicted in figure 2.11. The pooling operation is usually applied to the image's width and height, but it can also be applied to its depth, reducing the number of channels of the input image. While pooling performs downsampling, upsampling is also possible through unpooling. Finally, one can also employ global pooling, which is an operation that reduces a matrix from 3D to 1D, by choosing only one value (either the maximum or the average) from each feature map, therefore reducing an input of $w \times h \times d$ to $1 \times 1 \times d$. A global pooling layer is often used after the convolution-pooling stages and before fully connected layers, to prepare the features for these last layers, without the need for flattening.

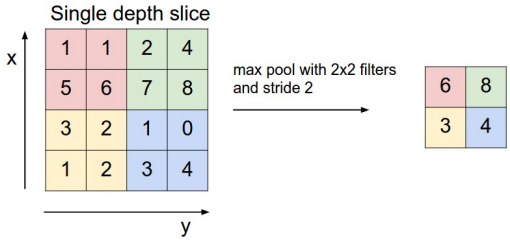


Figure 2.11: Max pooling example. In this case, a stride of 2 is used, which means that the input is 2 times the size of the output. A filter of size 2×2 is used, so every 4 pixels in the input are mapped to 1 pixel in the output. Extracted from [27].

[4] A filter or convolutional kernel is a representation of a neuron's weights as an image the size of its respective receptive field [3]. In other words, it is a $k \times k$ matrix of learnable weights.

2.2.3.2 Architectures

As previously mentioned, a typical CNN architecture, depicted in figure 2.10, comprises convolutional layers followed by ReLU and pooling layers. Usually, as the image progresses through the network it gets smaller and smaller, but also deeper and deeper (with more feature maps), depending on the filters used [3]. After these layers, a general feedforward network is added, composed of some fully connected layers and their respective activation layers, typically ending with a softmax layer to output estimated class probabilities.

Typical CNN architectures, briefly described below, include the famous LeNet-5, AlexNet, VGG16, GoogLeNet and ResNet. Also, a CNN architecture called U-net is reviewed, because it is relevant for the remainder of this work.

LeNet-5 LeNet-5 [28] is one of the best known CNN architectures. Created by Yann LeCun in 1998 and applied to handwritten digit recognition on the widely used MNIST dataset, it included two convolution-average pooling stages, followed by another convolutional layer and two fully connected layers. It used the hyperbolic tangent function as activation and a slightly different output layer that outputted the Euclidean distance between its input and weight vectors [3]. Its inputs had size 32×32.

AlexNet AlexNet [29] was introduced in 2012 by Krizhevsky *et al.* and achieved outstanding performances on the ImageNet Large Scale Visual Recognition Challenge[5] (ILSVRC), a classification challenge of over 14 million images belonging to 1000 classes. This CNN is similar to LeNet-5, but larger and deeper: with two convolution-pooling stages, three convolutional layers and three dense layers. Its inputs are seven times larger than those of LeNet-5, with size 224×224, and it uses large kernels in the first layers, namely 11×11 and 5×5. Contrarily to LeNet-5, it uses ReLU activation functions for the hidden layers and softmax for the output layer.

VGG-16 Proposed by Simonyan *et al.* [30] (members of the Visual Geometry Group hence the name VGG) in 2014, it was one of the best performing models in ILSVRC 2014, along with GoogleNet. It improved on AlexNet by using smaller kernels, of size 3×3, which stacked into 3 layers have the same receptive field as 7×7 kernels, but incorporating 3 instead of only 1 non-linear rectification layer, making the decision function more discriminative. Also, a single 7×7 kernel has 81% more parameters than 3 stacked 3×3 kernels [30]. This architecture, presented in figure 2.12, also takes as input 224×224-sized images, that go through 2 convolution-pooling stages, followed by 3 convolution-convolution-pooling stages and 3 fully connected layers, making a total of 16 weight layers. Other VGG architectures exist, such as VGG-19, with 19 weight layers.

[5] http://image-net.org/

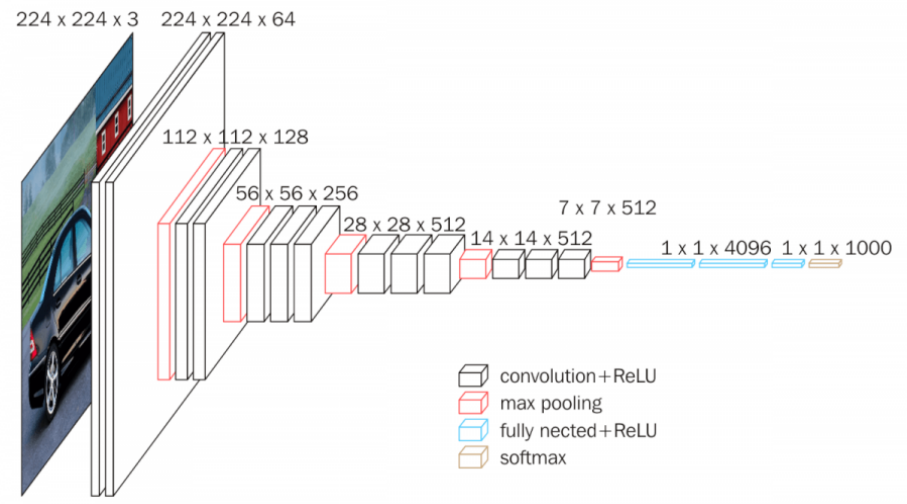


Figure 2.12: VGG-16 architecture. Extracted from [31].

GoogLeNet GoogLeNet [32] was developed by Szegedy *et al.* from Google Research and won the ILSVRC 2014 challenge, by building a much deeper network than previous CNNs. This architecture is composed by smaller networks called inception modules, depicted in figure 2.13, that allow GoogLeNet to use parameters more efficiently, leading to fewer parameters than its predecessors, such as AlexNet. The inception module uses 1×1 convolutions to perform dimensionality reduction and has different kernel sizes in the second set of layers, 3×3 and 5×5, to be able to capture patterns at different scales. The whole architecture includes 9 inception modules with 3 layers each.

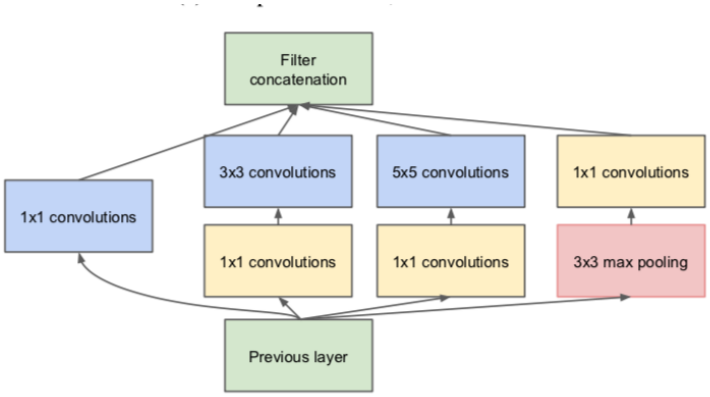


Figure 2.13: Inception module with dimensionality reduction. Extracted from [32].

ResNet ResNet [33] won the ILSVRC 2015 challenge with an extremely deep network of 152 layers and was developed by Kaiming He *et al.*. Training such a deep network was possible through the introduction of skip connections: connections that link one layer with another one a bit higher up in the stack. When training a DNN, the goal is to model a function $h(x)$; by adding the input x to the output of the network, it will be forced to model $f(x) = h(x) - x$, in what is called

residual learning [3]. Thanks to this technique, the signal flows easily across the whole network, which considerably speeds up training. A diagram of the developed residual block can be found in figure 2.14, where the skip connection is also depicted. The whole architecture is composed by a very deep stack of residual units: 2 convolutional layers with Batch Normalisation and ReLU activation, and 3×3 kernels. Typically, ResNet is used with 50 residual layers, also known as ResNet-50.

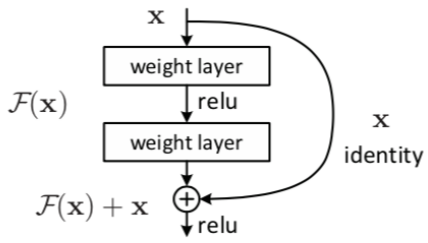


Figure 2.14: Residual learning block with a skip connection. Extracted from [33].

Figure 2.15 shows the evolution of the top 5% error on the ILSVRC competition throughout the years, from 2010 to 2015, with a significant increase in the number of layers used by 2015.

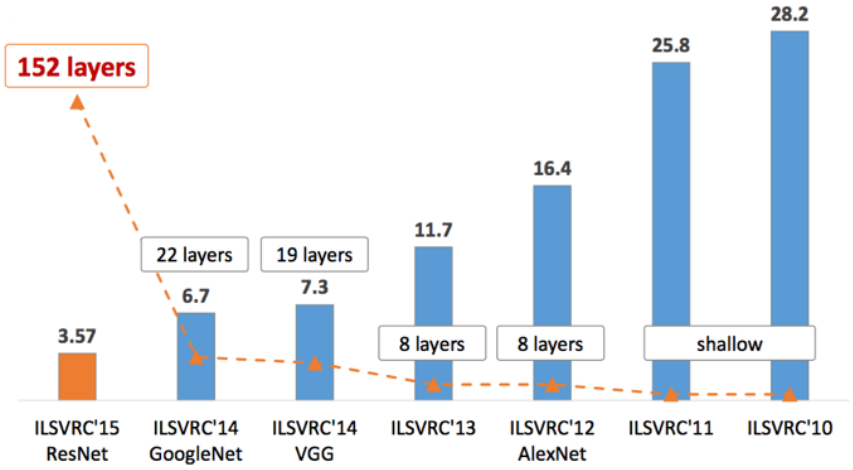


Figure 2.15: ILSVRC top 5% error on ImageNet. Extracted from [34].

U-Net The U-Net [35] is a different CNN architecture, because it outputs an image with the same size as the input, rather than class scores. It was developed by Ronneberger *et al.* to tackle biomedical imaging segmentation with small datasets. It is an encoder-decoder based architecture: the encoder maps the input space to a different/latent space, while the decoder learns the complementary function that maps from the latent space to the target space. In this case, the target space is the same as the input space and the network is trained end-to-end, pixels-to-pixels. The main idea is to only use convolutional layers and to perform transpose convolution operations to recover the spatial dimensions of the original input, altered by the downsampling introduced by the convolutional layers. The encoder is a typical feature extractor based on the VGG-16 network and

the decoder is a shape generator that outputs images from the extracted features. This upsampling part uses cropped feature maps from the downsampling part to avoid losing global information, as depicted in figure 2.16.

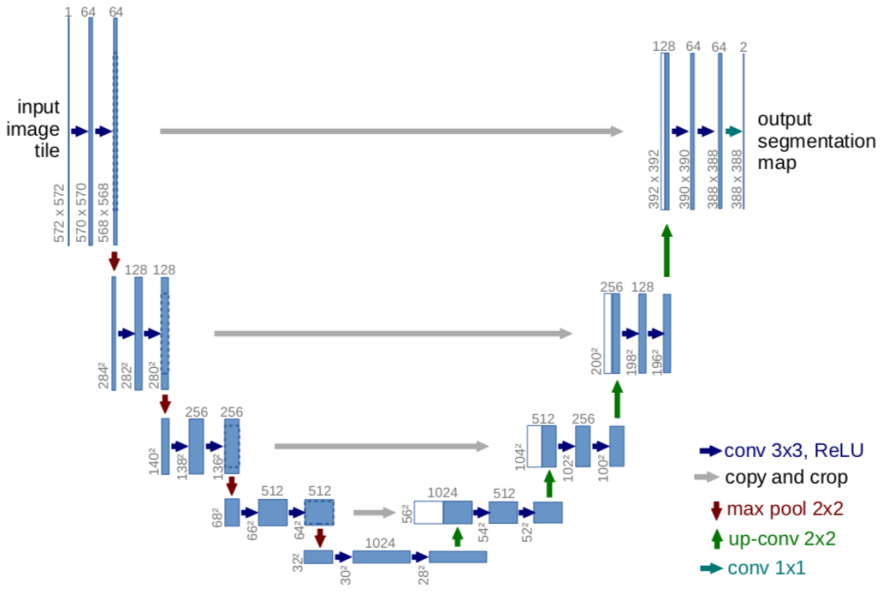


Figure 2.16: U-Net architecture, composed by an encoding and a decoding path. Extracted from [35].

Chapter 3

Explainable Machine Learning: Literature Review

This chapter is dedicated to a comprehensive and critical review of the existing literature on interpretability and explainability in ML, more specifically in DNNs. First, fundamental concepts such as XAI, interpretability and explainability are introduced. Then, it is explored why these properties are needed and what makes them relevant for ML systems, followed by the goals and applications of explainability, as well as the challenges that the research community has to face in order to create explainable systems.

After this contextualisation and motivation, different explainability taxonomies are described, along with various evaluation strategies found in the reviewed literature.

Then, the focus is shifted towards state-of-the-art explainability methods, starting with model-agnostic methods, i.e., methods that can be applied to any kind of ML model; these are further divided into global and local methods. The remaining methods fall under the category of model-specific methods and, since this work focuses on CNNs, this section revolves around the explainability of DNNs.

To close the chapter other explainability methods are briefly mentioned, such as intrinsically interpretable models, succeeded by a summary of the reviewed methods and, finally, a discussion of the reviewed literature.

3.1 Concepts and Definitions

3.1.1 Explainable Artificial Intelligence (XAI)

Artificial Intelligence (AI) is nowadays widely adopted in our daily lives, its presence ranging from movie recommendation systems to tailored advertising. As such, it is increasingly important to be able to understand the reasons behind the decisions of these systems (see section 3.2). This is where Explainable AI comes into play, proposing a paradigm shift towards transparent AI [36].

Formally, there is no standard definition of XAI, but it generally refers to research and initiatives towards AI transparency and trust [36]. According to the Defense Advanced Research

Projects Agency (DARPA), XAI is an emerging research field with the goal of developing explainable models without sacrificing high prediction accuracies and enabling humans to understand and trust AI partners [2]. The term first surfaced in 2004 in a paper by Van *et al.* [1] as a description of their AI training system developed for the U.S. Army capable of producing explanations and answering questions like "What is the platoon's ammo status?".

As can be seen if figure 3.1, the XAI paradigm differs from the present AI/ML paradigm by replacing classic Learning Models by an Explainable Model and an Explanation Interface capable of answering the user's questions about the decisions made by the system, leading to a more transparent, robust and trustworthy system.

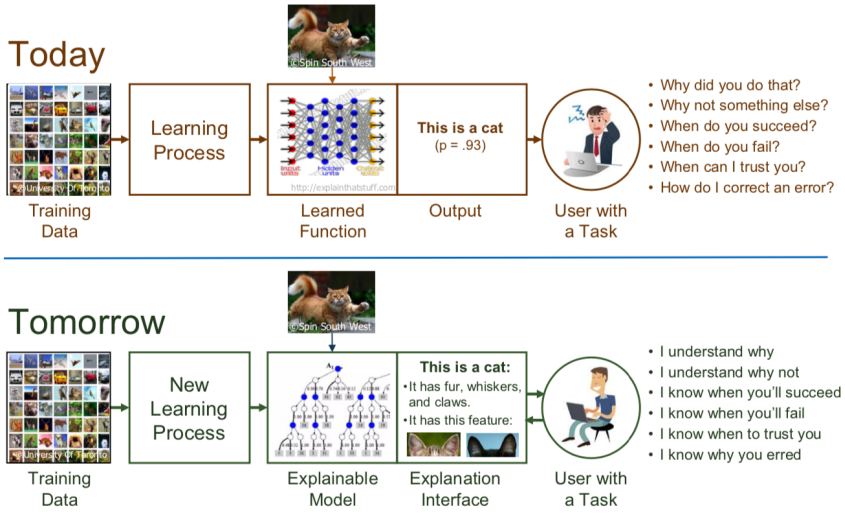


Figure 3.1: Comparison between today's ML paradigm with the future paradigm introduced by XAI [2].

3.1.2 Interpretability and Explainability

Despite the growing interest in XAI as a research field and as the future of AI/ML, the field still suffers from ambiguous terminology. In fact, work in this field usually employs the terms interpretability and explainability, and often does so interchangeably [37]. However, the word interpretability is preferred to explainability in the ML community, as observed in figure 3.2. Other terms such as understandability [38] or intelligibility [39] appear, but less frequently.

Figure 3.2: Google trends for interpretability and explainability in scientific and non-scientific contexts [36].

As noted by Lipton [40], there is no agreed upon definition of interpretability, but papers still use the term in a *"quasi-mathematical way"*, which increases the need for a clear taxonomy. According to the Merriam-Webster English Online Dictionary[1], to interpret means *"to explain or tell the meaning of, present in understandable terms"*. Based on this definition, Doshi-Velez *et al.* [41] define interpretability as the ability to present in understandable terms to a human being. The same authors also state that: *"Interpretability is NOT about understanding all bits and bytes of the model for all data points (we cannot). It's about knowing enough for your downstream tasks."* [42].

However, other definitions arise in literature, suggesting that the concept of interpretability consists of a myriad of distinct ideas, as suggested by Lipton [40]. For example, Montavon *et al.* [43] distinguish the terms *interpretation* and *explanation*:

Definition 1. An interpretation is the mapping of an abstract concept (e.g. a predicted class) into a domain that the human can make sense of.

Definition 2. An explanation is the collection of features of the interpretable domain, that have contributed for a given example to produce a decision (e.g. classification or regression).

Therefore, an example of an interpretable domain would be an image or text, whereas an explanation would be a heatmap which highlights the input image's pixels that contributed the most to a certain classification output, or highlighted text in the case of natural language processing [44].

Gilpin *et al.* [45] also makes a distinction between interpretability and explainability. Similarly to what Doshi-Velez *et al.* [41] state and was previously mentioned, Gilpin *et al.* define interpretability as a description of the internals of a system in a human-understandable way. Notwithstanding, explainability encompasses the ability to summarise the reasons for the model's behaviour, to gain user trust or to produce insights about the causes for the model's decisions. As a result, the authors consider interpretability a crucial but not single aspect to achieve explainability: completeness is also needed. In their view, a complete system should be able to describe its operation in an accurate way and, therefore, a system is more complete when it allows for its behaviour to be predicted in more situations. Thus, an explanation can be evaluated according to its interpretability and completeness; explainable models are interpretable by default, but the reverse is not guaranteed. Moreover, this differentiation introduces an *interpretability-completeness trade-off*: the more complete (accurate) the explanation, the less interpretable it is to humans; conversely, highly interpretable descriptions usually do not provide predictive power. As an example one can consider that a perfectly complete explanation of a DNN would be a listing of all the mathematical operations and parameters of the network, which is not understandable to a human and does not allow one to reproduce the mapping from inputs to outputs [45].

[1] Interpret [Def. 1]. (verb). In *Merriam Webster Online*, Retrieved February 03, 2019, from https://www.merriam-webster.com/dictionary/interpretability

In this work, we adopt the definitions introduced by Gilpin *et al.* [45], because we consider the interpretability-completeness trade-off an important balance to take into account, especially when working with CNNs and image data, where only certain image regions contribute to a model's decision. Also, because XAI systems are usually evaluated using human evaluations that prefer simpler descriptions, one must be careful not to produce persuasive systems that simplify descriptions in order to increase trust [45, 46]. As argued by Gilpin *et al.*, to avoid this problem one must always consider the interpretability-completeness trade-off. Therefore, the term explainability is used to describe methods that aim to summarise, be it by visual or textual means, a model's decisions, in a way that allows humans to understand such decisions.

3.1.3 Explanations

Having presented different definitions for interpretability and explainability found in the reviewed literature, we proceed to describe our findings on explanations and their desired properties.

Many attempts to define what an explanation is have been made across different knowledge areas, such as philosophy and mathematics, implying that a single definition is unobtainable, since it would be context- and application-dependent [47]. According to Miller [48], *"...an explanation is the answer to a why question"*. In fact, theories of explanation can be divided into two categories according to the answer to this *"why question"* [49]:

- **Non-pragmatic theory of explanation**: there is only one correct answer to the why question and the explanation is that answer.

- **Pragmatic theory of explanation**: the why question can be answered differently according to the target audience and, therefore, an explanation constitutes a good answer to be given by an explainer when facing a particular audience.

Simply put, pragmatic theory of explanation takes into account the target audience, its knowledge base and prior beliefs, while non-pragmatic theory of explanation does not. Considering non-pragmatic theory, it argues that only one true explanation exists, making the correctness of an answer to the *"why question"* independent from its understandability by the target audience. Naturally, when it comes to XAI, pragmatic theories are more aligned with is principles, goals and requirements [49]. Miller's work [48] also backs up this reasoning, when stating that the explanation process consists of two sub-processes, the cognitive and the social processes:

- **Cognitive process**: the abductive inference process (*"filling the gaps"*), in which causes for the event being explained are identified and a subset of these causes are selected as the explanation.

- **Social process**: the knowledge transfer process between explainer (human or machine) and explainee. Its goal is to provide the explainee with enough information to understand the causes of the event.

Doshi-Velez *et al.* [50] define explanation as being *"a human-interpretable description of the process by which a decision-maker took a particular set of inputs and reached a particular conclusion"*. This formal definition is accompanied by the requirement that *"an explanation must also have the correct type of content in order for it to be useful"*. The authors also state that an explanation should allow for the determination of the extent to which some input influenced an output, or put somewhat differently, an explanation should be able to answer one of the following questions:

- What were the main factors in a decision?
- Would changing a certain factor have changed the decision?
- Why did two similar-looking cases get different decisions, or vice versa?

All these considerations highlight the need to take the target audience and the problem domain into account when producing explanations, emphasising the subjectivity inherent to explanations.

3.1.3.1 Human-Friendly Explanations

Considering that explanations are first and foremost intended as a means to make humans understand ML models' decisions, it is important to dissect what humans consider good explanations. Once more Miller's work provides an insightful analysis of what humans consider good explanations [48]. The following properties are presented as the main aspects to take into account when referring to human-friendly explanations and what it implies for ML explainability. Thereby, explanations are:

- **Contrastive**: Humans tend to ask why a decision was made and not another, instead of simply asking why a certain decision was made. In other words, "How would the output change if input X had been different?". For example, in a loan application scenario where the user got a rejection, he would not be interested in knowing all the factors that usually lead to the rejection of a loan request, but he would want to know the difference between his application and the *"would-be-accepted"* version of his application, i.e., which factors in his application would need to change for him to get the loan. Thus, the best explanation highlights the difference between the object of interest and the reference object. This means that creating counterfactual explanations is application-dependent, because it requires a reference point (a prototype), which can be generated, an instance of the training set, or an average of representative instances of the training set.
- **Selective**: People usually select one or two main causes to explain an event, rather than preferring a complete list of causes. Consequently, explainable ML methods should be able to provide selected explanations or at least highlight a subset of the whole set of causes as the most important reasons for a decision.
- **Social**: As already mentioned in section 3.1.3, explanations involve a social process, in which there occurs a transfer of knowledge (an interaction) between the explainer and the

explainee. Therefore, the social environment and the target audience have to be taken into account when designing explainable ML systems.

- **Abnormal**: People tend to focus more on abnormal causes to explain events, i.e., causes that had small probabilities of happening but nevertheless happened. Eliminating these causes would have changed the outcome significantly (counterfactual faithfulness), which is why humans consider explanations that contemplate these abnormalities good explanations. For explainable ML systems this means that if some input feature that influenced a prediction was abnormal, then it should be included in an explanation, even if some other "normal" features contributed equally to the output.

- **Truthful**: Good explanations are true in reality, i.e., in other situations. However, it is more important that an explanation be selective. An explanation that does not list all possible causes is not considering all factors that interacted to generate a determined output. Although truthful, such an explanation would be intractable for a human to understand. Therefore, an explanation should predict an outcome as truthfully as possible, but always ensuring selectivity, contrastiveness and adequacy to the target audience.

- **Prior belief-consistent**: Humans tend to ignore information that does not align with their prior beliefs, a phenomenon known as *confirmation bias* [51]. Although good explanations are consistent with the explainee's prior beliefs, this is particularly difficult to integrate in ML systems and would probably affect predictive performance, as well as constituting a trade-off with truthfullness. One possible approach could be to enforce constraints like monotonicity to ensure that a certain feature can only influence the prediction in one direction.

- **General and probable**: A good explanation contains a cause that is able to explain a reasonable amount of events. Although this notion seems to contradict the abnormality property, one has to remember that abnormal causes are, by definition, rare, which means that when no abnormal causes arise, a good explanation should be general and probable. Regarding ML, the generality of an explanation can be measured by the feature's support, the number of instances to which the explanation can be applied divided by the total number of instances.

Some of the described properties are more relevant than others depending on the context and some are easier to incorporate in ML systems than others. However, it is always good to keep these principles in mind when designing such systems, because the target user of explainable ML systems are humans and what better way to generate human-friendly explanations than by introducing interdisciplinary knowledge from psychology and social sciences, disciplines which have been studying human explanations for far longer than the ML community?

For a more in-depth analysis we refer the reader to the extensive and insightful work of Miller [48].

3.2 Relevance

With the increasing ubiquitousness of AI, in particular ML systems, and their application to complex real world scenarios and critical areas like medicine, criminal justice and financial markets [40], concerns about properties such as safety, nondiscrimination and right to explanation arose [41, 50]. Explainability is particularly relevant in domains where decisions can have significant consequences [52].

As stated in the work of Goodman *et al.* [53], the new GDPR and its policy on the right to explanation for algorithmic decisions, which as of May 2018 entered into application in the EU, accentuates the growing importance of explainability in algorithm design, in order to build such algorithms that can operate within this new legal framework. This does not imply the need to explain everything all the time, but rather the possibility of doing so on demand [54].

Such legislation is particularly important considering that in the past years several controversies regarding AI systems producing discriminatory results have surfaced. In fact, in 2016, Angwin *et al.* [55] analysed the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) system, a widely used criminal risk assessment tool, and concluded that its predictions presented a racial bias. Moreover, DNNs are known for being vulnerable to adversarial attacks, in which it is possible to change the classification of one image by altering in an imperceptible way some pixels of the input image [56, 57, 58, 59]. Trojan attacks have also been demonstrated [60].

So, technically speaking, why do we need explainability? First of all, it is of the utmost importance that the following be absolutely clear: not every ML system requires explainability. Examples include air craft collision avoidance systems, which compute their outputs without human intervention. Therefore, explainability might not be needed when [41]:

- consequences of unacceptable results are not significant
- the problem in question is already sufficiently well-studied and/or validated in real world applications, such that people trust the system's decision, even if it is not perfect

However, Doshi-Velez *et al.* [41] argue that interpretability is needed when an *incompleteness* in the formalisation of the problem exists, leading to a barrier between optimization and evaluation or, in other words, when a fundamental *underspecification* in the problem exists, i.e., when more data or even a more complex algorithm will not increase the model's performance [42]. So, in the presence of an incompleteness, generating explanations might be a way to make the consequences of gaps in problem formalisation known to humans. Doshi-Velez *et al.* [42, 41] also provide some scenarios in which explainability may help diagnose underspecification:

- Scientific Understanding: the goal of humans is to gain knowledge about the world and, although there is not a complete statement of what knowledge is, explanations can be derived to afterwards be converted into knowledge.
- Safety: generally, an end-to-end system is not completely testable, meaning that is is impossible for one to create a complete list of possible failure scenarios, mainly because it

might be infeasible to enumerate all possible outputs given all possible inputs. Therefore, interpretability might help expose safety issues.

- Legal: one might be legally obliged to provide an explanation, as in the case of the GDPR's policy on the right to explanation discussed above.
- Ethics: humans may want to avoid discrimination and their notion of fairness may be too abstract to be included in the system. Even if some precautions are taken in relation to this particular topic, the model can have biases not considered *a priori*, such as patterns in the data that only became visible after the system was designed.
- Debugging: the system's developer might want to find out why the system is not working, in order to fix it, or to validate that it is working as expected.
- Mismatched Objectives: the algorithm might be optimising a proxy function for the ultimate goal, i.e., the algorithm's optimization objective is incomplete.
- Multi-objective Trade-offs: there can exist trade-offs concerning the objectives of the ML system, i.e., although desiderata for some system is well-defined, they can compete with each other, as happens in requirements such as privacy and prediction quality or privacy and nondiscrimination.

So, to sum up, we need explainability in AI/ML systems to debug, verify (quality control), improve and audit the systems, to learn from them, to comply to legislation [36, 61] and, finally, to achieve other desiderata of ML systems: fairness, unbiasedness, privacy, reliability, trust, robustness, causality and accountability [41].

3.3 Applications and Stakeholders

XAI can be applied and bring benefits to a wide range of real-world domains where AI and ML systems are used. As abovementioned, XAI can be remarkably useful in highly regulated and high-stakes decisions areas. One of such fields is transportation, which is nowadays concerned with autonomous driving and its ethical aspects. Autonomous vehicles need to be able to make split second decisions based on how they classify the objects in their surroundings. If misclassifications occur, the consequences can obviously be extremely dangerous. In fact, recently an Uber self-driving car killed a woman in Arizona, in what is believed to be the first fatality involving a fully autonomous car [62]. In such scenarios, not only is explainability needed to ensure accountability, but also to identify ambiguous classification situations and prevent them [36]. Some work is already being developed [63, 64], but it is far from being enough.

Healthcare is another domain where explainability is needed and can be applied to improve trust and reliability in ML systems. Such systems can be used to aid in diagnosing cancer, choosing the best course of treatment for a certain disease and several medical imaging-related tasks such as organ segmentation, for example. Works in the clinical domain such as [54, 65] and recent

Food and Drug Administration (FDA) approval of an AI device to detect diabetic retinopathy [66] reasserts the growing interest in employing such systems in healthcare.

Criminal justice is also a domain where AI can bring several benefits, like better recidivism risk assessment and cost reduction in both crime and incarceration [36]. In such high stakes decision areas, explainability is extremely important, mainly to guarantee nondiscrimination and fairness, and avoid situations like the one mentioned in the previous section, triggered by the COMPAS risk assessment software.

Besides having different applications, XAI also has different stakeholders with different intents and requirements. As explanations are ultimately dependent on the end-user of the system, it is important to not only take into account the application, but also the system's stakeholders. Preece *et al.* [67] identify four stakeholder communities: developers, theorists, ethicists and users.

The developer community consists of people who build AI systems. This group includes members from the industry, but also academic researchers, and is mainly interested in aspects like quality control, system testing and debugging, and improving system robustness.

Theorists also include researchers, both in academic and industry research units, but distinguish themselves from developers, because they are focused in advancing the state of the art in XAI, rather than in delivering practical applications. Such motivation to better understand fundamental properties of deep neural networks led to the discovery of the vulnerability of such networks to adversarial attacks.

Ethicists are concerned with fairness, accountability and transparency and may include computer scientists and engineers, but also social scientists, journalists, economists, lawyers and policy makers, making this the most interdisciplinary community. Their interests go far beyond the technical software aspects, ranging from fairness and unbiasedness to auditability and accountability, as well as compliance to legislation.

Finally, the users are, naturally, the people who use AI systems. This community needs explanations to aid in deciding according to the system's outputs. It is worth noting that this community can also overlap with the other communities considering, for example, an academic who developed an AI system (developer) to aid them in their research (user), while being concerned with the societal impact of their work (ethicist).

Since the end user and application give importance to different aspects of XAI systems, a sensible approach to building such systems would be to offer a system adaptable to the application domain and stakeholders involved, as suggested in [67]. This strategy will be further detailed in the next section concerning feasibility.

3.4 Feasibility and Challenges

Having discussed the relevance and applications of explainability, it is now important to address if it is feasible, particularly in the legal context; in other words, if it is possible to extract from AI systems the same kind of explanations that are nowadays required of humans. Doshi-Velez *et al.* [50] argue that it is, indeed, technically feasible, mainly because an explanation is not the same as transparency, in the sense that, as mentioned in subsection 3.1.2, explaining does not mean understanding every bit that flows in the system, no more than an explanation from a human being entails knowing every signal flow through every neuron.

Therefore, an explanation as required under the law can be summarised by the following properties, derived from the questions also presented in subsection 3.1.2: *local explanation* and *local counterfactual faithfulness*. A local explanation consists of explaining a specific decision, rather than the whole system's behaviour. Local counterfactual faithfulness relates to the fact that humans expect an explanation to be causal. Considering a credit loan prediction system, an example of a local explanation would be: person A got their loan denied because of payment history, while for person B it was insufficient income that resulted in the denial of the loan. An example of local counterfactual faithfulness would be that if a person was told that their income was the main reason for the denial of a loan, then they might expect that if that income increases, the system might change its previous decision and allow the loan.

Again, it is important to note that both these properties can be achieved without needing to know the process by which the system reached its decision, which addresses concerns regarding trade secrets, because an explanation can be provided without the need for uncovering the contents of said system. An example is also provided in [50]: if a legal question consists in knowing whether race biased a loan decision, then the AI system might be fed variations of the original inputs changing only the attribute race. If its outputs turn out to be different, then it is reasonable to argue that another attribute, such as gender, influenced the decision, which constitutes a legally sufficient explanation and no more information is needed under the law.

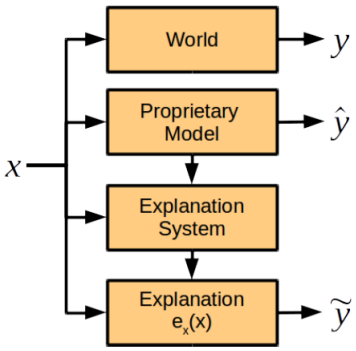


Figure 3.3: Framework for XAI systems, consisting of an explanation system, along with the proprietary model. With this approach, the ML model can remain protected, which addresses concerns regarding trade secrets. Extracted from [50].

Following this line of thought, in [50] the authors also propose a framework for explainable AI systems, as depicted in figure 3.3. It is argued that the model itself remain a black-box model, possibly proprietary, and that explanations are included in a separate system. Thus, the AI system can be optimized to produce predictions, $\hat{y}$, that match the real world, y, while the explanation system must provide a human-interpretable rule, $e_x(x)$, that takes in the same input as the prediction model and also outputs a prediction, $\tilde{y}$. So, to satisfy local counterfactual faithfulness, $\hat{y}$ and $\tilde{y}$ must be the same under small variations of input x. This framework allows for the concepts described above to be quantifiable: for any x, it can be checked if $\hat{y}$ equals $\tilde{y}$ and if these predictions remain consistent for small variations of x (x could be the race attribute as described in the aforementioned example), meaning that it can be measured during how much time an explanation system is faithful and the specific instances in which it is.

Another important aspect to take into account when building XAI systems, is the challenge that the *accuracy-explainability trade-off* poses. Generally speaking, the more accurate the model, the less explainable, and vice versa, as depicted in figure 3.4. Once more, finding the adequate point in this spectrum is tricky and highly dependent on the target domain and users of the system.

Figure 3.4: Accuracy-Explainability trade-off. The more accurate the model, the less explainable. DNNs are located on the far right of this spectrum, achieving outstanding predictive accuracies, but remaining black-boxes, while simpler models, such as Decision Trees or Linear/Logistic Regression exhibit lower accuracies, but are easier to understand.

As briefly mentioned in section 3.2, given these challenges that affect the feasibility of XAI systems, a sensible approach is to provide various levels of granularity of explanations, which make the system adaptable to different contexts of application. Preece *et al.* [67] suggest three layers of explanations:

- Layer 1 - Traceability: representation of internal states of the model, capable of showing that the system performed as expected, for example a saliency map of input layer features that led to the classification output of ”dog”. This first layer usually interests theorists and developers and not as much users.

- Layer 2 - Justification: representations linked to layer 1 that offer semantic relationships between input and output features, showing how the system performed as expected. These representations can be of different modalities, possibly presented simultaneously, complementing each other. An example would be a semantic annotation of the salient dog features. This layer is tailored to developers and users.

- Layer 3 - Assurance: representations linked to layer 2, designed to give their recipients confidence in the system's decisions, in more global terms than layer 2; for example counterfactual examples showing that the system does not classify a cat as a dog.

In addition to having different levels of explanations, combining different explainability methods is also a sensible approach, since such a system would be able to provide different modalities of explanations that can complement each other. Also, each user can choose which modality(ies) to analyse according to their preferences or expertise. An example of a possible system is depicted in figure 3.5. The proposed system predicts if a patient has cardiomegaly (a medical condition in which the heart presents an enlargement) based on chest X-Rays. The system is highly adaptable to the target audience, since it can output a prediction, its probabilities, a visual explanation, counterfactual examples and a semantic description. This way, the user can obtain complementary information from different sources, leading to a better understanding of the model and its decisions, as well as a greater flexibility to accommodate the target audience's preferences and knowledge.

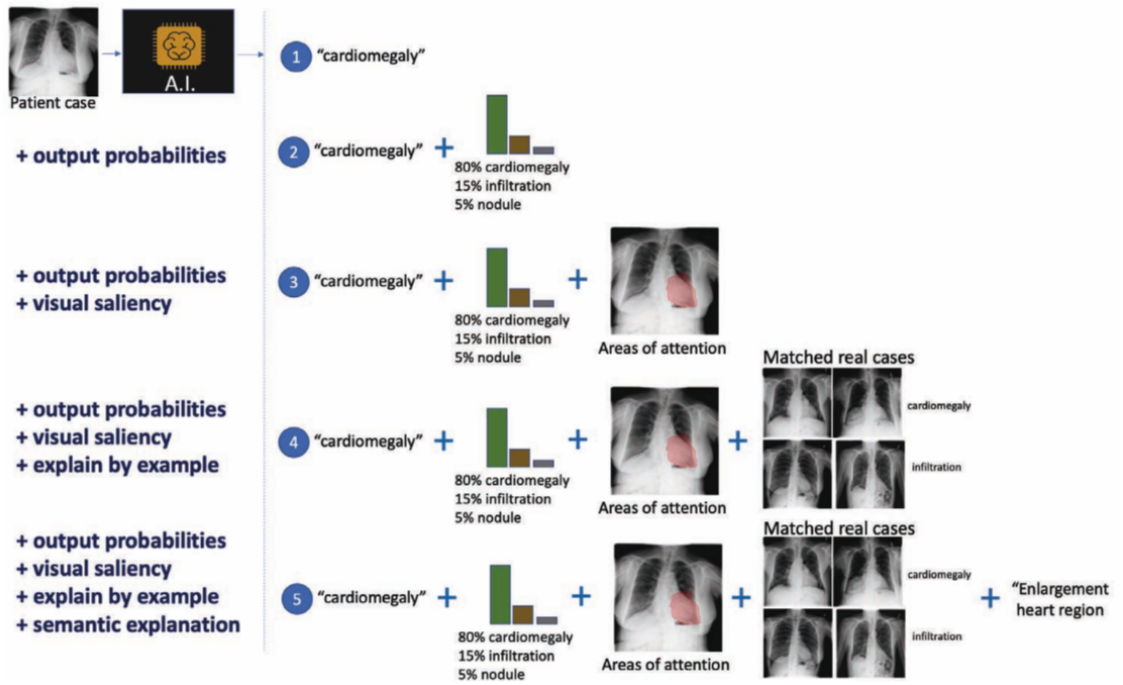


Figure 3.5: Combining explanation methods in a cardiomegaly prediction system. The system is able to produce a class label, output probabilities, visual saliency, counterfactual examples and a semantic explanation. Extracted from [68].

3.5 Taxonomy

Since explainable ML is still in its infancy as a research topic, the field lacks a clear and acknowledged taxonomy, similarly to what happens to the definitions of the core concepts such as explainability itself. Having such a taxonomy is extremely important to allow researchers to classify, evaluate and compare their methods.

This section focuses on categorising ML explainability methods according to different criteria presented in literature and partially surveyed in [69]. The mind-map of figure 3.6 represents the five taxonomies reviewed, as well as their subcategories.

Briefly, the locative criterion classifies a method according to its position relatively to the model, if it is applied before the model (pre-model), while building the model (in-model) or after the model (post-model or post-hoc). The agnosticity criterion simply divides methods as model-specific, tailored to a certain model, or model-agnostic, applicable to various types of models. The scope criterion separates local from global methods, while the output criterion, as the name implies, categorises the method according to its output, which can range from a feature summary to an intrinsically interpretable model. The transparency criterion relates to the internal mechanisms of the models. Finally, the methodology criterion distinguishes methods according to their function and purpose, be it investigate the processing or the representations of a model, for example. These taxonomies will be further detailed in the following sections.

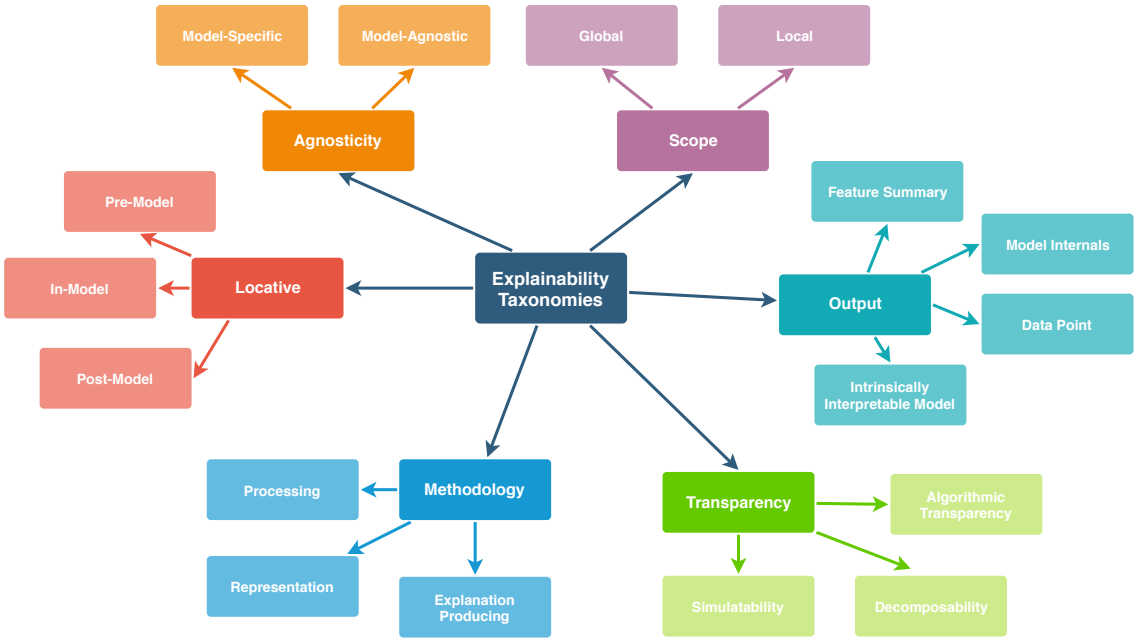


Figure 3.6: Mind-map of different criteria by which explainability methods can be categorised. A method can be described according to: its position relative to the ML model (locative criterion), its specificity towards the ML model (agnosticity criterion), its range (scope criterion), the type of output it produces (output criterion), its level of transparency (transparency criterion), or its methodology (methodology criterion).

3.5.1 Locative Criterion

ML explainability methods can be classified according to when the method is applied relatively to the model, i.e., if the method is applied before the model, when building the model or after the model [42]. Given this location property of the method in relation to the model and a lack of a unifying name for these three subcategories in the reviewed literature, we call this the Locative Criterion.

Pre-model Examples of pre-model methods include visualisation and exploratory data analysis. The main goal is to extract meaningful information from data before building a ML model. Exploratory data analysis can be achieved through clustering methods such as K-Means or K-Nearest Neighbours (K-NN) or through the use of prototypes and criticisms, as described in [52].

In-model Alternatively, explainability can be included within the model itself, when building it. This type of methods can also be said to be intrinsically interpretable [69]. These models can be rule-, per-feature-, case-, sparsity- or monotonicity-based.

Rule-based models normally include decision trees, rule sets and rule lists, all of which include a set of rules describing the different classes and predictions. Per-feature-based model examples include the linear and generalized linear models, as well as the generalized additive model [42]. Although these models are said to be explainable, one has to keep in mind that the size of the resulting model might invalidate its understandability, as the decision tree, the rule set or the features may become too complex for a human to keep track of. Nevertheless, a rule-based model is as interpretable as its original features [37].

Case-based models leverage the importance of example-based explanations found in human reasoning, for example when a doctor prescribes a treatment for patient X because it worked on a patient who had similar symptoms.

Sparsity-based models, as the name indicates, rely on the sparsity property. For example, the lower the number of activations in a neural network, the easier it is to find out which events led to the model's decisions. However, lowering the number of activations can render a decision impossible to achieve and thus, sparsity can reduce the model's accuracy [37].

Finally, monotonicity-based models guarantee the learnt function's monotonicity in relation to some of the inputs, which can facilitate its interpretability.

Post-model Lastly, there is post-model explainability, also known as post-hoc explainability. Post-hoc explainability entails selecting and training a black-box model and applying explainability methods only after the model is trained [69]. Several techniques for post-hoc explainability were proposed, such as Sensitivity Analysis, Saliency Maps, Backward Propagation, Mimic or Surrogate Models [42]. As in most cases this type of method is applied independently of the previously trained model, it can also be argued that a post-model/post-hoc explainability method is model-agnostic, as described in subsection 3.5.3.

3.5.2 Method Output Criterion

Another criterion for classifying an explainability method is proposed by [69] and it states that a method can be classified with respect to its outputs. Therefore, a method can output the following data.

Feature summary In this category, the output of the method can consist of a summary of how each feature affects model predictions, in terms of feature importance or feature interaction measures. The output can also be a visualisation of the summary statistics, such as Partial Dependence Plots (PDPs) [70].

Model internals The method produces a set of its internal parameters, such as the weights in linear models or the learned tree structure (features and thresholds) of decision trees. However, in linear models this classification overlaps with the previous typology, because the learned weights are both summary statistics for the model's features as well as its internal learned parameters. In CNNs one can consider the visualisation of learned feature detectors the method's output.

It is worth noting that this category superimposes both the aforementioned category of in-model methods (subsection 3.5.1) and the subsequent category of model-specific methods (subsection 3.5.3).

Data point The output can also be a data point, either an already existent or newly created point. Examples of these methods include counterfactual explanations or prototype identification. However, in order for these methods to be explainable, the outputted data points themselves have to be explainable, which is the case for images and texts, but not for tabular data with hundreds of features.

Surrogate intrinsically interpretable model The method generates an interpretable model, such as a linear regression, which approximates, locally or globally, the black box model. As mentioned before, the interpretable model itself is explained with the help of its model internal parameters or feature summary statistics, for example.

3.5.3 Agnosticity Criterion

According to [69], an explainability method can also be classified according to the agnosticity criterion, i.e., a method can be model-specific or model-agnostic. On one hand, model-specific methods are restricted to specific classes of models, for example, can only be used with neural networks. By definition, intrinsically interpretable models are model-specific. On the other hand, model-agnostic methods can be applied to any ML model and, by definition, are post-hoc methods, since they are applied after the model is trained. Thus, these methods do not have access to the model's internal parameters.

3.5.4 Transparency Criterion

Another criterion, proposed by Lipton [40], relates to the transparency property of interpretability methods, i.e., *"How does the model work?"*. As stated by the author, transparency, the opposite of *"blackbox-ness"*, is related to the understanding of the mechanisms by which the model works. The author defines different levels of transparency: entire model, individual components and training algorithm.

Simulatability Relates to transparency at the level of the entire model: a model is transparent if a human can consider the whole model at once, which suggests that an explainable model is a simple one. In other words, a model is completely understood if a human can, in reasonable amount of time and putting together input data and model parameters, produce a prediction by computing every necessary calculation. This notion of explainability is consistent with common claims that sparse linear models are more explainable than dense linear models for the same inputs. However, for some models the time it takes to make a prediction grows much slower than the size of the model (e.g. decision trees). This observation led the author to consider two sub types of simulatability, one based on the computation required to perform inference and other based on the size of the model.

Decomposability This level of transparency is tied to the notion that each part of the model is explainable. In this category, Lipton [40] considers model parts such as inputs, parameters and calculations. The author also highlights that this concept requires that the inputs be interpretable, which is often not the case when engineered or anonymous features exist.

Algorithmic Transparency As the name implies, algorithmic transparency consists of transparency at the level of the learning algorithm itself. Modern DL methods lack this type of transparency, because even though the heuristic optimization procedures are powerful, they are not fully understood nor can one guarantee *a priori* that they will converge on unseen data.

3.5.5 Scope Criterion

Additionally, a method can be considered local or global, in the sense that it explains single predictions or the whole model [69].

Global This type of interpretability follows the same definition as presented by [40] and previously described in section 3.5.4. However, Molnar [69] argues that global model interpretability is very difficult to achieve in practice, mainly because humans are unable to fit into memory more than a few parameters. In fact, a feature space with more than 3 dimensions is inconceivable for humans. Notwithstanding, some models can be understood at a modular level, such as linear models, where the interpretable modules are its weights, or decision trees, where the interpretable modules are the splits and leaf nodes. Nevertheless, one must be careful because the interpretation

of a single weight assumes that the other weights remain constant, which does not happen in real world scenarios.

Local Local interpretabillity can be taken into account considering only a single prediction or groups of predictions. In the first case, only a single instance and its predicted output are considered. This is based on the fact that locally, a prediction might depend linearly or monotonously on some features, rather than presenting a complex dependence on those features [69]. This same concept can be applied to a group of predictions, which can be achieved using previously described techniques such as global model interpretability on a modular level or interpretability for a single prediction. To apply the global methods one can take the group of instances as if it were the whole dataset and use the global methods on that data. The individual explanations can be applied to each instance in the group and then aggregated for the whole group.

3.5.6 Methodology Criterion

Finally, Gilpin *et al.* [45] propose a taxonomy specific for DNNs, which categorises methods according to how they approach explainability and what they are trying to achieve; be it either the explanation of the network's processing or the network's representations, or an explanation-producing system. Adadi *et al.* [36] refer to this criterion as a methodological approach for evaluating interpretability, so we call it the Methodology Criterion.

Processing These methods aim at explaining the neural network's processing, i.e., the way the network digests information and turns it into predictions. The fundamental problem of such methods consists in finding ways to reduce the complexity of the millions of operations performed by DNNs. This is usually done by finding a proxy model that approximates the original model, but is easier to understand, or by creating a saliency map that highlights portions of the computations that are most relevant. Examples of methods that fall under this category are Linear Proxy Models [71], decomposition of DNNs into decision trees [72], automatic rule extraction or saliency mapping [73, 74, 75, 76, 77, 78, 79].

Representations Explaining DNNs' representations involves understanding the role and structure of the data that flows through the network, which can be done by layer, by unit or by vector. When examining the internal structure of the network by layer, the information flowing through each layer is considered all together, while by unit, single neurons or filters are considered individually, as proposed by Bau *et al.* [80] with a method known as Network Dissection. Furthermore, we can consider other representation spaces, which is the case with vector analysis methods, where other vector directions in the representation space besides layers and units are considered. The most significant example of an analysis of representation vectors is the work of Kim *et al.* known as Concept Activation Vectors (CAVs) [81].

Explanation-Producing Systems Gilpin *et al.* [45] distinguish a third type of explainability methods, which the authors call *"Explanation-Producing Systems"*. As the name suggests, these systems intrinsically produce explanations or include methods that make the network easier to understand. The most common examples are attention networks (networks trained with explicit attention) [82], disentangled representations (methods that unravel representations into separate dimensions describing meaningful and independent factors) [83] or generative explanations (neural networks that produce explanations explicitly as part of their training, such as Visual Question Answering [84]).

3.5.7 Comparison and Discussion

As reviewed along this section, there are many ways to divide and categorise explainability methods; six taxonomies were reviewed and their subcategories explained with examples of some methods belonging to each of these subgroups. Some taxonomies are completely different from one another, while some present a certain degree of similarity, or even overlap. Moreover, different taxonomies present different approaches to the categorisation of explainability methods, as well as varying degrees of granularity.

The scope taxonomy simply divides methods as global or local. Although this is a relevant property to take into account, it is far too wide a categorisation, because it does not distinguish methods like SmoothGrad [79] from Shapley Values [85]. The first method is local and model-specific, while the latter is also local, but model-agnostic.

Conversely, PDPs [86] are agnostic and global, while Counterfactual Explanations [87] are also agnostic, but local. This shows that the agnosticity criterion is also not enough to classify explainability methods.

The output criterion divides methods according to their output, which constitutes a coarse grained classification, which may not be enough to separate explainability methods. For example, it classifies PDPs and Shapley Values [85] as feature summary methods, although they are global and local methods, respectively.

The transparency criterion separates methods according to their levels of transparency, focusing on how the model works. This taxonomy lacks simplicity, which makes it harder to use in practice. Furthermore, one of its sub-classes, algorithmic transparency, which relates to transparency at the algorithm level and how the algorithm learns a model from the data, does not refer to the learned model or how its predictions are made. In fact, this kind of transparency only requires knowledge about the algorithm, for example, backpropagation. However, when talking about explainability, we are usually more interested in the generated models themselves and not only on the algorithms that produce them [47].

The methodology criterion distinguishes three types of models, according to how they approach explainability: emulating the processing, explaining the representations or inherently produce explanations. However, this taxonomy classifies only methods designed for neural networks, excluding other ML models, such as SVMs, for example.

The last reviewed taxonomy, and probably the most widely known and used, is the Locative Criterion, which divides methods as pre-, in- and post-model. Despite being a simple and discriminative taxonomy, it has a fuzzy barrier between in- and post-model methods. Although one might be initially inclined to say that all post-model methods are model-agnostic, since they are applied after the model is built, this is not always true. There are some methods that are post-model, but are model-specific, such as Model Compression [88], which is tailored for DNNs. By definition all intrinsically interpretable models (in-model) are model-specific, but not all post-model/post-hoc methods are model-agnostic.

Figure 3.7 summarises the overlap found between some of the taxonomies reviewed. In-model methods are model-specific and pre-model methods are model-agnostic, but post-model methods can be either model-specific or agnostic. The methodology criterion only divides methods designed for neural networks, making it a model-specific taxonomy. Yet two of its three classes, Representation and Processing, are included in the post-model category, while Explanation-Producing models are in-model.

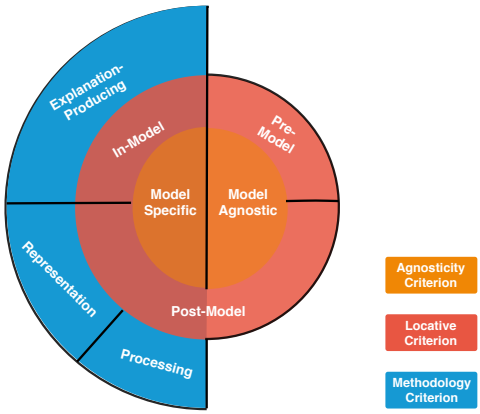


Figure 3.7: Differences and overlap between taxonomies: agnosticity, locative and methodology. The colours used for each taxonomy are consistent with the ones found in figure 3.6.

As detailed above, each taxonomy alone is not enough to divide most of the methods reviewed. As such, we propose the usage of two taxonomies together. First, we divide methods as model-specific (section 3.7) or model-agnostic (section 3.8). Model-agnostic methods are further divided into global or local methods. Then, since this work focuses on DNNs, we subdivide the model-specific methods according to the methodology criterion.

3.6 Evaluation

Just like there is little consensus on the definition of explainability or its taxonomy, there is still little consensus on how it should be evaluated and measured. As Silva *et al.* state in [37], the efficacy of an explanation is closely related to its ability to convince the target audience, making it susceptible to a myriad of subjective and intangible factors, such as the audience's willingness to trust an explanation or the audience's background knowledge. Despite this subjective nature,

finding ways to evaluate explainability methods is a pressing issue with the growing number of methods. So, the question that needs to be answered is *"Are all methods in all defined-to-be-interpretable model classes equally interpretable?"*. To answer their own question, Been Kim and Finale Doshi-Velez propose three levels for the evaluation of explainability methods in their works [42, 41]. The referred levels are discussed in the subsections below, presented by descending order of cost of application and validity of results [47].

3.6.1 Application-Grounded Evaluation

The first level, as described in [41], refers to an application-grounded evaluation, i.e., an evaluation involving **real humans** and **real tasks**. It consists in conducting human-subject experiments within the context of a real application. Since the goal is to test the quality of the explanations in a real-word scenario and with its end-user, the subjects are usually domain experts. A good baseline for this evaluation level is how well explanations produced by humans help other humans trying to fulfil the task. For example, if the final application of the model is to help doctors diagnose a certain disease, then showing that the model works entails evaluating it with respect to the task, i.e., doctors diagnosing.

Although this seems intuitive and it constitutes a real evaluation, it is costly and it makes it hard to compare works. However, this metric directly tests the system's goal, which means that performance relative to that goal is a strong evidence of the system's success.

3.6.2 Human-Grounded Evaluation

Human-grounded evaluation emerges effortless and naturally from the previous level, because it involves **real humans** but **simplified tasks**. The goal is to conduct simpler human-subject experiments that maintain the core of the end application. These experiments are useful when it is costly to run the experiments with the target community or one needs a bigger subject pool, for example, one can run the experiments with lay humans, instead of having to gather highly trained domain experts. Thus, this kind of evaluation is adequate for scenarios in which one wants to test more general notions of the quality of an explanation. Examples of experiments range from binary forced choice, where the subject has to choose the higher quality explanation between pairs of explanations, forward simulation/prediction, where the subject has access to an explanation and an input and has to correctly simulate the model's output, to counterfactual simulation, where the subject has access to an explanation, an input and an output, and is asked what must be changed in order to alter the model's prediction to a desired output.

3.6.3 Functionally-Grounded Evaluation

Lastly, Doshi-Velez *et al.* [41] define functionally-grounded evaluation as the use of a formal definition of interpretability as a proxy for assessing explanation quality. Thus, this evaluation requires **no humans** and only **proxy tasks**. The advantages of this evaluation are obvious: it allows sparing costs and time necessary to conduct human-subject experiments. Therefore, the

authors argue that this evaluation is more appropriate for cases in which the models have already been validated, e.g. via human-grounded experiments, or scenarios in which a method is not yet mature, or experiments with human-subjects are considered unethical.

The question now is what proxy functions to use. Examples range from the depth of a decision tree to proxies that model sparsity. Once a proxy is defined, the challenge is merely an optimization problem, as the model is probably discrete, non-convex and non-differentiable [41].

Figure 3.8 summarises the levels proposed by Doshi-Velez *et al.* in [41]. While application-grounded evaluation is the most specific approach, it is the most costly, since it involves human-subject experiments. On the other end of the spectrum lays functionally-grounded evaluation, which is the less costly. However, the results produced by such evaluation are low on validity, because the proxy tasks are not real measures of explainability. Human-grounded evaluation presents an acceptable trade-off, substituting the high cost of needing domain experts by laymen, while having higher validity, because they still incorporate human feedback, just from different domains.

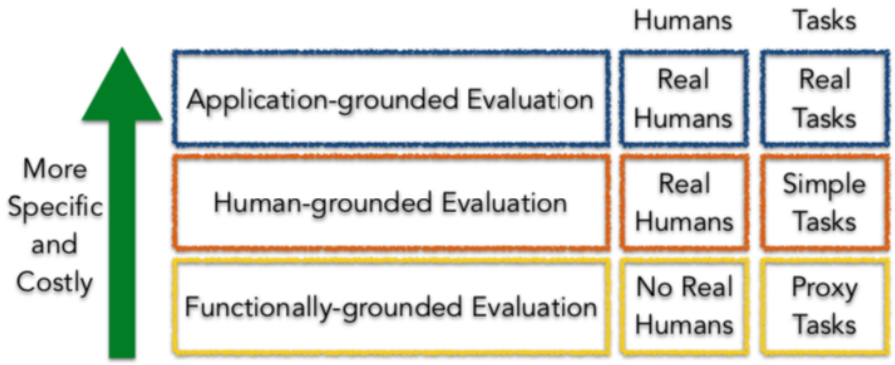


Figure 3.8: Taxonomy of evaluation levels for explainability. The most costly evaluation approach is application-grounded, involving real humans, usually domain experts, and real world tasks. Human-grounded evaluation is less costly and involves real humans, usually laymen, and simplified tasks. Finally, there is functionally-grounded evaluation, which involves no humans and proxy tasks. Extracted from [41].

Most of the work in the area focuses on functionally-grounded evaluation, with several proxy functions being proposed. In [37], Silva *et al.* present "The three Cs of interpretability": an explanation should be a simple model, applicable to local contexts of the data, and that maximises the following properties:

- **Completeness** - It should be possible to apply the explanation in other cases where its validity could be verifiable by the audience.
- **Correctness** - The explanation should generate trust, i.e., should be accurate.
- **Compactness** - The explanation should be succinct.

The authors use the fraction of the training set covered by the explanation as completeness, accuracy as correctness and the size in bytes of the explanation after compression as compactness. This last proxy function is based on the intuition that the time necessary to understand an explanation is proportional to its length.

Other proxy functions were proposed by Montavon *et al.* [43], such as:

- **Explanation Continuity** - This metric reflects a desirable property of explanation techniques; that they produce continuous explanations. It can be measured by computing the maximum variation of the explanation, $R(x)$, in the input domain.

- **Explanation Selectivity** - This is related to the desirable property that an explanation redistributes relevance to the most impacting variables with respect to the function value, $f(x)$. This metric, "pixel-flipping", was originally proposed by Bach *et al.* [74] and Samek *et al.* [89]. The authors quantify selectivity by measuring how fast $f(x)$ decreases when the features with the highest relevance scores are removed. A low AUC score represents a sharp drop in $f(x)$, which implies that the correct features have been identified as relevant. The method was first introduced in the context of image data, but was also applied to text data in [44].

When evaluating explainability, it is also important to find benchmarks and "reference" points, which are still very unexplored areas [68]. Gilpin *et al.* [45] propose using the occlusion method as a ground truth [90]. This method consists in testing the model in different versions of an input image, each with a different portion of the image occluded [73]. This can be thought of as a brute force sensitivity analysis that determines which parts of the input change the model's output the most, thus constituting a possible benchmark for other saliency methods. Therefore, these saliency methods can be evaluated according to how closely their produced maps are to the maps generated by occlusion.

Finally, when evaluating explainability one must be careful when using human evaluations. Human evaluations prefer simpler descriptions, which can lead researchers to produce persuasive instead of transparent systems [46].

In this work, the explanations generated by the proposed joint architecture are evaluated with a hybrid approach, between human- and functionally-grounded evaluation. The explainer is trained alongside the classifier, in a way that the classifier performance affects the produced explanations, so it indirectly constitutes a functionally-grounded evaluation. The visual aspect of the explanations are evaluated by humans, thus consisting in a human-grounded evaluation.

3.7 Model-Agnostic Methods

This section covers some of the most famous model-agnostic approaches. These methods have the advantage of not depending on a specific model, allowing for the possibility of explaining different models, without the need to change the explainability method, or to alter a certain method to comply with the model's structure. The majority of model-agnostic methods are post-model/post-hoc methods by nature, although some are pre-model methods (see figure 3.7).

The reviewed methods are divided into global or local methods, according to the scope of their explanations: if they try to explain the whole model or single/groups of predictions.

3.7.1 Global

The most widely known global model-agnostic methods explain the predictions of a model by computing the effect that some features have on the predicted outcome. Different ways of measuring this effect are proposed, giving rise to different methods, like PDPs [86] and ALE (Accumulated Local Effects) plots [91]. Both these methods reduce the prediction function to one that depends only on one or two features, by averaging the effects of the other features. They diverge in how these averages are computed [69]. Although simple to compute, these methods become harder to interpret when more than 3 features are considered and the effects of some features might actually cancel out [68]. Figure 3.9 presents an example of a PDP for a bicycle rental problem. These PDPs show the influence of temperature, humidity and wind speed on the amount of rented bicycles. Representations of ALE plots are similar to the ones presented in figure 3.9 and can be found in [69].

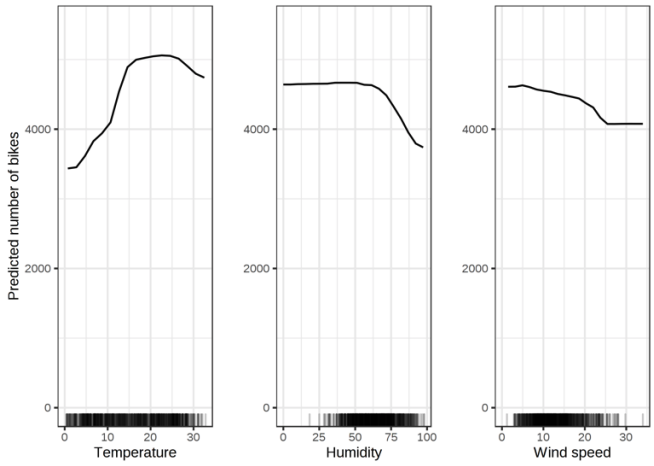


Figure 3.9: Example of some PDPs for a bike rental scenario and the relationship between predicted number of bikes and temperature, humidity and wind speed. From the first PDP one can see that the hotter, the more bikes are rented. This trend holds up until 20 degrees Celsius, flattening afterwards and dropping after 30 degrees. Vertical marks on the x-axis indicate the data distribution. Extracted from [69].

Another global model-agnostic method is Prototypes and Criticisms (or Maximum Mean Discrepancy-Critic, MMD-Critic) [52]. This is a pre-model method, which consists in finding the instances that best represent the data distribution, the prototypes, and the instances that most differ from the data distribution and are not well represented by the set of prototypes, the criticisms. Both prototypes and criticisms are found via greedy-search. This method is considered as a pre-model method, as it is applied before the model is trained: before training, prototypes and criticisms are found, then the ML model is trained as usual and afterwards the outcomes of the model are predicted on those same prototypes and criticisms [69].

All the described methods are global, because they focus on the model as a whole. While both PDPs and ALE plots output a feature summary statistic visualisation, MMD-Critic outputs already existent data points.

3.7.2 Local

Based on PDPs, Goldstein *et al.* [92] propose other type of plots of feature effect on the predicted outcomes, the Individual Conditional Expectation (ICE) plots. These plots simply represent the effect of a certain feature for each instance, and can be, therefore, considered local methods. In short, a PDP is the average of an ICE plot over all instances.

One of the most famous interpretability methods is Local Interpretable Model-agnostic Explanations (LIME); proposed by Ribeiro *et al.* originally in [71] and later reviewed in [93], it is a local model-agnostic method that outputs a Surrogate Intrinsically Interpretable Method.

As depicted in figure 3.10, the goal of surrogate/mimic models is to train a simpler model, usually an intrinsically interpretable model such as a linear regression, on the input data and on the output of the black-box model one is trying to mimic. Instead of training a global model, LIME only trains local models to explain predictions for individual instances.

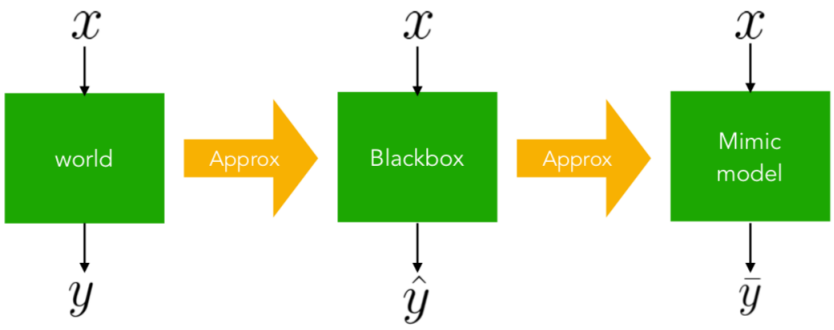


Figure 3.10: Diagram of a generic surrogate model. Extracted from [42].

More specifically, LIME's goal is *"...to identify an interpretable model over the interpretable representation that is locally faithful to the classifier..."* [93], i.e., to locally approximate, via an

interpretable model, the behaviour of a complex non-linear black-box model in the neighbourhood of the input being explained. The explanations, $\xi(\boldsymbol{x})$, are obtained by solving:

$$\xi(\boldsymbol{x}) = \underset{g \in G}{argmin} \mathcal{L}(f, g, \Pi_x) + \Omega(g) \tag{3.1}$$

where g denotes a given model, G is a class of potentially interpretable models, $\Omega(g)$ is a measure of complexity of g, f is the model being explained and Π_x is a proximity measure that defines locality around $\boldsymbol{x}$.

Thus, loss $\mathcal{L}(f, g, \Pi_x)$, e.g. mean-squared error, is a measure of how unfaithful g is in approximating f in the neighbourhood defined by Π_x. So, to ensure interpretability and local fidelity, one must minimise $\mathcal{L}(f, g, \Pi_x)$ and keep a low value for $\Omega(g)$, e.g. by preferring fewer features, so that the explanation is understandable by humans. In practice, LIME only optimises the loss part, so the user needs to determine the complexity, e.g. by selecting the maximum number of features that the linear regression model may use [69]. Figure 3.11 constitutes a toy example presented by the authors of LIME to explain the intuition for the method: the ML model's complex function f corresponds to the background image and it clearly cannot be globally approximated by a simple linear model. LIME samples instances, makes predictions on these instances and weighs how close the predictions are to the instances being explained. This proximity is represented by size in the picture. Finally, it fits a surrogate interpretable model, in this example a linear model, that is locally faithful to the original model.

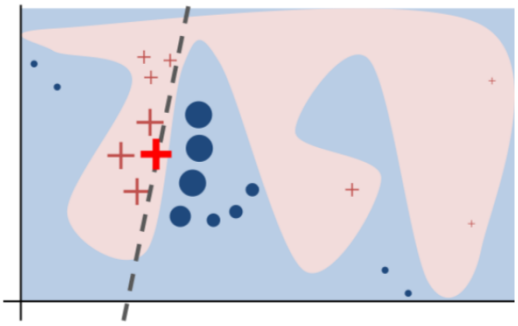


Figure 3.11: LIME toy example. The ML model's complex decision function f (unknown to LIME) is represented by the blue/pink background. The predictions made by the model on LIME's selected instances and their proximity to the instances being explained are represented by size. The dashed line is the learned surrogate model that is locally faithful. Extracted from [71].

The authors of LIME also proposed a method called Anchors [94], that instead of generating a linear model as a surrogate model for a specific prediction, generates simple rule-lists explaining that prediction. Each rule anchors a prediction if changes in other feature values do not affect the prediction [69].

Finally, another famous method is SHAP (SHapley Additive exPlanations), proposed by Lundberg *et al.* [85]. This method's goal is to explain the prediction of a single instance by computing

the contribution of each feature to that prediction, similarly to what PDPs and ALE plots do for all instances. This method computes Shapley values, an algorithm from coalitional game theory, where it is assumed that each feature is a ”player” in a game where the prediction is the payout, and that distributes this payout fairly among features [69].

SHAP allows visualisation of feature importance for a single prediction by means of several plots: explanation force plots, where it is shown how much each feature either increases or decreases the prediction; feature importance plots, where the features are ordered by decreasing importance and computed across all data; summary plots, that exhibit the feature importance (the Shapley value) for a feature and a specific instance; and feature dependence plots, where for each instance it associates the corresponding feature and Shapley values.

Except for MMD-Critic, all these model-agnostic methods, either global or local, explain models or predictions, respectively, by computing some kind of feature importance. They depend on visualisation tools to allow interpretation, which limits these models to less than three features, otherwise the interactions start to become too difficult for humans to understand. The reader is suggested to [69] for more model-agnostic methods.

Although model-agnostic methods have the advantage of being used with any ML model, without the need to retrain the model, they do not leverage intrinsic properties of specific methods, which hinders their explanation capability.

3.8 Model-Specific Methods: Explainability of DNNs

The high predictive capacity and complexity of DNNs have made them the object of study of most explainability methods. Therefore, and since this work focuses on CNNs, this section is dedicated to methods specific for DNNs. The reviewed methods are divided according to the Methodology taxonomy presented in section 3.5.6: network processing, network representation and explanation-producing systems.

3.8.1 Explaining Network Processing

DNNs have millions of operations, which means that explaining network processing involves finding ways of reducing the complexity of all these operations [45]. Saliency mapping, the creation of a *saliency map* that highlights small portions of the computations which are most relevant, is perhaps the most common type of method when one talks about explainable DNNs.

As mentioned in section 3.6.3, the occlusion method [73], a brute force sensitivity analysis that works by occluding parts of the image and feeding them to a network, in order to find out which parts influence the most the output, can be used as a benchmark for other saliency methods. If one has access to parameters of the DNN, saliency maps can be created by directly computing the input gradient [95].

Saliency mapping methods are gradient-based and the main idea behind them is that the gradient reflects the importance/relevance of input features to the predictions: the data point $\boldsymbol{x}$ is seen as

a sum of features $\left((x_i)_{i=1}^{d}\right)$ and each one is assigned a *relevance score*, R_i, determining how relevant each one is for explaining the individual decision. This is done by mapping back to the input domain a prediction for a certain image, creating a *heatmap*, which is the explanation. The various existing methods simply constitute alternatives for producing the aforementioned *relevance scores* [43], which involves propagating different quantities other than gradients [45], since such derivatives can miss important information that flows in the network.

These methods were also implemented in the context of [96], "iNNvestigate neural networks!", a library developed in order to provide a common interface and out-of-the-box implementation of different analysis methods, allowing for a systematic comparison between them. Some of the obtained results are shown in figure 3.12, where the analysis was conducted on ImageNet [97]. Each column shows results from different methods and each row corresponds to one input data image. For each input, the ground truth and predicted labels are also shown on the left, as well as the model's softmax output (prob) and its logit output before the softmax layer (logit). It is worth noting that all analyses took into consideration only the logit output.

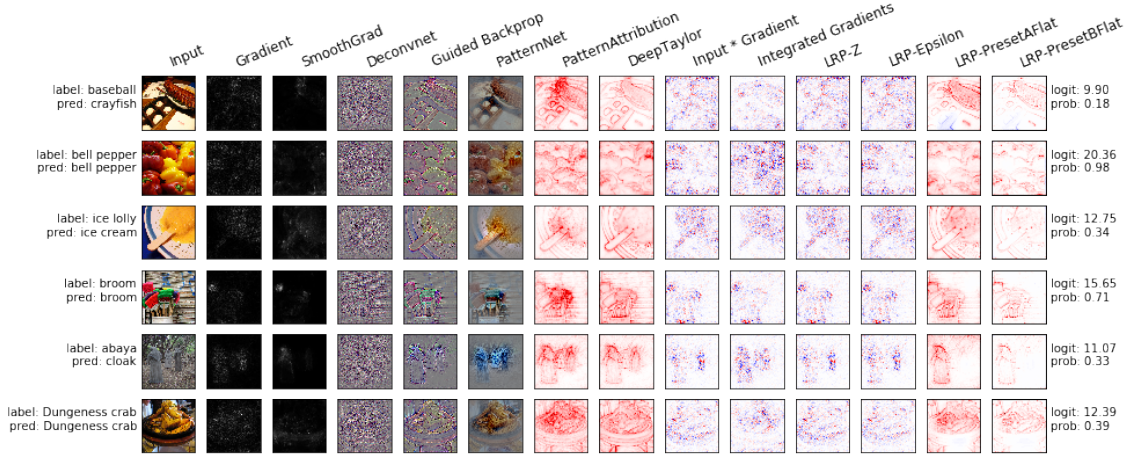


Figure 3.12: Results obtained with different methods via the iNNvestigate library. Extracted from https://github.com/albermax/innvestigate (Accessed on 12-02-2019).

According to Kindermans *et al.* [98], saliency methods can be divided into function, signal and attribution visualisation. These groups all present different but complementing information about the network, and are reviewed in the following subsections.

Function This category aims at explaining the function implemented by the model. In DNNs, the function is represented by the output neuron that encodes a certain concept [43]. So, *"...explaining the function in input space corresponds to describing the operations the model uses to extract y from x"*. However, in DNNs, which are highly nonlinear, it is only possible to approximate the complex function $f(x)$ [98]. The methods that will be described below constitute different ways of making that approximation.

Sensitivity Analysis estimates how moving along the model's locally evaluated gradient direction in input space influences y [43]. In this case, the most relevant features are those to which the output is most sensitive. The technique is easy to implement considering that the gradient can be computed using backpropagation during the DNN's training. However, its results are not great, as can be seen in figure 3.12. This behaviour can be explained if one remembers that sensitivity analysis only takes into account the local variation of $f(x)$ instead of its value, which means that sensitivity analysis only provides an explanation of the function's local slope.

Simple Taylor Decomposition is based on decomposing $f(x)$ as a sum of relevance scores using a Taylor expansion [43]. If one ignores the higher order terms of this expansion, which can be done for some piecewise linear functions, the relevance score becomes a product of sensitivity and saliency, i.e., a feature is relevant if it positively affects the model's prediction and if it is present in the input data. Simple Taylor Decomposition performs better than Sensitivity Analysis, but is characterised by a large amount of negative relevance.

SmoothGrad was originally proposed by Smilkov *et al.* in [79] as a way to visually sharpen gradient-based sensitivity maps. The core idea can be simply put as *"removing noise by adding noise"*: first take an image as input data, then sample similar images by adding noise to the original instance and finally compute the average of the sensitivity maps for each sampled instance. Since directly computing a local average in a high-dimensional input space is not feasible, the authors use a stochastic approximation. As can also be seen in figure 3.12, SmoothGrad slightly improves the saliency map's visualisation, smoothing some of its noise.

Signal According to [98], a signal detected by a neural network is the component of the input data that caused the network's activations. The methods described below attempt to visualise this signal. Specifically, Guided BackProp and DeConvNet use the same algorithm as the saliency map, but use different approaches when dealing with the rectifier neuron. PatternNet emerges as an improvement over the other two approaches.

The **DeConvNet** was proposed by Zeiler *et al.* in [73] and constitutes a *"...way to map these activities back to the input pixel space, showing what input pattern originally caused a given activation in the feature maps."*. In the sentence, the authors refer to *"these activities"* as the feature activities in the intermediate layers of CNNs. As explained by them, this architecture consists of a reverse CNN, that uses the same building blocks but in reverse direction, mapping features to pixels. Therefore, examining a CNN involves attaching a DeConvNet to each layer. As a result, the obtained reconstruction from an activation resembles a small piece of the original image, weighted according to their contribution to the given feature activation. The authors also note that *"..these projections are not samples from the model..."* [73]. The results of applying this method are also shown in figure 3.12. Although it is not depicted, results obtained in [73] show that each layer's projections represent the hierarchical nature of the network features: layer 2 detects corners and edges, layer 3 captures more complex structures, such as similar textures and patterns, and the higher layers are more specific to each class [73].

Guided BackProp was proposed by Springenberg *et al.* in [99] and constitutes a variation of the DeConvNet previously described, solely in the way both methods handle backpropagation through the ReLUs.

PatternNet was proposed in [98] by Kindermans *et al.* and emerges as an improvement upon the DeConvNet and GuidedBackProp visualisations, after the authors discovered that for these approaches the direction of the filters did not coincide with the direction of the signal. In the paper, the input data $\boldsymbol{x}$ is modelled as the sum of a signal component, $\boldsymbol{s}$, and a distractor component, $\boldsymbol{d}$. The signal contributes to the output, whereas the distractor does not. Therefore, the goal is to estimate as best as possible the signal. PatternNet in particular makes use of a two-component signal estimator, tailored for ReLUs. Hence, this approach *"...yields a layer-wise back-projection of the estimated signal to input space."*, where the signal estimator is approximated by superposition of neuron-wise nonlinear two-component estimators in each layer. As can be seen in figure 3.12, this leads to a visual improvement of the activations compared to previously described methods.

Attribution Another approach for visualising neural networks involves attribution visualisation. As stated in [98], an attribution represents *"...how much the signal dimensions contribute to the output through the layers..."*. Several techniques were proposed to allow attribution visualisation, such as simply multiplying the Input by the Gradient [100], Layer-Wise Relevance Propagation (LRP) and Deep Taylor Decomposition (DTD). The concept of attribution is referred to as *relevance* in LRP. PatternAttribution emerges as an improvement over these approaches, just as PatternNet was an improvement over DeConvNet and GuidedBackProp.

LRP originally proposed in [74] and later reviewed in [43], is a backward propagation technique that, unlike DeConvNet and GuidedBackProp, possesses a conservation property, where the share of $f(x)$ received by each neuron is equally redistributed to lower layer neurons, from the network output to the input. Just as in previous methods, there is a first phase where a standard forward pass is applied and the activations at each layer are saved. The second phase is basically a backpropagation step, where specific propagation rules are applied. LRP applies one rule that, simply put, either redistributes relevance or "counter-relevance" to lower-layer neurons in proportion to their excitatory or inhibitory effect, respectively, on the activation of the neuron whose relevance is being distributed. This relevance redistribution rule can be understood by looking at figure 3.13, where positive relevance is shown in red and negative relevance in blue.

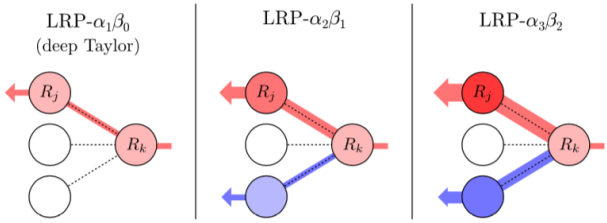


Figure 3.13: Relevance distribution process in LRP for one neuron, for different α and β values. Extracted from [43].

DTD When $\alpha = 1$ and $\beta = 0$, as shown in figure 3.13, the LRP is reduced to a DTD [101]: the LRP-$\alpha_1\beta_0$ rule applied to a given layer can be seen as the computation of a Taylor decomposition of the relevance at that layer onto a lower layer. The method's name then *"...arises from the iterative application of Taylor decomposition from the top layer down to the input layer."* [43]. Various results of LRP and DTD application are also shown in figure 3.12.

Deep-Lift Proposed by Shrikumar *et al.* [102], proceeds similarly to LRP in a backward fashion. In this case, each unit is assigned an attribution that represents the relative effect of the unit activated by an input compared to the activation by some reference input. The reference values are computed by running a forward pass using as input the reference image, which is usually chosen to be zero [103].

PatternAttribution was proposed alongside PatternNet in [98]. It is an extension of DTD, addressing one of the latter's main problems: the choice of root point used in the Taylor expansion. Thus, PatternAttribution can be seen as a root point estimator that learns from data.

Integrated Gradients was proposed in [78] and combines techniques employed by Gradient-based methods and LRP. This method simply consists in accumulating the computed gradients at all points along a straight line path from the baseline $\boldsymbol{x}'$ to input $\boldsymbol{x}$. Specifically, the integrated gradients are defined as the path integral of the gradients along the straight line path from the baseline $\boldsymbol{x}'$ to the input $\boldsymbol{x}$. This baseline input could be a black image for CNNs. In practice, the integrated gradients can be approximated by a summation of gradients at path points separated by sufficiently small intervals, which can be done simply by computing the gradients in a loop over the set of inputs. This method can, therefore, be easily incorporated into any DNN architecture. The results of this implementation in the context of the iNNvestigate framework are shown in figure 3.12.

Figure 3.14 presents a division of the reviewed methods, as proposed by [104]. DeepLIFT, Gradient $\times$ Input or Integrated Gradients are considered Gradient and Decomposition-based methods, while LRP is essentially a Decomposition method, from which many of the other methods can be derived. DeConvNet and Guided BackProp are deconvolution methods by nature, as they invert the normal operations of CNNs, in order to produce visual explanations.

As demonstrated by the number of methods briefly reviewed, saliency methods constitute a large percentage of the research in this field. Many alternatives were proposed, but most of these methods share a lot of their core ideas, as was concluded in [90]. All reviewed methods seem to produce visual appealing explanations (see figure 3.12). However, relying solely on visual inspection can be misleading [105]. In this work, *"Sanity Checks for Saliency Maps"* the authors found out that some methods, e.g. GuidedBackProp are independent of both the data and the model parameters. The authors base their findings in two tests, the model parameter randomisation test and the data randomisation test.

Work by Kindermans *et al.* [106], *"The (un)reliability of saliency methods"*, also found that adding a constant shift to the input data, which does not affect the model, causes several methods to incorrectly attribute, mainly Gradient $\times$ Input and Integrated Gradients.

Although saliency maps are powerful tools to gain intuition about the way the networks process the data, some fail basic tests like the ones mentioned above. Therefore, determining where these methods fail and ensuring that they consistently guarantee reliability for all possible transformations is imperative for these methods to be applied in critical areas, such as medicine [106]. There is also a pressing need for developing tests that identify these problems and for benchmarking and comparing the various existing methods.

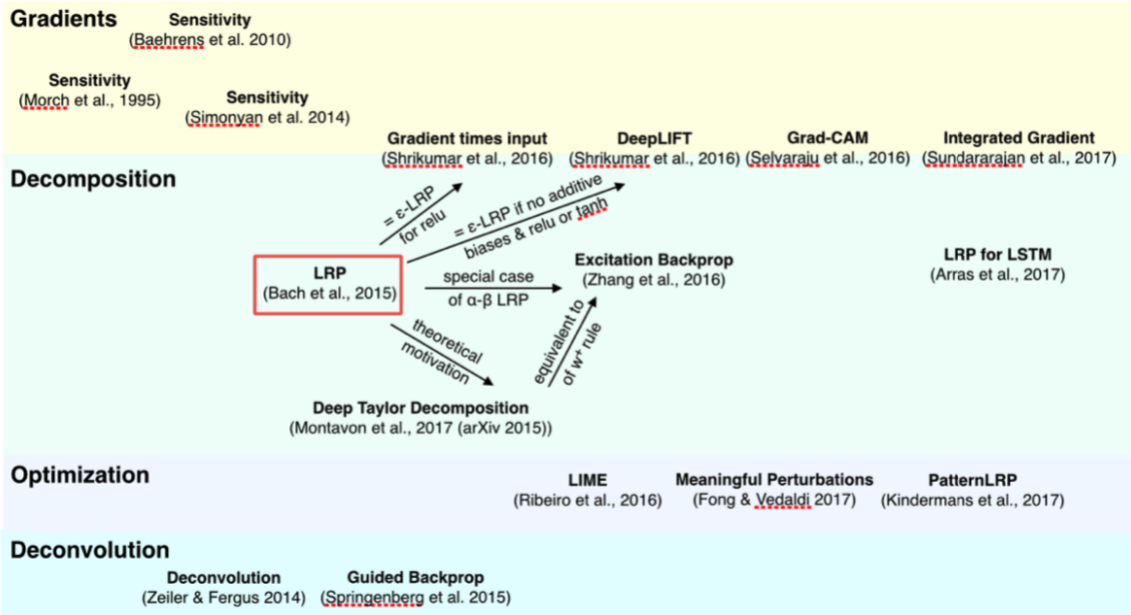


Figure 3.14: Categorisation of different saliency maps. Extracted from [104].

3.8.2 Explaining Network Representations

Despite the enormous amount of operations in a DNN, these are internally organised into smaller substructures. Explaining DNN representations focuses on understanding the role and structure of data flowing through the network [45]. This classification further subdivides methods according to the granularity examined: by layer, by unit or by vector.

As mentioned in section 3.5.6, when methods explain DNNs' representations by layer, the information flowing through one layer is considered all together. Transfer learning, the technique of using layers from one network to solve other problem, fits into this category and was proposed by Razavian *et al.* [107]. It is nowadays one of the most popular and useful techniques for training DNNs. The authors found out that the output of an hidden layer of a network trained to classify objects on ImageNet produced a feature vector that could be directly reused to solve other problems, such as fine-grained classification of species of a certain animal [45].

Explaining DNN representations by unit can be done by creating visualisations of the input patterns that maximise the response of a single unit (qualitative approach) or by testing the ability of a single unit, either a neuron or a convolutional filter, to solve a transfer problem (quantitative approach) [45]. These visualisations can be created either by optimising an image via gradient

descent [95], by sampling images that maximise activations [108] or by training a generative network to create such images [109]. An example of such visualisations can be found in figure 3.15, where the class specific spatial support for the class Goose is visualised. Regarding the quantitative approach, methods like Network Dissection [80] measure the ability of individual units to identify objects, parts, colours and textures not present in the training set, thus allowing the characterisation of the kind of structures present in each unit of the network [45]. Net2Vec [110] is another example of a method that quantifies how concepts are encoded by filters in DNNs, showing that multiple filters are required to encode a concept and often filters are not concept-specific and help encode multiple concepts.

Figure 3.15: Class saliency visualisation for the Goose class. Extracted from [95].

Pruning of networks [111] is another way to understand the role of neurons: the authors showed that large networks contain smaller networks with initialisations conducive to optimisation, which means that there exist training strategies capable of solving the same problem, but with much smaller networks that may be more explainable [45].

A review of methods that focus on understanding unit representations can be found in [112].

Finally, one can also characterise other directions in the representation vector space formed by linear combinations of individual units [45]. The most famous and recent approach in this category is Testing with Concept Activation Vectors (TCAV) [81]. Figure 3.16 shows how TCAV explains a DNN's internal state in terms of human-friendly concepts. TCAV aims at giving quantitative explanations: how much one concept was important to a prediction, even if that concept was not part of the training. In order to be applicable, this method needs:

- (a) a set of examples for a concept, e.g. "striped", and some random examples
- (b) labelled instances of the class being studied, e.g. zebra
- (c) a trained network

Having this, TCAV quantifies the model's sensitivity to the concept for that class resorting to Concept Activation Vectors, the red arrow in figure 3.16 (d), that are learned by training a linear classifier to distinguish between activations produced by the examples of the concept and activations in any layer. The CAV is the vector orthogonal to the decision boundary. In order to quantify the conceptual sensitivity of the class of interest, TCAV uses the directional derivative (e). This method is a post-model method, since it does not require the network to be retrained. However,

CAVs can also be used to sort images with respect to their relation to the concept, making this an interesting exploratory analysis tool to better understand and spot biases in the training data.

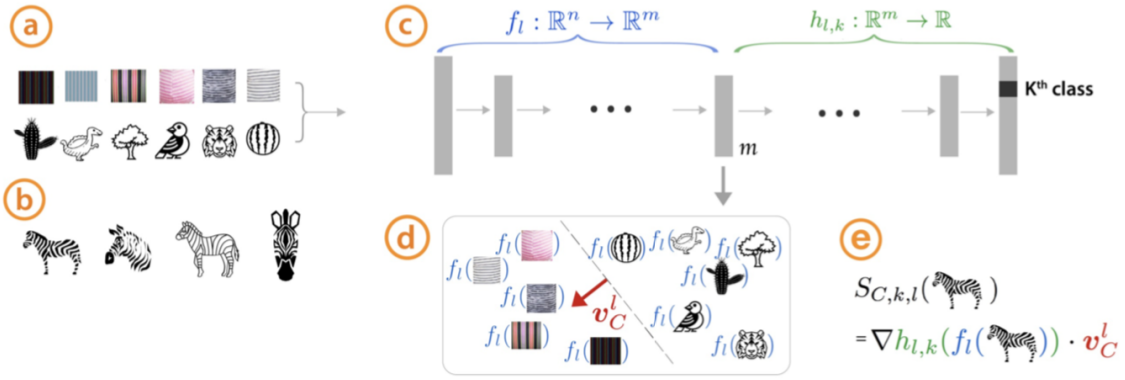


Figure 3.16: TCAV explanation pipeline. Given a user-defined set of examples for a concept (e.g. "striped") and random examples (a), plus labelled data examples for the class being studied (e.g. "zebra") (b), and a trained network (c), TCAV quantifies the sensitivity of the model to the concept for the specified class. This quantification is done through CAVs, which are learned by a linear classifier that distinguishes the activations produced by the examples of the concept and activations in any layer (d). The CAV is the vector orthogonal to the decision boundary (red arrow in d). The conceptual sensitivity is then calculated by the directional derivative (e). Extracted from [81].

3.8.3 Explanation-Producing Systems

The last category in the Methodology taxonomy is Explanation-Producing Systems. The goal of these methods is to create networks that are designed from the beginning to be easier to explain. Three ways of doing this have been presented in the reviewed literature: networks can be trained with explicit attention, can be trained to learn disentangled representations or trained to generate explanations [45].

Attention-based networks learn functions that weigh over inputs or features to steer that information towards other parts of the network. These networks were initially employed to natural language translation to allow the appropriate processing of words in non-sequential order [113]. Attention networks have also been used in fine-grained image classification [114], visual question answering [115] and image captioning [116]. Although modules that control attention are not trained explicitly to produce explanations, they reveal a map of the information that flows through the network, which can serve as a form of explanation [45].

Disentangled representations are representations that have individual dimensions, describing meaningful and independent factors of variation. Learning disentangled representations, or the problem of separating latent factors, can be used to create interpretable CNNs, whose individual units detect coherent meaningful patterns instead of the normal difficult-to-interpret mixtures of patterns, by means of special loss functions [83].

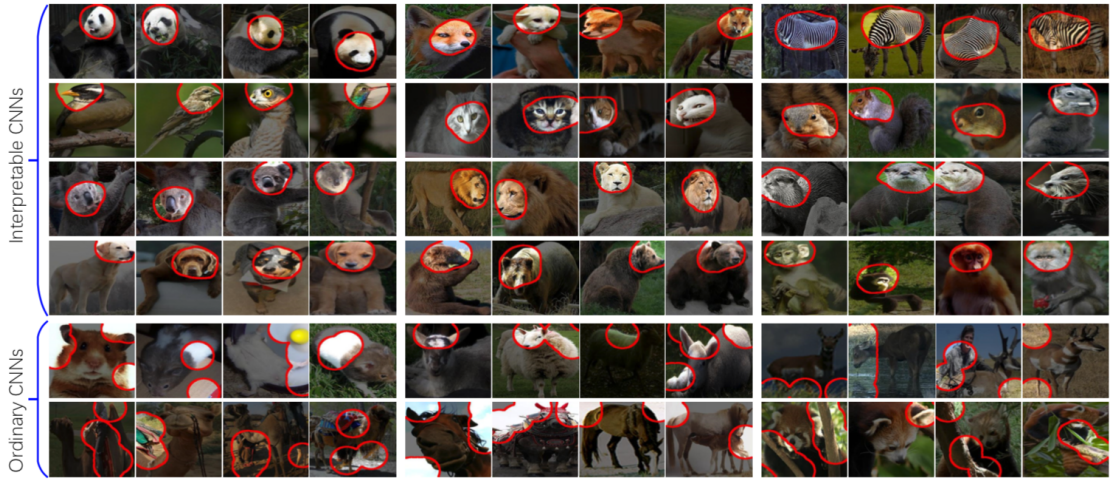


Figure 3.17: Interpretable vs Ordinary CNNs. The top 4 rows correspond to the visualisation of filters in top convolutional layers in interpretable CNNs, while the bottom 2 rows correspond to those same filters in ordinary CNNs. Interpretable CNNs focus more on the animals' heads, while ordinary CNNs focus on other parts of the images to classify the animals. The authors found that interpretable CNNs usually encoded head patterns of animals in the top convolutional layers. Extracted from [83].

Finally, networks can be trained to explicitly generate explanations. Figure 3.18 represents a typical block diagram for explanation generation networks. The black-box ML model is accompanied by an Explanation Method that produces an explanation based on the prediction. Explanation generation is still a very unexplored field, having been applied to visual question answering [84] and fine-grained classification [117]. In both these works, the systems generate a "because" sentence that explains the decision in natural language [45]. However, as far as this literature review went, there are no systems that explicitly generate visual explanations, and that is where this work is inserted, as will be further detailed in chapter 4.

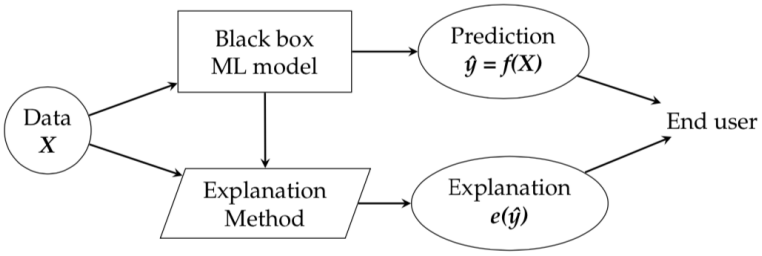


Figure 3.18: Block diagram of a typical explanation-producing system. The black-box model is accompanied by an Explanation Method that produces an explanation alongside the prediction. Extracted from [47].

3.9 Intrinsically Interpretable Models

This section briefly covers some models that the ML community considers intrinsically interpretable. As mentioned in section 3.5, these are in-model and model-specific by definition. These models are global on a modular level: their parameters and features are understandable for humans and allow one to perform manually the same calculations these models do, in order to reach the same outputs.

Interpretable models can be characterised by the following properties, that help increase their understandability [69]:

- Linearity - a model is linear if the mapping between features and target values is linear.
- Monotonicity - the relationship between an input feature and the target follows the same direction over the entire feature space. When an input feature increases (or decreases) it always leads to an increase or a decrease in the target value. This constraint can be enforced in any model to increase its interpretability, similarly to what Silva *et al.* did in [37].
- Sparsity - it improves understandability, because the less non-zero valued features, the easier it is for a human to assimilate all of them.
- Interaction - the ability to automatically include interactions between features. Interaction can also be included manually in any type of model through feature engineering [47]. Although it usually improves predictive performance, if too many or too complex interactions are present, interpretability will suffer.

The most widely known interpretable models are shown in table 3.1, where they are compared with regards to linearity, monotonicity and feature interaction, as well as to what tasks they can be used in.

Concisely, Linear Regression models predict the target as a weighted sum of the feature inputs. These models, as the name suggests, cannot be applied to classification tasks, so Logistic Regression models extend on Linear models by using the logistic function to squeeze the output of a linear equation between 0 and 1 [69]. Other extentions to linear models exist, being worth noting Generalised Linear Models (GLMs) and Generalised Additive Models (GAMs).

Linear and Logistic regression models fail when the relationship between features and outcome is nonlinear or when there is feature interaction. To overcome these drawbacks, one can resort to Decision Trees; the data is split multiple times according to certain thresholds in the features, creating subsets of said data, with each instance belonging to one subset [69].

Finally, the Naïve Bayes classifier is a probabilistic classifier based on the Bayes' theorem, along with assumptions on the independence between the features.

Table 3.1: Comparison between interpretable models according to desired properties, such as linearity, monotonicity and feature interaction. Adapted from [69].

Model	Linear	Monotone	Interaction	Task
Linear Regression	✓	✓	✗	Regression
Logistic Regression	✗	✓	✗	Classification
Decision Trees	✗	Some	✓	Classification and Regression
Naïve Bayes	✗	✓	✗	Classification

These models have been widely used in innumerous applications throughout the years, increasing their trustworthiness, even though their predictive performance is limited when compared to modern DNNs. However, one has to be particularly careful when saying these models are interpretable, because, in fact, they are only interpretable until they reach a certain level of complexity. For example, if one considers a complex decision tree which was given lots of input data and has a depth with an order of magnitude in the dozens, it is intractable to follow each path until a leaf node is reached, and therefore manually make predictions. As such, in this example, one cannot guess which feature was more important to the predicted output, rendering the model non interpretable [42].

3.10 Summary and Discussion

To close this chapter, a summary table of the reviewed methods in presented below. Each method is classified according to the agnosticity, scope, locative, methodology and output criteria. Also, the method proposed in this work, identified as **"Our Joint Approach"**, is positioned in relation to the other methods. It is a model-specific method, tailored to CNNs, that explicitly produces explanations as part of its training, making it an in-model, explanation-producing method, that outputs a feature summary visualisation in the form of a heatmap. It is a global method in the sense that it explains how the whole model works, by explicitly producing explanations as part of the training process, but it can also be considered a local method, because explanations are generated for single predictions, one at a time.

Table 3.2: Comparison of the reviewed methods according to the reviewed taxonomies and positioning of the method proposed in this work in relation to existing methods.

Method/Taxonomy	Agnosticity	Scope	Locative	Methodology	Output	References
PDP	AG	G	POST	NA	FS	[86]
ALE						[91]
Prototypes and Criticisms			PRE		DP	[52]
ICE		L	POST		FS	[92]
LIME					SIM	[71]
Anchors						[94]
SHAP					FS	[85]
Sensitivity Analysis	SP			PROC		[95]
STD						[43]
SmoothGrad						[79]
DeconvNet						[73]
GuidedBackprop						[99]
PatternNet						[98]
DTD						[101]
DeepLIFT						[102]
PatternAttribution						[98]
Integrated Gradients						[78]
Input × Gradient						[100]
LRP						[74]
Network Dissection	SP		IN	REP		[80]
Net2Vec						[110]
Other Representation Methods						[108, 109, 111]
TCAV			PRE/POST		DP/FS	[81]
Attention Networks		G	IN	PROD	FS	[114, 115, 116]
Disentangled Representations						[83]
Generated Explanations		G/L			DP/FS	[84, 117]
Our Joint Approach					FS	[118]
Linear/Logistic Regression		G		NA	MI	[119]
GLMs/GAMs						[119]
DT/RL						[120, 121, 122]

Agnosticity	SP - Specific; AG - Agnostic
Scope	L - Local; G - Global
Locative	PRE - Pre-Model; IN - In-Model; POS - Pos-Model
Methodology	PROC - Processing; REP - Representations; PROD - Explanation-producing
Output	FS - Feature Summary; SIM - Surrogate Interpretable Model; DP - Data Point; MI - Model Internals

A quick analysis of this table shows that the majority of existing explainability methods are post-model methods that aim at explaining network processing. Although they present the advantage of not needing for the model to be retrained, they also do not leverage aspects of the inherent processing, training and representations of DNNs. Besides, there is a growing need for models that simultaneously produce decisions and explanations, and only need an image as input, as part of a unified framework that can be trained end-to-end.

Other methods were reviewed, starting by model-agnostic methods, such as PDPs and LIME. Then, the focus shifted to model-specific methods, tailored for DNNs, since these are at the core of this work. Finally, intrinsically interpretable methods were briefly mentioned, as these are recognised by the community as trustworthy methods, because they have successfully been employed in critical areas for many years, despite their reduced predictive performance.

For more detailed descriptions of all of the reviewed methods, the reader is referred to each of its original papers, as shown in table 3.2, as well as to other surveys, such as [36, 45, 47], or [43, 76, 89, 96, 106] (reviews focused on saliency methods).

This chapter also focused on the definitions and taxonomies of explainability, as well as its relevance, feasibility, challenges, applications and evaluation. A lack of a universally accepted definition standard, of a clear taxonomy and of quantitative evaluation metrics was clearly identified and stems from the fact that interpretability and explainability are very subjective topics and, therefore, hard to formalise. Besides, these concepts are often domain-specific notions and depending on the context and target audience, different types of explanations might be preferred [47]. One possible way of tackling this problem is to progress towards systems that aggregate different types and levels of explanations, as was recently done in [123, 124].

Chapter 4

Proposed Joint Architecture

This chapter is dedicated to the proposed joint approach and starts by presenting some related work, namely the CNN architecture in which the proposed approach is based on. Afterwards, a detailed description of the proposed architecture is given, along with its integral parts: the classifier and the explainer. The chapter ends with the analysis of the proposed training procedure and losses.

4.1 Related Work

The proposed architecture was inspired by the work of Ferreira *et al.* [125]. The authors designed a CNN architecture tailored for Emotion Recognition, embedding domain knowledge in the network. The idea was *"...to explicitly drive the model towards the most relevant facial areas for the expression recognition, such as the facial components (i.e., eyes, eyebrows, nose, mouth) and expression wrinkles"* [125].

In order to do so, their architecture was comprised by an encoder-decoder component that predicts a relevance map ("Facial Parts Component") and a typical CNN architecture ("Representation Component") followed by a feedforward classification neural network ("Classification Component") that classifies the emotion shown in each input image. The output of the "Facial Parts Component", i.e., the relevance map, is used to filter the learned features in the "Representation Component", forcing them to strongly respond to the most relevant facial parts.

Besides the end-to-end architecture, the authors propose carefully-designed loss functions based on the strong prior that facial expressions result from motions of some facial muscles and components. Three supervision approaches are proposed: fully supervised, weakly supervised and a hybrid approach that combines the other two. The fully supervised scenario requires annotations of the coordinates of facial landmarks to create ground-truth relevance maps, and the Facial Parts loss is simply the mean squared error between the generated and those ground-truth relevance maps. The weakly supervised approach does not require ground-truth relevance maps and the Facial Parts loss regularises the activations of the relevance map by imposing sparsity and

spatial contiguity constraints. In all of the approaches, the Classification Component is trained with the categorical cross-entropy as loss function.

4.2 Overview

As reviewed in the previous chapter, the majority of the literature focuses only on interpretability, and more specifically on post-model or post-hoc interpretability methods. While a few works focus on in-model approaches, these are for the most part application oriented. Although such in-model methods exist for CNNs [37, 123], these are still not considered intrinsically interpretable, making this category still dominated by classic methods like decision trees.

To try and tackle this lack of in-model explainability methods, an end-to-end in-model joint approach is proposed, based on an explainer+classifier architecture. A simple block diagram of such architecture can be found in figure 4.1. The concretisation of the aforementioned model is depicted in figure 4.2, where a detailed diagram of the proposed architecture is shown.

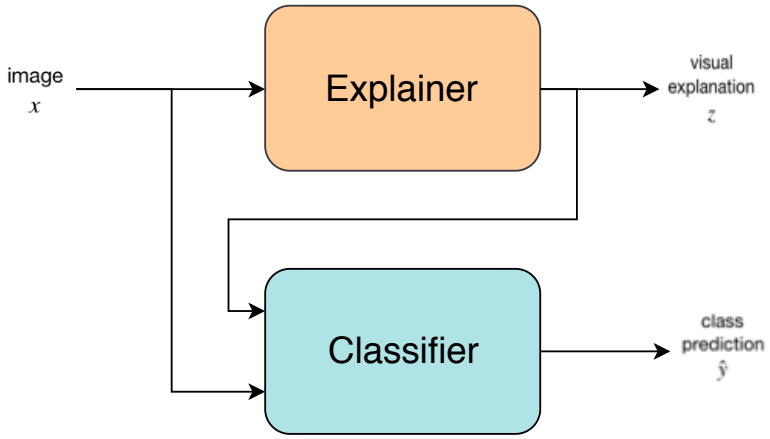


Figure 4.1: Proposed architecture block diagram. It is composed by an explainer and a classifier, outputting not only a class label, $\hat{y}$, but also a visual explanation, $\hat{z}$.

This architecture outputs a class prediction, $\hat{y}$, and what in this work is called a visual explanation of such decision, $\hat{z}$. The explainer takes as input an image, x, and outputs the corresponding visual explanation, which is an image with the same spatial dimensions as the input. On the other hand, the classifier takes as input an image, as well as the output of the explainer, i.e., it is trained using the explanation. Therefore, the classifier focuses only on the image regions the explainer deemed relevant and does not take into account regions where the explanation for that class is not present. This approach aligns with the intuition that, when explaining a decision, for example, whether or not an image contains a car, humans tend to first separate what is an object and what is not, and then proceed to look for patterns in the region where the object is in order to classify it correctly. Conversely, sometimes humans cannot tell if an object belongs to some class, but can tell which regions of the image do not contain said class [118].

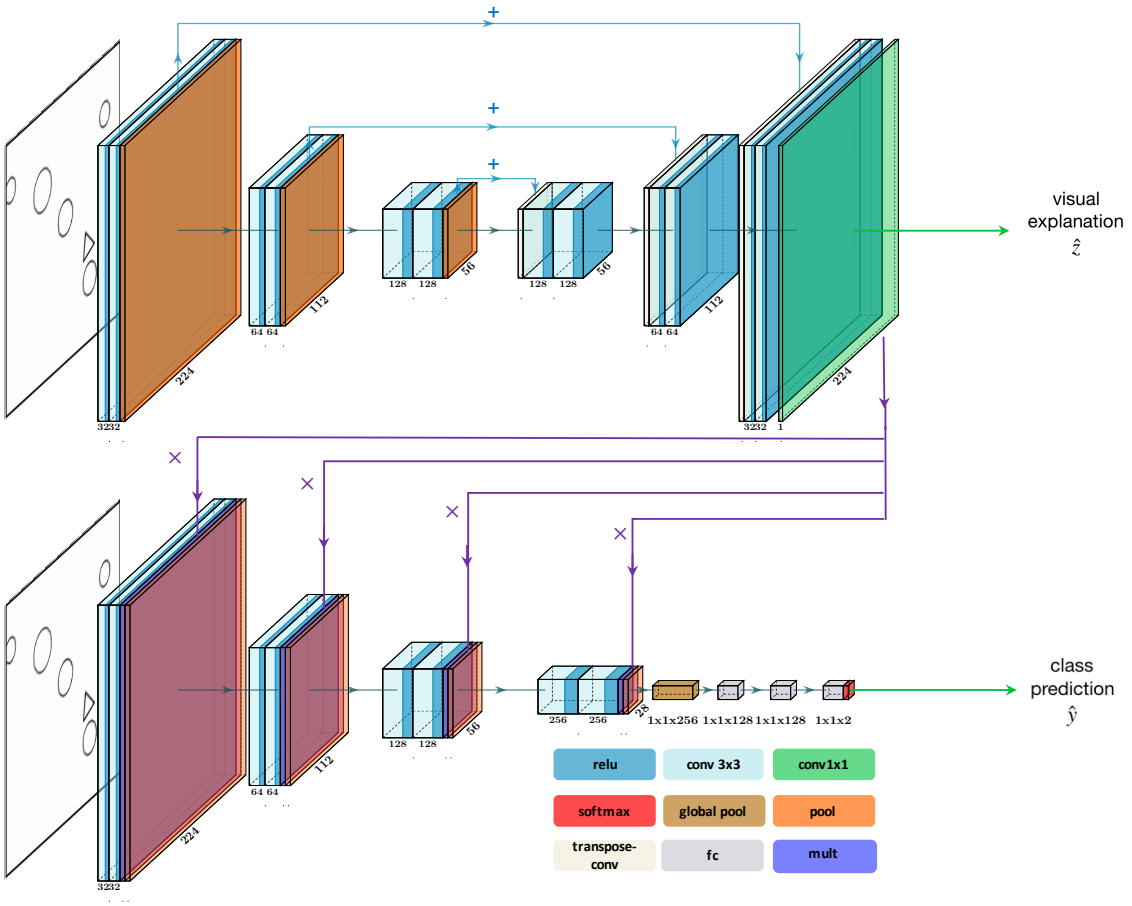


Figure 4.2: Detailed view of the proposed classifier+explainer architecture. The explainer (top row) is based on an encoder-decoder network and the classifier (bottom row) is based on VGG-16. The sums correspond to the simple addition of the outputs of the respective layers. The multiplications involve the concatenation and resizing of the explainer's output before computing the element-wise multiplication with the involved classifier layer.

4.3 Classifier

The classifier (bottom row in figure 4.2) is inspired by the VGG architecture [30], taking as input a Red-Green-Blue (RGB) image and having 4 consecutive convolution-convolution-pooling stages, ending with 3 fully connected layers, followed by a softmax layer. Each convolutional layer is composed by 3×3 kernels and ReLU activation functions. Each pooling layer uses a stride of 2, halving the layer's input spatial dimensions, and resorts to max pooling. The number of filters starts at 32 and decreases as a power of 2 with the stage level, while the spatial dimensions start at 224×224 and decrease also as a power of 2 with the stage level. Also, each pooling layer takes as input the result of a special multiplication layer, that is represented by the purple connections and layers shown in figure 4.2. This layer is responsible for the pixel-wise multiplication of the preceding classifier layer's output with the correctly downsampled version of the explainer's output. These connections are what allow the classifier to focus only on the important image

regions highlighted by the explainer and to discard regions where the explanation for the class being predicted is not present, as will be further detailed in section 5.3. After the 4 convolution-convolution-pooling stages a global pooling layer is introduced to flatten the resulting feature maps, in order to feed them to the dense layers. Finally, the network ends in two $1 \times 1 \times 128$ fully connected layers with sigmoid activation functions and 30% dropout rate, and 1 fully connected layer with a softmax activation, to allow multiclass classification.

However, it is worth mentioning that any classifier can be used, provided that the correct connections are introduced. In fact, the experiments performed on the cervical cancer dataset (refer to chapters 5 and 6) include the ResNet-50 [33] as a classifier, modified only to introduce the aforementioned connections. As the proposed architecture aims at explaining the classifier's decisions, the classifier should be chosen first, depending on the classification problem and the available data; the explainer must adapt to the classifier and not the other way around.

4.4 Explainer

The explainer (top row in figure 4.2) takes as input an RGB image from the chosen dataset and also outputs an RGB image, the visual explanation, $\hat{z}$, with the same spatial dimensions as the input image. It consists of a convolutional encoder-decoder network, based on U-Net [35].

The downsampling path, or encoder, is a simple convolution-convolution-pooling scheme repeated 3 times, similar to the classifier. Each convolutional layer uses 3×3 kernels and is followed by a ReLU activation layer. All pooling layers use max pooling and a downsampling factor of 2. The first convolution-convolution-pooling stage has 32 stacked filters and 224×224 feature maps. Afterwards, the number of filters increases as a power of 2 according to the stage level.

The upsampling path, or decoder, follows a "transpose convolution-convolution-convolution" scheme, where the first convolution operation is applied to the pixel-wise sum of the previous layer's output (output of the transpose convolutional layer) with the corresponding convolutional layer in the downsampling path. These connections allow for the subsequent layer to learn a more precise output. The whole decoder has 3 "transpose convolution-convolution-convolution" stages, ending in a convolutional layer with a 1×1 kernel, used to keep the spatial dimension space (height and width), but to reduce the filter space dimensionality, i.e., reduce the number of filters. In this specific case, the number of filters is reduced from 32 to 1. This last convolutional layer has a linear activation and is followed by a batch normalisation layer with an hyperbolic tangent activation. Conversely to what is done in the downsampling path, the number of filters starts at 128 and decreases as a power of 2 with each stage. On the other hand, the spatial dimensions increase in the same way.

4.5 Training

Training this architecture involves the three phases described below and depicted in figure 4.3.

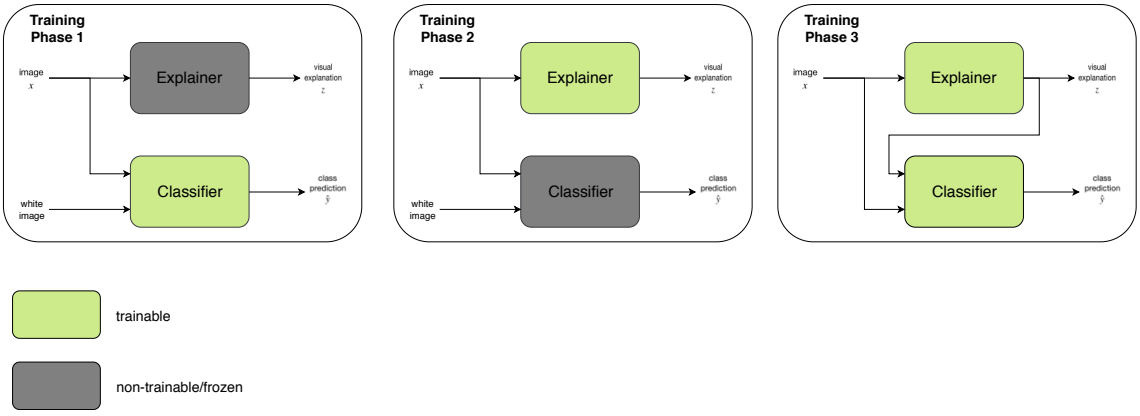


Figure 4.3: Training procedure of the proposed architecture. In the first phase only the classifier is trained, keeping the explainer frozen. Afterwards, the process is inverted. In the last phase, the whole architecture is fine tuned end-to-end.

First, in **phase 1**, only the classifier is trained, taking a ”white image”, i.e., a 2D array filled with ones, as explanation, while the explainer's layers remain frozen, i.e., unable to be updated. This means that the initial explanation is the whole image and that the classifier is not taking the explainer's output into account, as the explainer is not trained yet. As such, feeding the classifier a ”white image” renders the multiplications introduced by the explainer (purple arrows in figure 4.2) irrelevant. It is imperative that at the end of phase 1 the classifier remain somewhat unstable, i.e., that its loss does not *plateau*, so that in phase 3 the classifier can still learn from the explainer's output. Otherwise, in phase 3 the classifier would not update its parameters with the new information provided by the now trained explainer.

Afterwards, in **phase 2**, the process is reversed: the classifier is frozen and the explainer is the only module being updated by backpropagation. The classifier still takes as input a ”white image”, so that it outputs the same predictions as it did in phase 1, guaranteeing that, at this point, both explainer and classifier are still trained independently. If at this point the classifier had already made use of the explainer's output, it would be focusing on information provided by an explainer during its training process. In turn, based on this erroneous information, the classifier would predict during the forward pass and this prediction would alter the loss function accordingly, influencing the error propagation in the explainer, although not influencing the classifier's parameters. Then, in the next forward pass, the classifier would again use the explainer's output, produced by an inadequately updated explainer, and again predict accordingly, backpropagating this new error to both explainer and classifier, updating them. This process would repeat, culminating in an incorrect training of both explainer and classifier, as a result from this accumulation effect.

Finally, in **phase 3**, the whole architecture is fine-tuned end to end. The classifier is learning with the new information provided by the explainer, through the multiplications of the explainer's

output at every classifier's stage, as explained in section 4.3. As the classifier is training, ideally the global loss function is decreasing, improving the explainer as well. As the explainer is training, ideally the global loss function also decreases, and the classifier improves. So, in the end, both explainer and classifier improve, fruit of this dynamic interaction between the two and its respective loss functions.

4.5.1 Loss

The whole training process described above resorts to the end-to-end loss function defined in equation 4.1. This loss constitutes a weighted sum of the influence of both classifier and explainer ($\mathscr{L}_{class}$ and $\mathscr{L}_{expl}$, respectively), as realised by the α hyperparameter.

$$\mathscr{L} = \alpha \times \mathscr{L}_{class} + (1-\alpha) \times \mathscr{L}_{expl} \tag{4.1}$$

In general, $\alpha > 0.5$, because the correct classification is an indispensable part of the system, it directly affects the classifier and indirectly affects the explainer. In fact, through this joint loss, the explainer is always influenced by the classifier. However, a slight decrease in classification accuracy is tolerable, as long as it means that the system is also able to provide meaningful explanations. If $\alpha = 1$, then the explainer is being trained without any kind of regularisation of its outputs, being only indirectly influenced by the classification component via the global loss.

The classification loss, $\mathscr{L}_{class}$, defined in equation 4.2, is the commonly used categorical cross-entropy loss, employed in the majority of multiclass classification scenarios.

$$\mathscr{L}_{class} = -\sum_{i=1}^{N} y_i^{\top} \times log(\hat{y}_i) \tag{4.2}$$

where N is the number of instances in the training set, y_i is the one-hot encoded column array of the target class labels for instance i and $\hat{y}_i$ are the classifier's softmax predictions for instance i.

Regarding the explanation loss, two different approaches are proposed, an unsupervised and a weakly supervised loss, further detailed in subsections 4.5.1.1 and 4.5.1.2, respectively.

4.5.1.1 Unsupervised Explanation Loss

As it would be very hard to supervise the explainer's training, not only because explanations are user-, context- and domain-dependent, but also because an explanation constitutes a higher conceptual level, the explainer is trained with an unsupervised loss. This loss, represented in equation 4.3, is also a weighted sum of relevant properties one wants to promote on the resulting explanations, as realised by the β hyperparameter. These properties, sparsity and contiguity, are imposed by means of two regularisation terms: the penalised $l1$ norm and total variation ($\mathscr{L}_{sparsity}$ and $\mathscr{L}_{contiguity}$ respectively) [125].

$$\mathscr{L}_{expl\_unsup} = \beta \times \sum_{i=1}^{N} \mathscr{L}_{sparsity}(\hat{z}_i) + (1-\beta) \times \sum_{i=1}^{N} \mathscr{L}_{contiguity}(\hat{z}_i) \tag{4.3}$$

$$\mathscr{L}_{sparsity}(\hat{z}) = \frac{1}{m \times n} \sum_{i,j} |\hat{z}_{i,j}| \tag{4.4}$$

$$\mathscr{L}_{contiguity}(\hat{z}) = \frac{1}{m \times n} \sum_{i,j} |\hat{z}_{i+1,j} - \hat{z}_{i,j}| + |\hat{z}_{i,j+1} - \hat{z}_{i,j}| \tag{4.5}$$

where m and n represent the spatial dimensions of the feature map $\hat{z}$.

The penalised $l1$ norm (equation 4.4), promotes sparsity, as it shrinks less important features to zero, performing a kind of feature selection. This penalty works as an explanation budget, limiting the percentage of the input image that can be considered an explanation. The greater the penalty, the more regularisation, and therefore the less explanation budget is available.

The total variation factor (equation 4.5) promotes spatial contiguity, i.e., encourages smoothness and spatial localisation of the activations of $\hat{z}$, by minimising the local spatial transitions of $\hat{z}$. This factor was introduced because initially obtained results, further detailed in chapter 6, hinted at the need for sparse explanations, but also connected, i.e., explanations that connect semantically related parts of the images.

4.5.1.2 Weakly Supervised Explanation Loss

One way to impose stronger constraints on the produced explanations without needing to fully supervise their generation is to use a weakly supervised strategy, provided that some type of annotated data is available from another task. In this case, masks from object detection or segmentation tasks are needed. The aim is to drive the explanations not to focus on regions that are outside the interest regions; in the case of object detection annotations, the regions of interest would be the areas inside the ground-truth bounding boxes. Therefore, the weakly supervised explanation loss is defined as:

$$\mathscr{L}_{expl\_weakly} = \left| \frac{\sum_{i,j}(1 - z_{i,j}) \times \hat{z}_{i,j}}{\sum_{i,j}(1 - z_{i,j})} \right| \tag{4.6}$$

where $\hat{z}_{i,j}$ is the value of pixel (i, j) of feature map $\hat{z}$ and $z_{i,j}$ is the value of pixel (i, j) of mask z. This mask is an image where regions of interest are represented by ones, and other regions by zeros.

This loss punishes explanations outside the region of interest, by computing the product between the inverted mask $(1 - z_{i,j})$ and the explanation $(\hat{z}_{i,j})$: in the region of interest, the explanation is multiplied by zeros, and the explainer is not punished; outside the region of interest, the explanation is multiplied by ones, so the explainer will have to punish these pixels, by lowering their value from one, in order to decrease the explanation loss.

The divisor minimises the impact of the size of the regions of interest in the mask.

Although this approach needs masks from other tasks, it eliminates one of the hyperparameters, greatly reducing the time to tune the proposed architecture.

Chapter 5

Experimental Methodology

This chapter describes the experimental methodology employed during the course of this work. The first section describes the datasets in which the proposed architecture was tested. The developed synthetic data generation tool is detailed, followed by the characterisation of the real datasets used, namely 16-class-ImageNet [126], the Cue Conflict dataset by Geirhos *et al.* [127] and the Cervical Cancer dataset of the National Institute of Health (NIH) and the National Cancer Institute (NCI) [128].

Afterwards, data preprocessing steps and hyperparameter tuning strategies are discussed and, to close the chapter, a brief Explainer-Classifier Connection Study is presented.

5.1 Datasets

5.1.1 Synthetic Datasets

Data is the core ingredient in every DL pipeline. However, it is incredibly difficult to come by labelled datasets and it is highly expensive to collect and annotate one. Moreover, the explainability of a model or a particular instance is user- and domain-dependent. As such, a synthetic data generation tool with automatic annotation was developed[1].

This tool generates images containing simple polygons, like circles, triangles and squares. This way it is possible to analyse the proposed network's produced explanations without the need for expert knowledge, since it can be considered common knowledge, easily understandable by any human. Furthermore, a synthetic dataset entails other advantages, such as the generation of as much images as needed, the definition of the number of instances for each class, automatic annotation for different problems ranging from classification to detection and the inclusion of custom characteristics like overlap, occlusion, object type, object colour, image dimensions, etc.

Each polygon is defined by its p and q factors, where p is the number of vertices and q represents how these vertices are connected: the vertices are placed along the circumference that circumscribes the polygon with an equally spaced angle given by $\frac{2\pi}{p}$. For example, if q equals 3 then the first vertex connects to the fourth and so on. On the other hand, if p equals -1 then the

[1] The code is publicly available at https://github.com/icrto/xML.git.

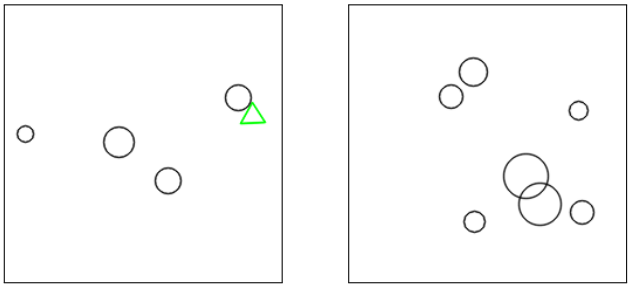
polygon is a circle and the value of q is ignored. Each polygon is randomly placed in the image, considering overlap, occlusion and rotation constraints. For each generated image, annotations are created in Pascal VOC format, containing information regarding image characteristics and also the classification problem being solved. The target polygon is given, along with information about its presence in the image. If the polygon is present in the image, then bounding boxes are provided for each instance of the desired polygon. This way, one dataset can be used for simple binary classification ("Is there at least one triangle in this image?"), multiclass classification ("Does this image contain 0, between 1 and 3 or more than 3 triangles?") or detection ("Where are the triangles located in each image?").

To make the tool easier to use and the generated images as custom as possible, various configuration parameters can be set in a JSON configuration file. A list of these configurable parameters can be found in A.1 and an example of a generated XML annotation file in Pascal VOC format can be found in appendix A.2. The pseudo-code description of the algorithm for the synthetic images' generation is presented below.

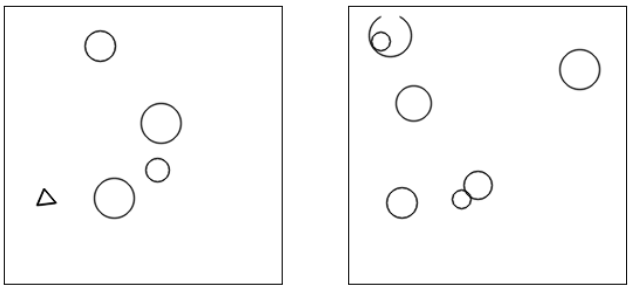
Algorithm 1: Synthetic Image Generation

```
Result: RGB image and corresponding XML annotation file in PASCAL VOC format
create image with the defined background colour
create annotation file
randomly choose number of polygons (between min_nr_shapes and max_nr_shapes)
count ← 0
tries ← 0
while (count < nr_shapes - 1) or (tries > nr_tries) do
    randomly choose p, q, x_orig, y_orig, rad and rotation angle for the polygon
    for every polygon in the image do
        check if polygons are overlapping
    end
    if polygons are NOT overlapping then
        add the centre coordinates of the new polygon to the list of polygons
        draw polygon by placing its vertices along the circumscribing circumference
        if drawn polygon is target polygon then
            append the polygon's bounding box coordinates to the corresponding XML
              annotation file
        end
        count ← count + 1
    else
        tries ← tries + 1
    end
end
update XML annotation file with number of target polygons drawn
```

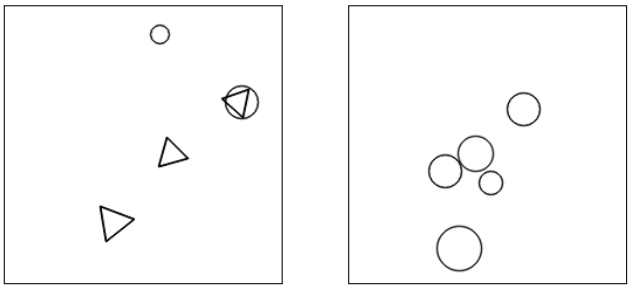
Figure 5.1 shows some examples of generated images. Preliminary experiments showed that a dataset composed by images similar to those of the fourth row **(d)** constituted an over complex classification problem. As such, only the first three datasets were chosen for the rest of the experiments with synthetic data.



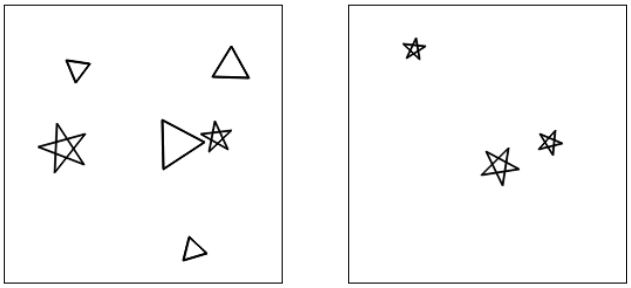
(a) Images from a simple dataset with colour cues



(b) Images from a simple dataset without colour cues



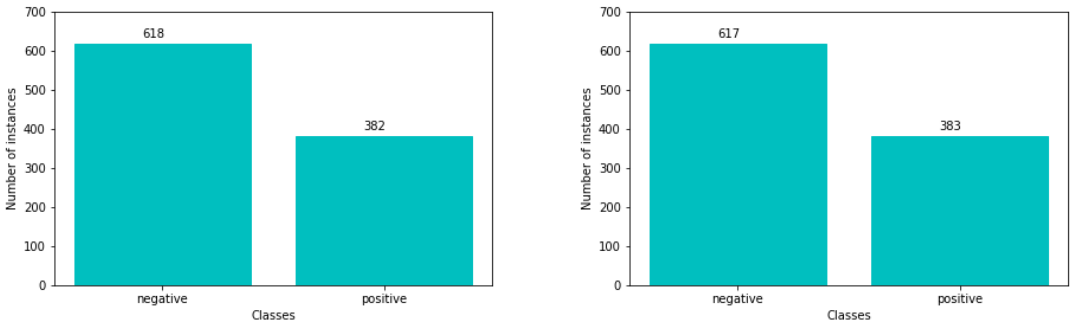
(c) Images from a dataset with multiple target polygons without colour cues



(d) Images from a dataset with multiple target polygons and 5-pointed stars without colour cues

Figure 5.1: Example images of 4 generated datasets. For each row, the left column illustrates an example of the positive class, while the right column illustrates the negative class. The target polygons are the triangles.

Finally, figure 5.2 depicts how many instances there are in each class for the two synthetic datasets used. Despite being possible to generate as much images as one needs, it was chosen to fix the number of training samples to 1000, as this is a rather simple binary classification problem that should not need more data, and also to reduce the duration of the training process. It was also decided that the datasets should present some degree of unbalance, as to mimic what happens in real world scenarios, mainly in medical contexts, when usually there are many more instances of healthy patients than of sick patients. Therefore, both datasets consist of 1000 224×224 RGB images, each containing a triangle (target polygon) and a variable number of circles. For each image it is only taken into account the presence (positive class) or absence (negative class) of the target polygon, thus making these experiments binary classification tasks.



(a) (b)

Figure 5.2: Simple dataset without (left) and with (right) colour cues' class distribution.

5.1.2 Real Datasets

5.1.2.1 16-class-ImageNet

16-class-ImageNet was first introduced in the work of Geirhos *et al.*, in which the authors develop an experimental paradigm to compare the generalisation abilities of human observers and neural networks [126]. This dataset is based on the ILSVRC 2012 database [129], in which more than 1000 fine-grained classes are present. However, such level of granularity is not needed in both the original paper nor in this work. This stems from the fact that humans tend to categorise objects into "entry-level" categories, such as dog instead of Labrador [126]. Therefore, the authors mapped the initial 1000 classes into 16 "entry-level" classes, using the WordNet hierarchy [130]: airplane, bicycle, boat, car, chair, dog, keyboard, oven, bear, bird, bottle, cat, clock, elephant, knife, truck. Figure 5.3 shows the number of instances per class. In total, this dataset is composed by 213555 RGB images, of which some examples are shown in figure 5.4. In fact, this dataset was not directly used in this work, but its description is needed for the next subsection.

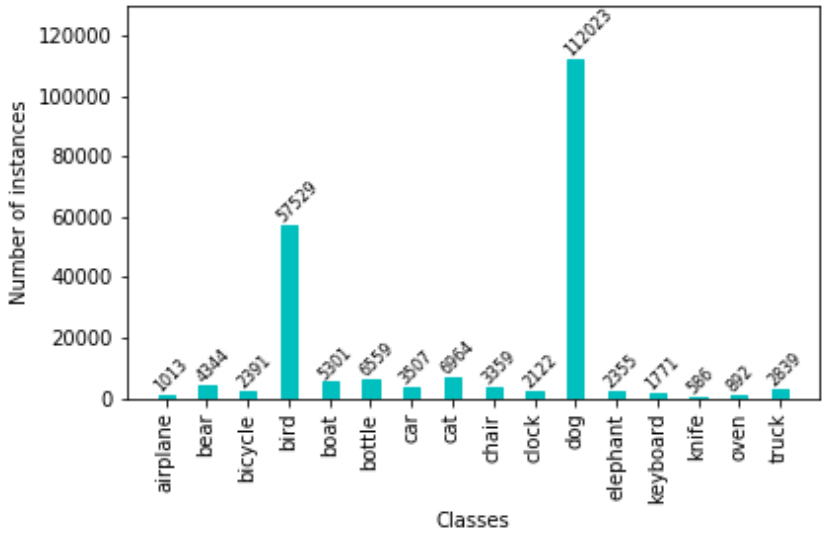


Figure 5.3: 16-class-ImageNet dataset class distribution.

(a) (b)

Figure 5.4: Example images from 16-class-ImageNet.

5.1.2.2 Cue Conflict Dataset

This dataset was first introduced in [127], where the authors validate that ImageNet-trained CNNs are biased towards texture. In order to test this hypothesis, the authors propose a cue conflict experiment in which style transfer is employed, introducing texture in images. Figure 5.5 represents the results of one of the conducted experiments with a standard ResNet-50 architecture. It can be seen that the network is able to correctly classify the texture and original images, but it fails to identify the image in which there is a texture-shape cue conflict.

This dataset[2] contains the same 16 classes mentioned in section 5.1.2.1, with 80 images each, making a total of 1280 images available in the dataset. Examples of such images are shown in figure 5.6.

[2]publicly available at https://github.com/rgeirhos/texture-vs-shape

(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan

(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat

(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

Figure 5.5: Experiments with shape and texture: classification accuracy of a standard ResNet-50 of **(a)** a texture image (elephant skin), **(b)** a normal (content-only) image of a cat and **(c)** an image with texture-shape cue conflict generated by means of style transfer between the first two images. Extracted from [127].

(a) shape class: cat
texture class: elephant

(b) shape class: bottle
texture class: clock

Figure 5.6: Example images from the Cue Conflict dataset.

5.1.2.3 NIH-NCI Cervical Cancer Dataset

This cervical cancer dataset is made publicly available by the NIH/NCI and was collected in the Guanacaste project [128].

Cervical cancer remains a significant cause of mortality in low-income countries [131] and is characterised by the abnormal growth of cells on the cervix, also known as Cervical Intraepithelial Neoplasia (CIN). CIN can be detected by means of a colposcopy, a procedure during which the cervix is examined under magnification. This visual examination of the cervix allows the detection of macroscopic changes in the tissue, such as colour and morphology, as depicted in figure 5.7.

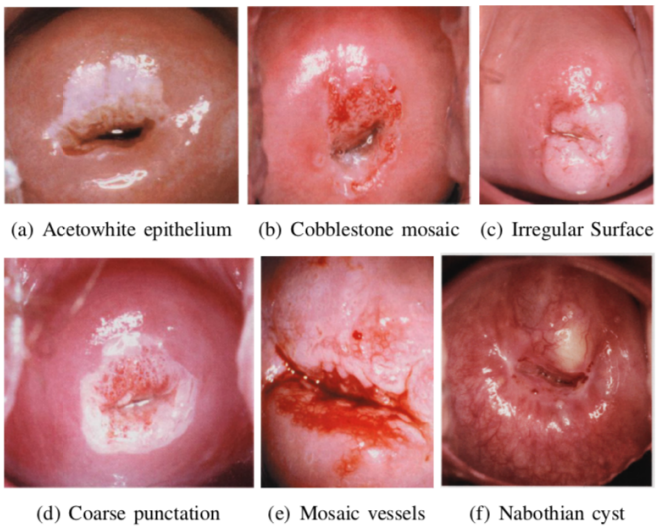


Figure 5.7: Types of cervical lesions identifiable through visual features present in photographs of the cervix. (a) Acetowhite epithelium that biopsy revealed as CIN2. (b) High-grade lesion with cobblestone mosaic. (c) High-grade lesion with dense acetowhite epithelium and an irregular surface contour. (d) High-grade lesion with coarse punctation. (e) Irregular mosaic vessels. (f) Large cyst. Extracted from [132].

More specifically, CIN refers to the potentially precancerous transformation of cells of the cervix and is graded on a 1-3 scale, with 1 being mild dysplasia, and 3 severe dysplasia. Therefore, in this database, the digital colposcopy images are annotated with the corresponding CIN progression level: normal, CIN1, CIN2, CIN3, and cancer. Bounding boxes for the cervix regions are also available. Figure 5.8 presents the class distribution for this dataset, which has 2120 images in total. Since the data is clearly unbalanced, a coarser division is made: instances with no histology[3] done, normal or less than CIN2 are considered as not cancer; the rest are treated as cancer.

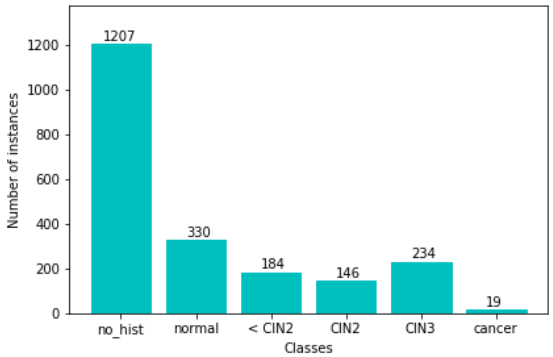


Figure 5.8: NIH-NCI Cervical Cancer dataset class distribution. The dataset is annotated according to CIN levels (CIN1, CIN2 and CIN3), as well as cases where no histology was done, normal cases and cancer cases.

[3]Usually when no histology is done it means that the patient's cervix was healthy and did not require any further testing.

(a) Healthy cervix. (b) Cancerous cervix.

Figure 5.9: Example images from the NIH-NCI Cervical Cancer dataset. The image on the left corresponds to a healthy cervix (positive instance) and the image on the right corresponds to a cancer case (negative instance).

5.1.3 Data Preprocessing

Synthetic and the cue conflict datasets were split into 75%-25% for training and validation, respectively, and 200 additional images were generated to compose the test set. In this test set, half of the images are from the positive class. Since all synthetic datasets and the cue conflict dataset were used to test the original architecture (classifier based on VGG-16), each image was normalised to an input pixel value range of $[0,1]$. The unsupervised approach does not involve any more processing steps, but, for the weakly supervised approach, masks were generated from bounding boxes, as described in section 4.5.1.2.

The NIH-NCI cervical cancer dataset was split into 80%-20% for training and testing, and the training set was further divided into 80%-20% for training and validation, respectively. Thereby, the training set has 1356 images, the validation set has 340 images and for testing there are 424 images. Since this dataset is used to test the architecture where the classifier is the ResNet-50 architecture, each image was normalised to an input pixel value range of $[-1,1]$, as recommended by Keras documentation[4] and implemented in their preprocessing function for this classifier. For the unsupervised approach, the images were cropped by the cervix bounding box, to evaluate this approach only on the region of interest, since it is out of the scope of this work to correctly segment the cervix before classification. Similarly to what was done in the synthetic datasets, masks were generated to be used in the weakly supervised approach.

5.2 Hyperparameter Tuning

Unfortunately, common hyperparameter tuning techniques, such as grid or random search, cannot be directly applied to tune the proposed network's hyperparameters. Traditional techniques involve training several models with different combinations of hyperparameters and automatically choosing the model that performs best according to the selected performance metric. Since no metric exists for assessing the performance of the explainer, this assessment is done by human visual inspection. This means that choosing the final model cannot be done automatically. However, the choice of hyperparameter values to test can still be done similarly to traditional techniques: by

[4] https://keras.io/applications/

performing an exhaustive search through a manually specified subset of the hyperparameter space (grid search) or by randomly selecting combinations from this subset (random search). Other techniques exist, but were not explored in this work, mainly due to the difficulty in applying them when no performance metric is defined.

5.2.1 Learning Rate and Number of Epochs

To train the original architecture, the Keras Adadelta optimiser[5] with default values was used, which employs an adaptive learning rate based on a moving window of gradient updates [18].

The ResNet-based joint architecture was trained using the Stochastic Gradient Descent (SGD) optimiser. The learning rate value was chosen by first ensuring that the classifier alone would converge to a minimum with said optimisation scheme. Testing started with an initial learning rate value of 1, that was divided by 10 until a converging model was found. Then, the whole architecture was tested with that same learning rate to ensure that both classifier and explainer reach acceptable results. This means that it is possible that the best learning rate to train the classifier is not the best to train the explainer, so one needs to find the learning rate that accommodates both networks' needs. In practice, it was verified that high learning rates that cause the classification loss to *plateau* very quickly, usually prevent the explainer to learn. So, it is preferable to slow the classifier training and ensure that the explainer has time to properly learn. After such analysis, the chosen learning rate for the ResNet-based architecture applied to the cervical cancer dataset was 0.001. A learning rate schedule was also employed, consisting in a reduction of learning rate by a factor of 0.2 every 10 epochs, during which the validation loss had *plateaued*, if a minimum of 0.00001 had not been reached.

The number of epochs was chosen in a similar empirical way: first only the classifier was trained to assess how many epochs it would need to converge. Then, roughly half of that number of epochs is employed in training phase 1 and training phase 3. The number of epochs for phase 2 was chosen according to how many epochs were needed to reach a minimum in the validation loss, which was considered as reached after 10 epochs of no improvements. Therefore, for the synthetic datasets used in the VGG-based architecture (both unsupervised and weakly supervised approaches), training involved 10-50-50 epochs for each training phase, respectively, and for the cue conflict dataset it involved 50 epochs per phase. For the cervical cancer dataset, training involved 10-25-40 epochs per phase in the unsupervised approach and 30-30-50 in the weakly supervised approach.

In all datasets, a batch size of 32 was used, as well as Keras' default initialiser, the Xavier uniform initialiser[6].

[5] https://keras.io/optimizers/
[6] https://keras.io/initializers/

5.2.2 Alfa and Beta

In both VGG- and ResNet-based architectures, α was first fine tuned, keeping β constant at 1. First, a coarse grained search was performed, by testing this subset of values: $[0.5, 0.75, 0.9]$. Then, a finer search was conducted between the range that produced the most reasonable explanations, while maintaining the predictive performance. For all datasets and architectures, these values were closer to 0.9 than to 0.75 (cf. chapter 6). For the weakly supervised approach, since it does not involve a β hyperparameter, tuning stops here.

For the unsupervised approach, after finding an α value that produced reasonable visual explanations compared to what was expected, β was tuned. Similarly, the search started at a broader range, $[0.0, 0.25, 0.5, 0.75, 1.0]$, and was afterwards restricted to a smaller range. For synthetic datasets, this smaller range was $[0.0, 0.25]$, while for the cervical cancer dataset was $[0.9, 1.0]$. Results for these experiments are shown in chapter 6.

5.3 Explainer-Classifier Connection Study

In chapter 4, the explainer's output is fed to every convolution-convolution-pooling stage of the classifier. However, other connection typologies are possible, so a brief study was conducted, in order to decide the best approach moving forward. Therefore, 3 types of connections were made: only to the classifier's last, first and every conv-conv-pool stage.

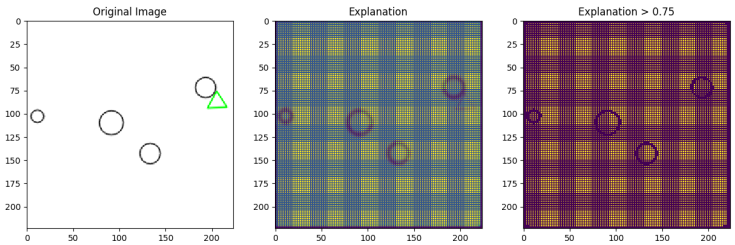
This brief study was conducted on synthetic datasets, with and without colour cues. Figures 5.10 and 5.12 present randomly selected examples of the obtained results for a synthetic dataset with and without colour cues, respectively. All experiments were conducted under the same settings, changing only the explainer-classifier connections, and using $\alpha = 1$ and $\beta = 0$ as the values for the loss hyperparameters. For each row, the original input image is presented, as well as the explainer's outputted explanation and that same explanation after applying a threshold of 0.75. Each row corresponds to a different connection type. Figures 5.11 and 5.13 show the evolution of the classifier's training and validation accuracy per epoch. Each column corresponds to a different connection type.

In the synthetic dataset with colour cues, one is able to verify that connecting the explainer's output only to the last classifier conv-conv-pool stage yielded unsatisfactory explanations (figure 5.10 **(a)**). When the connection is made only to the first classifier conv-conv-pool stage, the explanation becomes better, but some artifacts, mainly vertical stripes, are still present in the image (figure 5.10 **(b)** middle image), especially when compared with the explanations produced by connecting the explainer to every classifier stage (figure 5.10 **(c)** middle image). Similar results were obtained with the synthetic dataset without colour cues: when connecting the explainer to the last classifier stage, the explanation is considered to be absent in the image, because no region is highlighted (figure 5.12 **(a)**); when connecting the explainer to the first classifier stage, some artifacts appear as well, but the polygons are slightly highlighted (figure 5.12 **(b)**); finally, when connecting

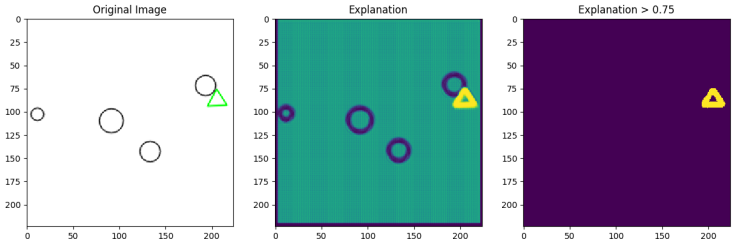
the explainer to every classifier stage, all polygons are highlighted, but the target triangle is clearly distinguishable.

Regarding the evolution of the classifier's accuracies, in the synthetic dataset with colour cues, the classifier stabilises faster when only its last stage is connected to the explainer's output. However, the obtained explanation is so unsatisfactory, that this type of connection was discarded. In fact, this type of connection does not affect the feature extraction learning process, i.e., it is more of a post-processing filtering of the resulting feature maps before the dense layers. Between the accuracies obtained when only the first or every classifier stage is connected to the output of the explainer, the difference is negligible: the classifier stabilises first when only connected to the first stage on both datasets (cf. figures 5.11 and 5.13 **(a)** and **(b)**). Regardless, since this work focuses on the explanations, rather than on classification, and since both classifiers converge to the same final accuracy value, the chosen architecture was the one described in previous sections and depicted in figure 4.2.

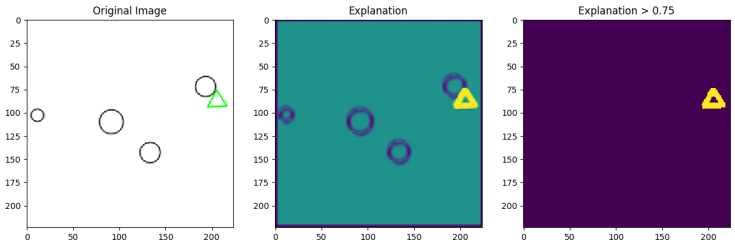
Connecting the explainer's output only to the first classifier conv-conv-pool stage would, in fact, facilitate the process of connecting any given classifier to this architecture and to use pre-trained classifiers as well. However, the explanations are not as good as the ones obtained when connecting the explainer's output to every classifier layer. Also, we conjecture that for deeper classifier architectures the effect of multiplying the explainer's output only by the classifier's first stage versus by every classifier stage, would produce more visible differences between the two approaches than those produced in these experiments, because in the first case the information provided by the explainer would have to go through more layers and its influence would gradually vanish between all the convolutions, which would not happen if at every stage the original unprocessed explanation is again fed to the classifier.



(a)

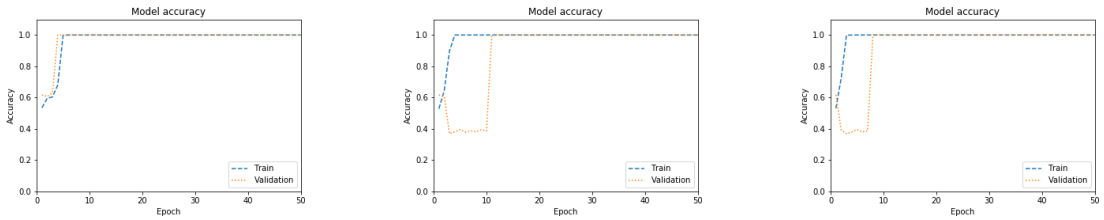


(b)



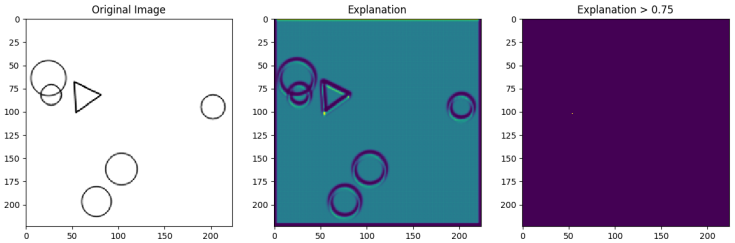
(c)

Figure 5.10: Resulting explanations obtained on a synthetic dataset with colour cues when connecting the explainer to the classifier's **(a)** last, **(b)** first and **(c)** every convolution-convolution-pool stage. For each row the original input image is presented, as well as the explainer's outputted explanation and the same explanation after applying a threshold.

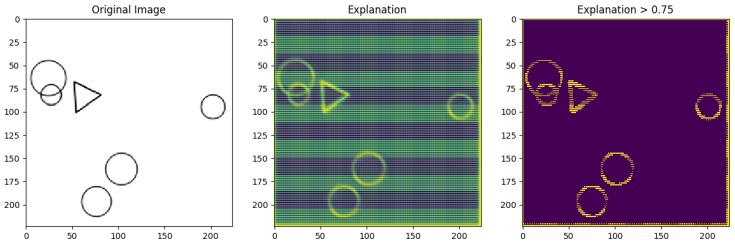


(a) (b) (c)

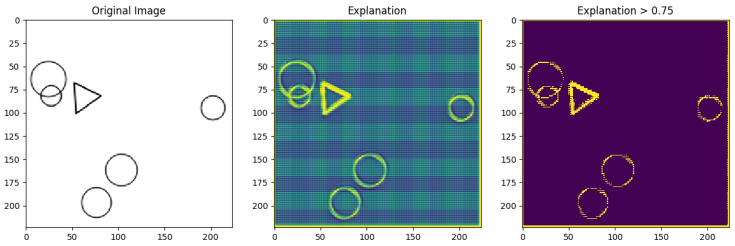
Figure 5.11: Evolution of the classifier's training and validation accuracy obtained on a synthetic dataset with colour cues when connecting the explainer to the classifier's **(a)** last, **(b)** first and **(c)** every convolution-convolution-pool stage.



(a)

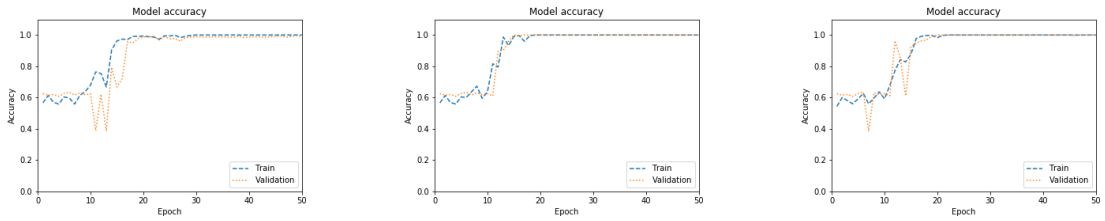


(b)



(c)

Figure 5.12: Resulting explanations obtained on a synthetic dataset without colour cues when connecting the explainer to the classifier's **(a)** last, **(b)** first and **(c)** every convolution-convolution-pool stage. For each row the original input image is presented, as well as the explainer's outputted explanation and the same explanation after applying a threshold.



(a) (b) (c)

Figure 5.13: Evolution of the classifier's training and validation accuracy obtained on a synthetic dataset without colour cues when connecting the explainer to the classifier's **(a)** last, **(b)** first and **(c)** every convolution-convolution-pool stage.

Chapter 6

Results and Discussion

This chapter is dedicated to the obtained results on the unsupervised and weakly supervised approaches. It is further divided into synthetic and real datasets.

Regarding the unsupervised approach on synthetic data, the results obtained without any regularisation on both synthetic datasets are detailed, followed by sections devoted to the study of the influence of the α and β hyperparameters, as well as their recommended values, and ending with a comparison with some of state-of-the-art methods. Then, results of this approach applied to the cue conflict dataset and to the NIH-NCI cervical cancer dataset are presented.

The end of the chapter focuses on the results obtained with the weakly supervised approach on both synthetic and the NIH-NCI cervical cancer datasets.

For all of the following images it is worth noting that the colour code ranges from purple to yellow, where yellow represents higher pixel values, as can be seen in figure 6.1. The results are presented in images containing three columns, where the left column corresponds to the original image, the middle column to the outputted explanation and the right column to the explanation after an absolute threshold of 0.75 is applied.

Images from sections 6.1.1.1 through 6.1.1.3 were part of the validation set, since these sections refer to hyperparameter tuning and effect on the produced explanations. All other images are from testing sets, after the hyperparameter tuning process described in section 5.2.2.

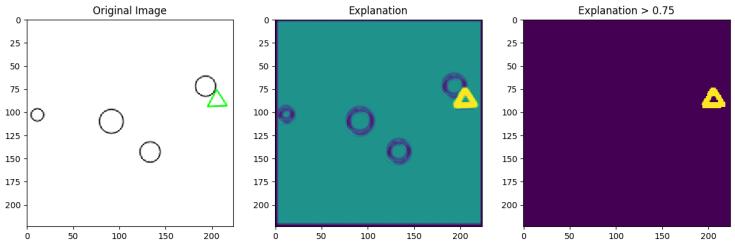
Figure 6.1: Colour map used in the explanation results. Higher pixel values are represented by the colour yellow, while lower pixel values correspond to the colour purple.

6.1 Unsupervised Approach

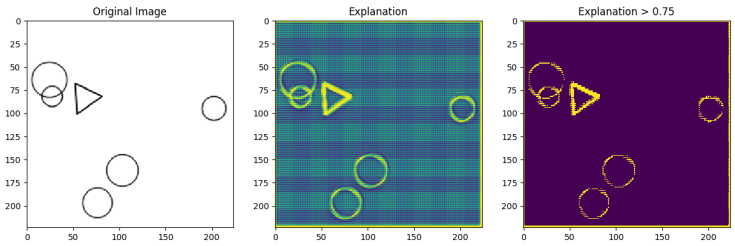
6.1.1 Synthetic Datasets

6.1.1.1 Without Regularisation ($\alpha = 1$)

Figure 6.2 presents examples of the obtained results for positive instances for simple synthetic datasets with and without colour cues. Both images are the result of training the explainer without any kind of regularisation, i.e., with $\alpha = 1$. The value of β is unimportant, since it would be multiplied by $1 - \alpha$, which in this case equals 0. For such simple datasets, it is expected that the explanation focuses on the target polygon, the triangle, rendering the rest of the image as irrelevant for the predicted class. While on the dataset with colour cues **(a)** the resulting explanation consists only of the target polygon, as expected, in the slightly more complex dataset without colour cues **(b)** the explanation becomes degraded, as both triangle and circles are highlighted, which corroborates the need for regularising the explainer's output. As stated in subsection 4.5.1.1, the penalised $l1$ norm is introduced, because it allows for the selection of the relevant parts of the explanation, ensuring that only a small part of the image is in fact the explanation of the classifier's decision, i.e., that the explanation is sparse. Thus, this regularisation ensures that the explanation is not only interpretable, but also complete, as desired.



(a) $\alpha = 1.0 \quad \beta = 0.0$



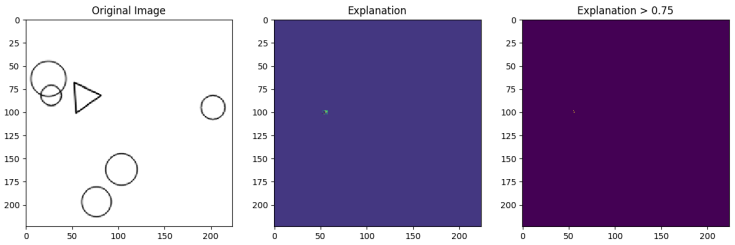
(b) $\alpha = 1.0 \quad \beta = 0.0$

Figure 6.2: Results obtained in an unsupervised scenario without any regularisation of the explainer, $\alpha = 1.0$, on synthetic datasets, with **(a)** and without **(b)** colour cues.

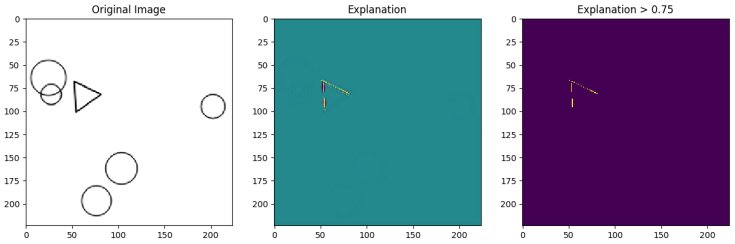
6.1.1.2 With Regularisation: Influence of the α Hyperparameter

The result of the experiments with different regularisation penalties are presented in figures 6.3 and 6.4; these were obtained with $\alpha = [0.5, 0.9, 0.95, 0.99999]$, on the dataset without colour cues, for positive and negative instances, respectively. In every experiment, β was kept constant at 1, so that only the penalised $l1$ norm would take effect, and that it would be possible to better evaluate the influence of the α hyperparameter.

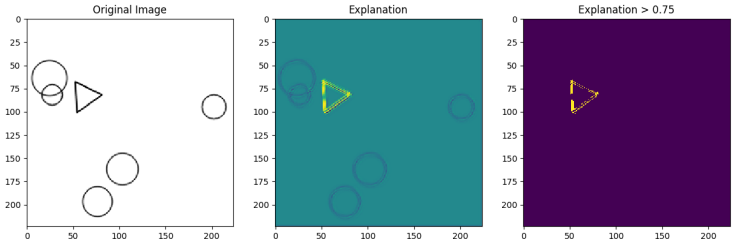
Regarding only the positive examples, figure 6.3, in which a triangle is present, one can see that the more regularisation (from **(d)** to **(a)**) the less pixels are highlighted, as was already foreseen in subsection 4.5.1.1. With $\alpha = 0.5$ no region of the image is considered an explanation, which contradicts the intuition that if there is a triangle in the input image, then its pixels would constitute the explanation for its presence. This stems from the fact that this value of α, more specifically, the value of $1 - \alpha$, is too high, limiting the explanation budget too much. In fact, when this budget increases ($\alpha = 0.9$), i.e., when a greater percentage of the image can be considered an explanation, the target polygon starts being highlighted and even more so when $\alpha = 0.95$. In fact, the more regularisation, the less redundancy: since the explanation budget is tighter, the explainer has to find a way to reduce its explanation to the absolutely essential pixels; this is what happens in figure 6.2 **(b)**, where only two sides of the triangle are highlighted, because when only distinguishing triangles from circles a single straight line is enough to identify the triangle from the circles. Finally, when $\alpha = 0.99999$, the triangle is still highlighted and the circles become even more negatively highlighted. However, increasing α even further, approximates the case without any regularisation (figure 6.2 **(b)**), in which the explanation budget is not constraining the explanations enough, resulting in both triangle and circles highlighted.



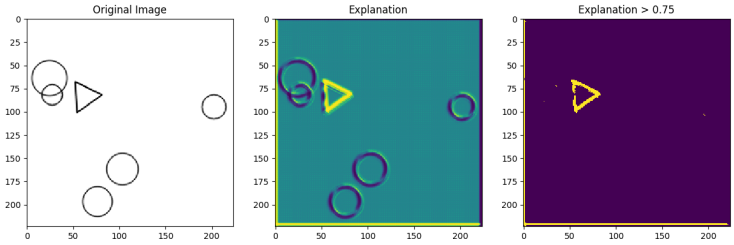
(a) $\alpha = 0.50 \quad \beta = 1.00$



(b) $\alpha = 0.90 \quad \beta = 1.00$



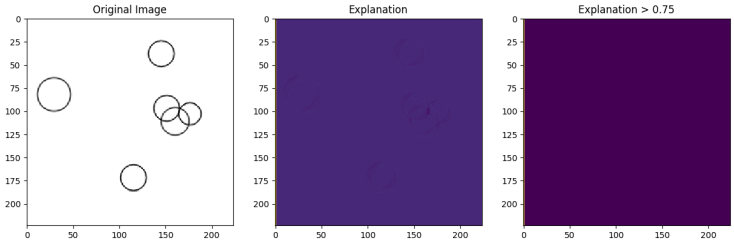
(c) $\alpha = 0.95 \quad \beta = 1.00$



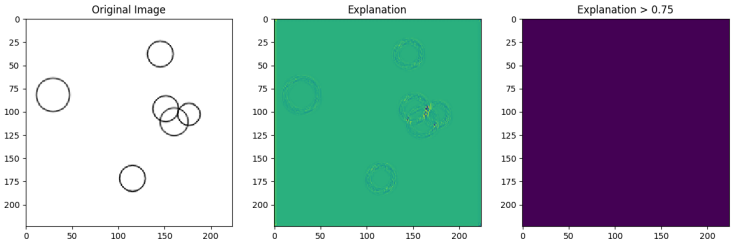
(d) $\alpha = 0.99999 \quad \beta = 1.00$

Figure 6.3: Examples of positive instances obtained in an unsupervised scenario with different α values, $\alpha = [0.5, 0.9, 0.95, 0.99999]$, on a synthetic dataset without colour cues.

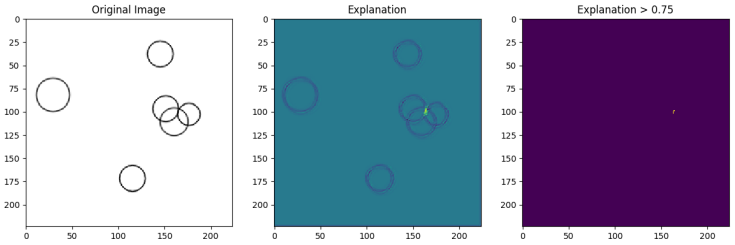
Regarding the negative examples, figure 6.4, as expected, in every case no circles are considered explanations, as they do not explain the absence of the triangle. When $\alpha = 0.5$, similarly to what happens with the positive instance, no region of the image is considered an explanation, because the explanation budget is too tight. When α increases, the circles are more and more negatively highlighted, which means that the explainer is more confident that those regions do not constitute explanations for the absence of the triangle. Also, the whole image excluding the circles is considered an explanation: note that the background colour of the images from the middle columns is not dark purple, but is green and light blue, meaning that its pixel values are located in the middle of the colour map. This aligns with the intuition that, when no triangle is present the whole image except the circles is the explanation for why no triangle exists in it. It also interesting to note that with higher values of α, the intersection of the three circles resembles part of a triangle and, therefore, is also highlighted.



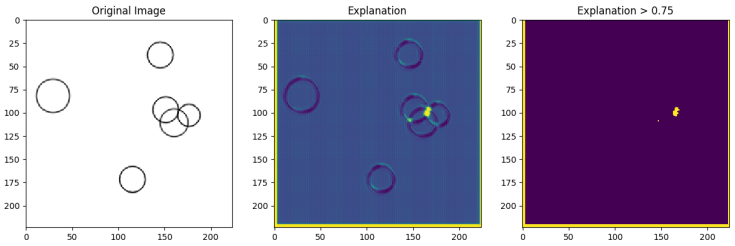
(a) $\alpha = 0.50 \quad \beta = 1.00$



(b) $\alpha = 0.90 \quad \beta = 1.00$



(c) $\alpha = 0.95 \quad \beta = 1.00$



(d) $\alpha = 0.99999 \quad \beta = 1.00$

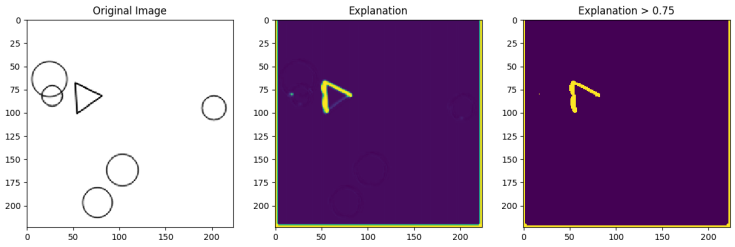
Figure 6.4: Examples of negative instances obtained in an unsupervised scenario with different α values, $\alpha = [0.5, 0.9, 0.95, 0.99999]$, on a synthetic dataset without colour cues.

6.1.1.3 With Regularisation: Influence of the β Hyperparameter

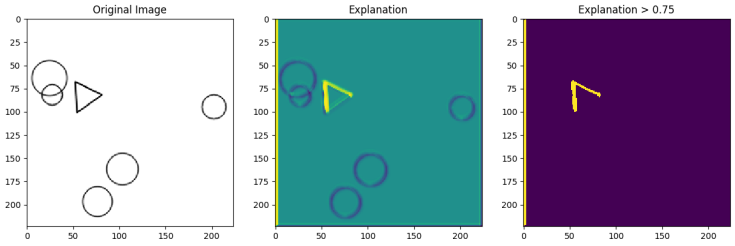
Results of experiments with different β values are presented in figures 6.5 and 6.6. These experiments were conducted with α fixed to 0.9. This α value was chosen to better evaluate the influence of the β hyperparameter, because lower values of α regularise the explainer too much, causing the triangles to "disappear" regardless of the value of β. Higher α values make the differences between experiments with different β values too subtle to be easily visualised.

Regarding positive instances, with lower β values (figures 6.5 **(a)** and **(b)**) the total variation factor dominates over the sparsity constraint, making the triangles visible. With higher β values the sparsity constraint excessively regularises the explanation, causing the triangles not to be highlighted. However, it is interesting to note that with $\beta = 1.0$ (cf. figure 6.3 **(b)**), the triangle is slightly highlighted, which might indicate that for values of β between $]0.5, 1.0[$, the interaction

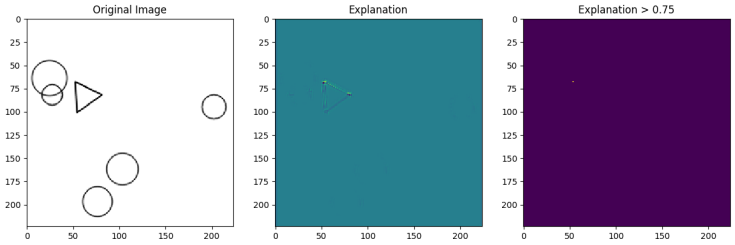
between the sparsity and the contiguity constraints might be prejudicial, causing the triangles not to be highlighted.



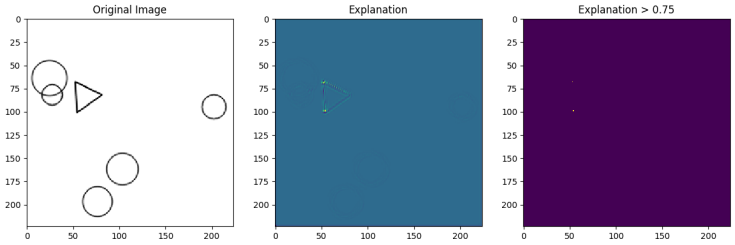
(a) $\alpha = 0.90 \quad \beta = 0.00$



(b) $\alpha = 0.90 \quad \beta = 0.50$



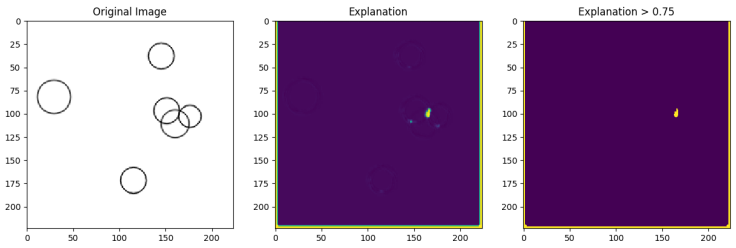
(c) $\alpha = 0.90 \quad \beta = 0.75$



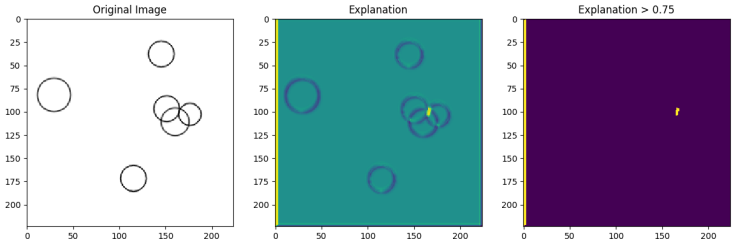
(d) $\alpha = 0.90 \quad \beta = 0.90$

Figure 6.5: Examples of positive instances obtained in an unsupervised scenario with different β values, $\beta = [0.0, 0.25, 0.50, 0.75, 0.9]$ on a synthetic dataset without colour cues.

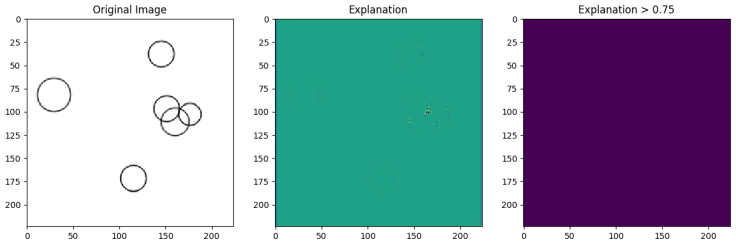
Regarding negative instances (figure 6.6), the same behaviour is verified: for lower β values the intersection between circles is highlighted, while for higher β values it is not. In terms of background, in both positive and negative instances, it is better negatively highlighted by lower β values, even more so if β is closer to 0.5.



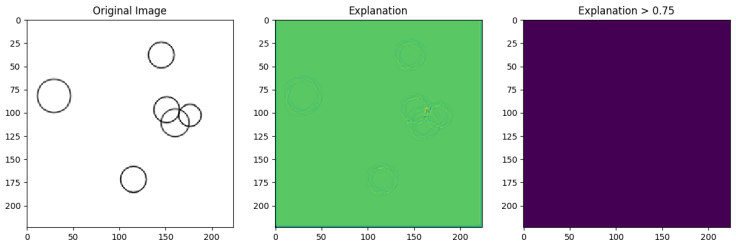
(a) $\alpha = 0.90 \quad \beta = 0.00$



(b) $\alpha = 0.90 \quad \beta = 0.50$



(c) $\alpha = 0.90 \quad \beta = 0.75$

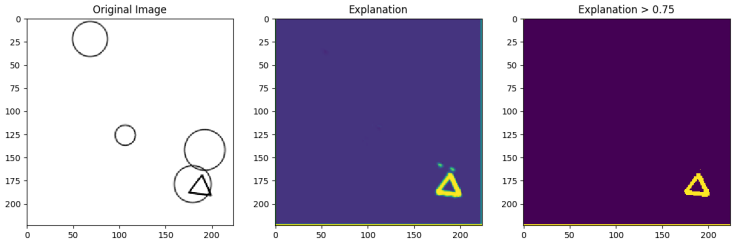


(d) $\alpha = 0.90 \quad \beta = 0.90$

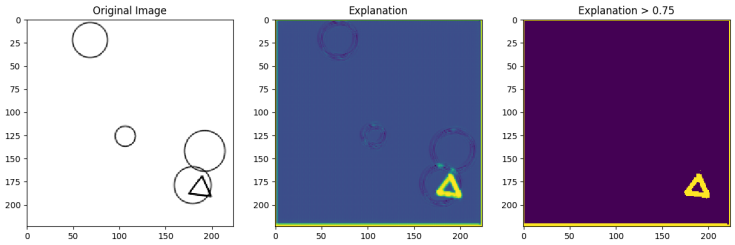
Figure 6.6: Examples of negative instances obtained in an unsupervised scenario with different β values, $\beta = [0.0, 0.25, 0.50, 0.75, 0.9]$ on a synthetic dataset without colour cues.

6.1.1.4 With Regularisation: α and β Recommended Values

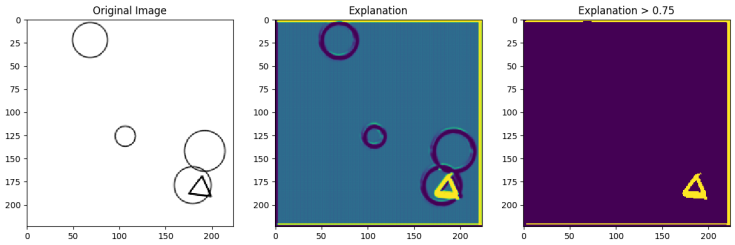
From the analysis conducted in the previous sections, the values of the α and β hyperparameters that produce more reasonable explanations are: $\alpha \in [0.9, 0.99999]$ and $\beta \in [0.1, 0.5]$. Examples of the obtained results for some of the recommended values are presented in figure 6.7. It is worth noting that all of these values lead to the same predictive performance (cf. section 5.3).



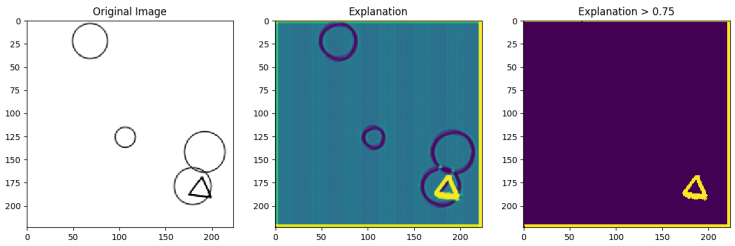
(a) $\alpha = 0.99 \quad \beta = 0.10$



(b) $\alpha = 0.99 \quad \beta = 0.25$



(c) $\alpha = 0.99999 \quad \beta = 0.10$



(d) $\alpha = 0.99999 \quad \beta = 0.25$

Figure 6.7: Examples of positive instances obtained in an unsupervised scenario with recommended α and β values on a synthetic dataset without colour cues.

6.1.1.5 Comparison with State of the Art Methods

A qualitative comparison of the proposed architecture with various methods available in the iNNvestigate toolbox [96] was also made. This toolbox aims to facilitate the comparison of reference implementations of post-model interpretability methods, by providing a common interface and out-of-the-box implementation of various analysis methods. This toolbox was, therefore, used to compare the proposed architecture with methods like SmoothGrad [79], DeconvNet [73], Guided BackProp [99], DTD [101] or LRP [74]. In the proposed architecture, the explainer is the component that produces a visual representation of the reasoning behind the classifier's decisions, just like the analysers available in the iNNvestigate toolbox. As such, these analysis methods are applied only to the classifier, in order to compare only the explanation generators. Since these methods are applied after the model is trained, the classifier was first trained on the simple dataset without colour cues. Then, the various analysis methods are applied to the trained classifier and their generated visual explanations are compared to the ones outputted by the explainer trained in the previous experiments with $\alpha = 0.95$.

The results are shown in figure 6.8. As can be observed, the majority of the methods are unable to produce reasonable explanations for the chosen dataset, for example, highlighting corners of the image, while the proposed architecture is able to only highlight the relevant regions for the classifier's decision (figure 6.8 second row, eleventh column or third row, sixth column). Furthermore, when no target polygon is present (negative instances: first and second rows), the proposed architecture indicates that it is unable to identify said polygon, by highlighting the whole image as the explanation for why no triangle exists in it; and negatively highlighting the circles, as these polygons for sure do not constitute triangles. Conversely, the state of the art methods analysed highlight non-relevant elements when faced with negative instances.

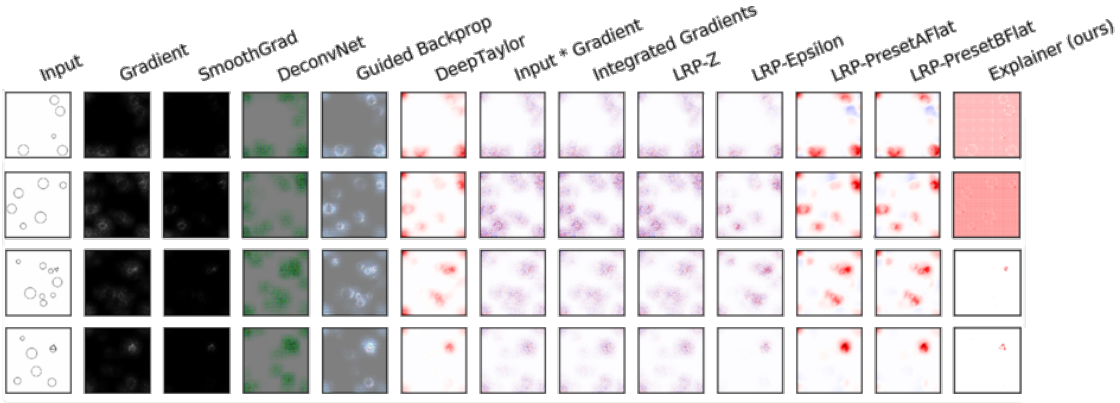
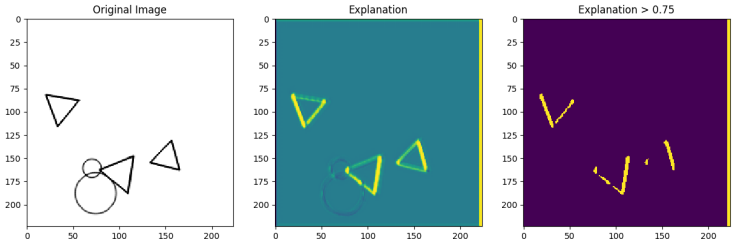


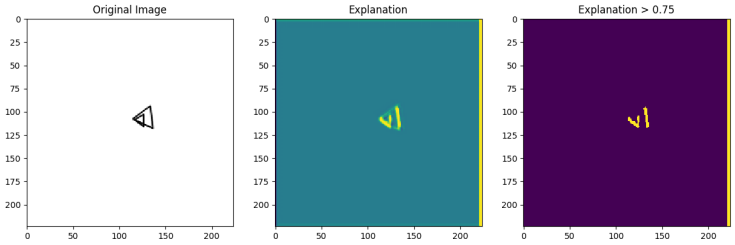
Figure 6.8: Results of the application of some state of the art methods available on the iNNvestigate toolbox [96] to the proposed classifier and comparison with the proposed architecture. The first two rows represent results when a negative instance is given as input, while the last two rows correspond to results with positive instances. The colour map of the right column's images was adjusted to help visualisation, due to the small size of each image and for easier comparison.

6.1.1.6 Other Synthetic Datasets

The unsupervised approach was also tested on another synthetic dataset without colour cues, but with multiple targets. As can be seen in figure 6.9, the proposed architecture is scalable to a task with multiple targets. Furthermore, it is able to classify, and consequently explain, images where no circles are present, a case which was not contemplated in the previous synthetic datasets.



(a) $\alpha = 0.99 \quad \beta = 0.25$



(b) $\alpha = 0.99 \quad \beta = 0.25$

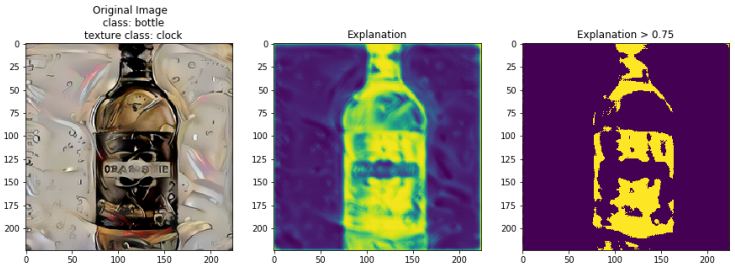
Figure 6.9: Examples of positive instances obtained in an unsupervised scenario with $\alpha = 0.99$ and $\beta = 0.25$ on a synthetic dataset without colour cues and multiple targets.

6.1.2 Real Datasets

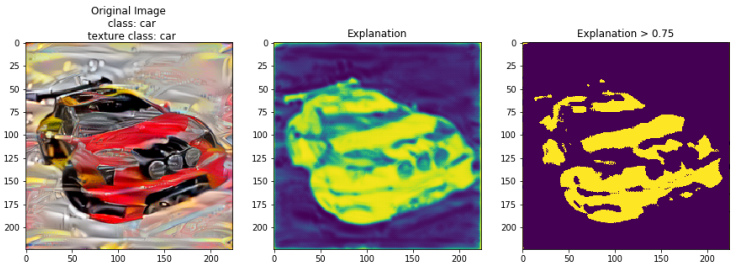
6.1.2.1 Cue Conflict Dataset

This section is dedicated to the results obtained after training the proposed architecture on the cue conflict dataset without any regularisation. Some examples of the obtained explanations are shown in figure 6.10. One can see that the generated explanations are oriented towards semantic components of the objects. For example, for the bottle case the explanation focuses more on the neck of the bottle and on its label. In the car example, the explanation highlights the car's bumper and in the bicycle case, the handles and the seat are highlighted, while in the chair example the chair's legs are highlighted. It is worth noting that although the resulting explanations highlight different semantic components of the objects, they do not appear connected to each other (for example, the handle and the seat of the bicycle). This result hints that improving the quality of these explanations can be made by ensuring that explanations are sparse, i.e., cover a smaller part of the whole image, and also connected. Thus, these preliminary results inspired us to try using the

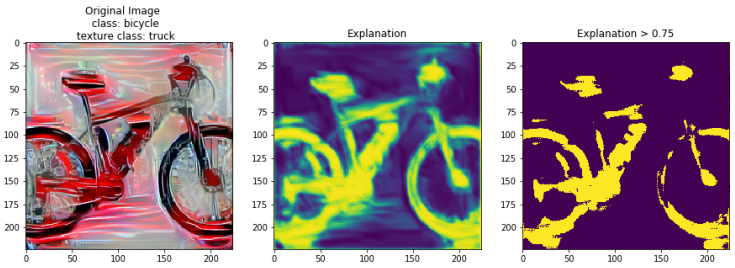
penalised $l1$ norm along with total variation as regularisation of the explainer's training, leading to the final loss presented in section 4.5.1.1.



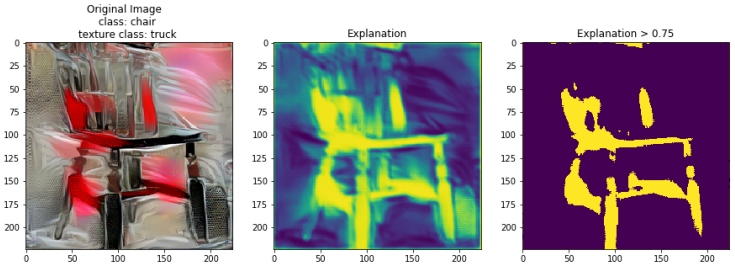
(a)



(b)



(c)



(d)

Figure 6.10: Selected examples of results obtained when training on the cue conflict dataset. The generated explanations are oriented towards semantic components of the objects.

6.1.2.2 NIH-NCI Cervical Cancer Dataset

The proposed architecture, now **using the ResNet50 as classifier instead of the previously used VGG-based classifier**, was also tested on the **NIH-NCI cervical cancer dataset (RQ3-RQ4)**. Before proceeding, a caveat should be introduced: the visual evaluations were performed by lay humans, with the help of an expert[1] in specific cases only. As a thorough evaluation by more experts was not possible during the course of this work, the focus of this analysis is on limit cases (either CIN1 and less, or CIN3 and higher), so that the presence or absence of lesions would be more accentuated and easily identified.

It is also worth noting that in the synthetic cases, **the model reached an accuracy on the test set of 100%**, so no analysis was done on misclassifications. However, in this case, after hyperparameter tuning, **the model achieved 81.32% accuracy on the test set**, so now misclassifications have to be taken into account **(RQ2)**. So, for each instance, the logit probability before the softmax layer is presented.

Figure 6.11 shows the results obtained on examples of the test set, for $\alpha = 0.99$ and $\beta = 0.9$. The first row **(a)** corresponds to a healthy case, which was correctly classified. Most of the cervix is highlighted as the relevant region for this classification. In fact, this aligns with the knowledge gathered that healthy cervixes are planar and pinkish; also, no evident lesions are present.

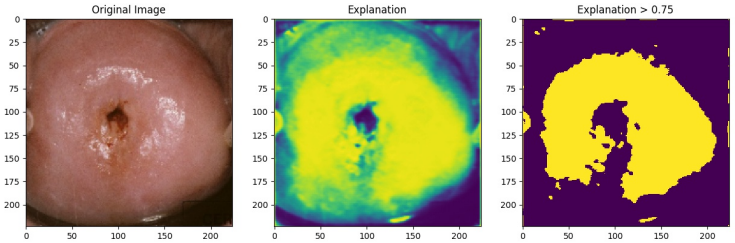
A normal case that was classified as cancerous is presented in figure 6.11 **(b)**. However, the confidence is not very high. When asked, the expert actually classified this as a cancer case as well, but again with many doubts as to the correct classification. This is mainly due to the fact that the colour of the cervix is not very different from other healthy cases, and the morphology of the cervix is not greatly altered, except for the tumescence of the middle-right section of the image. Indeed, the produced explanation points that area as significant for the outputted decision.

Figure 6.11 **(c)** is an example of a cancer case (CIN3) correctly classified, and the explanation seems to corroborate that, by highlighting the majority of the cervix, which might indicate the change in morphology typical of cancer cases. Also, one can observe whitish lesions on the lower-middle part of the cervix, which are highlighted in the explanation as well, and were also pointed out by the expert. In figure 6.11 **(d)** a misclassified cancer case is shown, where the explainer failed to highlight relevant regions such as the irregular contours in the centre of the cervix.

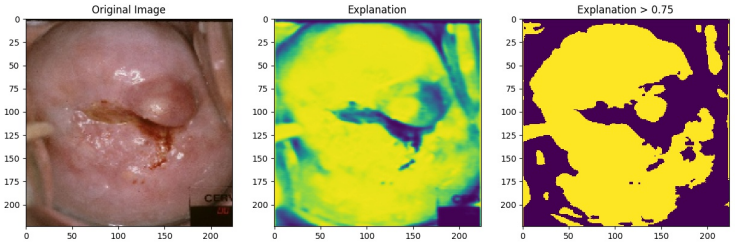
Finally, the last figure **(e)** shows another normal case correctly classified, introduced here mainly for comparison with the weakly supervised approach. Although correctly classified, it seems that the explanation is only focused on reflections present in the original image.

The analysis of these examples allows one to conclude that the explainer focuses on relevant regions for the decision when no misclassification errors occur. When the classification fails, due to the nature of the global loss and the indirect influence of the classification loss on the explainer, the explainer also highlights regions that would not be considered relevant. However, the identification of these regions is also important to better understand why the misclassification occurred and debug the model.

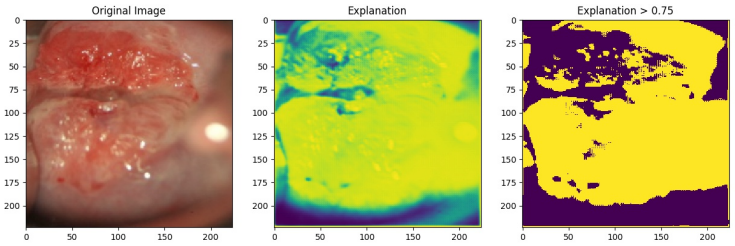
[1] We would like to point out that this person is, in fact, an expert on stomach cancer, but that works in a tissue and tumour bank, so is familiar with cervical cancer as well.



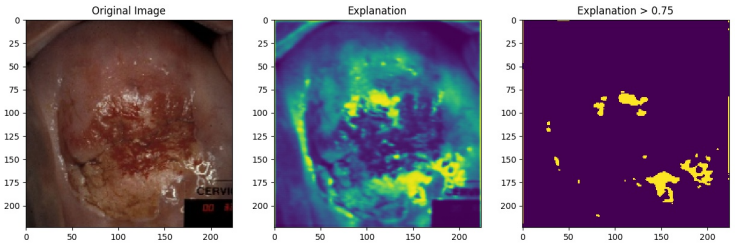
(a) Normal case correctly classified (91.52%).



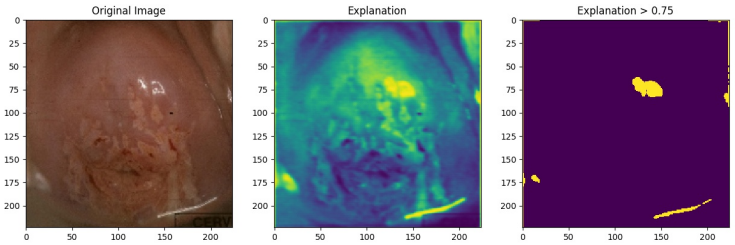
(b) Normal case misclassified (51.08%).



(c) Cancer case correctly classified (55.47%).



(d) Cancer case misclassified (72.62%).



(e) Normal case correctly classified (80.42%).

Figure 6.11: Examples of explanations obtained in an unsupervised scenario with $\alpha = 0.99$ and $\beta = 0.9$ on the NIH-NCI cervical cancer dataset.

6.2 Weakly Supervised Approach

6.2.1 Synthetic Datasets

Figure 6.12 shows an example of a positive instance and the corresponding explanation using the weakly supervised approach with $\alpha = 0.99$. **Similarly to what happened in the unsupervised scenario, the triangle is highlighted, which validates this approach (RQ1).** Comparing this result with the one obtained in the unsupervised scenario with the same α value (cf. figure 6.7 **(a)**), one can observe that in this approach the circles are more negatively highlighted, fruit of the punishment that the weakly supervised loss imposes on non-interest regions.

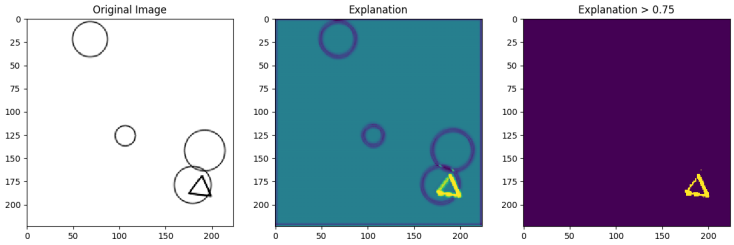


Figure 6.12: Example of the obtained results in a weakly supervised scenario with $\alpha = 0.99$ on a synthetic dataset without colour cues.

Regarding the multiple targets case, the main difference is that here all of the triangles' edges are highlighted, because the weakly supervised loss only punishes regions outside the interest zone, while in the unsupervised case the loss constraints the number of pixels used in an explanation.

On such simple toy datasets the visual differences between the two approaches are not significant (RQ1). As such, and given the fact that generating bounding boxes and their corresponding masks is automatic and does not require any human annotation effort, this approach reduces the hyperparameter tuning time by at least half. This advantage is even more significant on larger datasets and deeper networks, where training takes considerably more time, specially in an architecture with three training phases as this one.

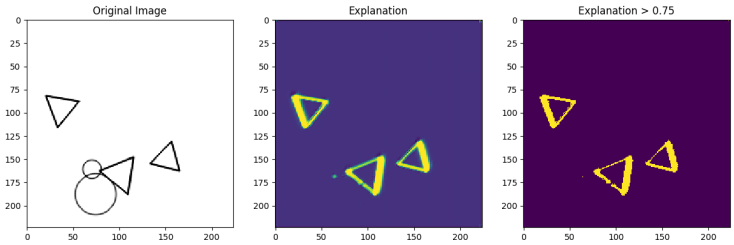


Figure 6.13: Example of the obtained results in a weakly supervised scenario with $\alpha = 0.99$ on a synthetic dataset without colour cues and multiple targets.

6.2.2 Real Datasets

As already mentioned, **in this approach some kind of ground-truth masks from other task are needed (RQ1).** In this case, bounding boxes that delimit the cervix area were used, so no preprocessing was done to the original images, except for a normalisation of the pixel values (c.f. section 5.1.3). After hyperparameter tuning, **the final model reached 81.37% accuracy on the test set (RQ2)**, a marginally higher value than with the unsupervised approach.

Similarly to the unsupervised approach, the first case was correctly classified as healthy (figure 6.14 **(a)**). Despite having highlighted the specular reflections, the model is very confident that it is a healthy cervix, and the whole cervix is highlighted as an indication that no regions have lesions.

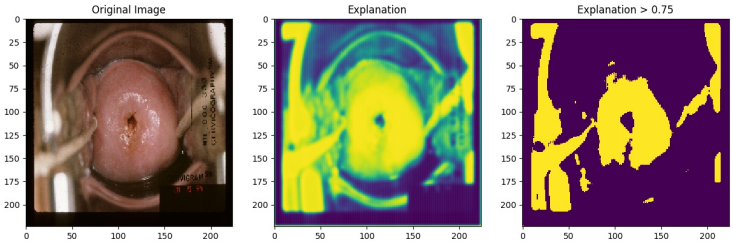
The second case **(b)** is also a normal case correctly classified, but that with the unsupervised approach had been classified as cancerous with low confidence (51.08%). Here, once more the whole cervix is highlighted, except for the bump, which was highlighted in the previous approach. Not considering this bump a relevant region is aligned with the intuition that that region was what mislead the classifier into predicting this as a cancer case. At first glance this bump might seem an indicator of a lesion, but its regular, planar and pinkish nature are consistent with visual features of healthy tissue.

Case **(c)** was misclassified as non cancerous with this approach, while in the unsupervised scenario it was correctly classified. This can be caused by the fact that, although the majority of non-interest regions were punished, the classifier focused on the reflection on the right side of the image. Also, this image has some blurred areas within the interest region, which did not affect the classifier as much in the unsupervised scenario, because the image was cropped and, thereby, the cervix area was amplified and less affected by these blurred regions.

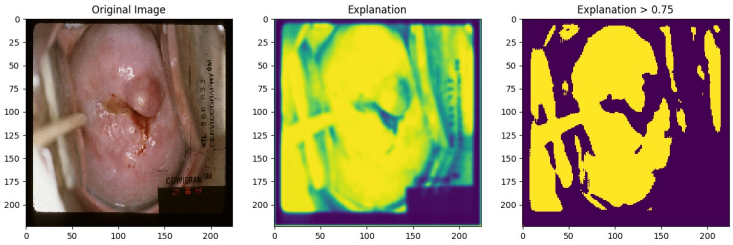
Figure 6.14 **(d)** was correctly classified as cancerous, which did not happen in the other approach. In this case, the produced explanation does not provide very insightful information as to why this decision was made. Similarly to what happened before, very few regions are highlighted as relevant, maybe due to poor lighting conditions and the presence of reflections.

Figure 6.14 **(e)** is an example in which a normal case was misclassified as cancerous. In fact, the classifier focused too much on specular reflections, mainly on the left side of the image.

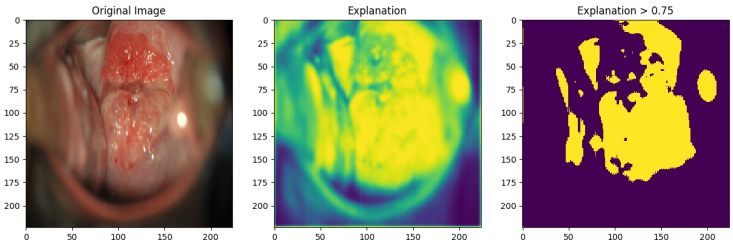
Once more, the analysis of these examples leads us to conclude than when no misclassification occurs, **the explanations highlight regions of the image consistent with *a priori* knowledge of visual features associated with cancerous cases**, such as morphology, colour and contour changes of the cervix tissue. Therefore, **the produced explanations are indicative of injured areas, but not exact topographical areas of lesions (RQ4).**



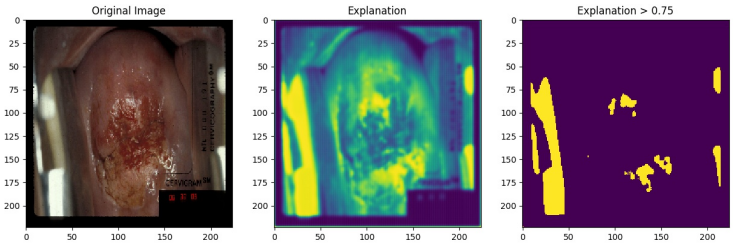
(a) Normal case correctly classified (99.78%).



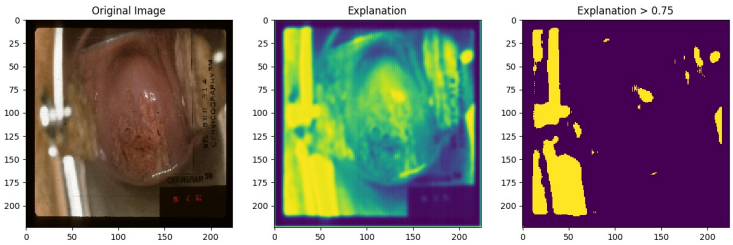
(b) Normal case correctly classified (99.91%).



(c) Cancer case misclassified (86.09%).



(d) Cancer case correctly classified (90.52%).



(e) Normal case misclassified (86.87%).

Figure 6.14: Examples of explanations obtained in a weakly supervised scenario with $\alpha = 0.88$ on the NIH-NCI cervical cancer dataset.

Chapter 7

Conclusions and Future Work

The outstanding predictive performance of DL models has recently led to the deployment of such systems in the real world, raising a myriad of new technical and legal problems and requirements that need to be tackled, especially when deploying these systems in highly regulated areas such as medicine. One of the most important and challenging problems nowadays is known as XAI: methods that allow AI systems to be understandable and trusted by humans. Ideally, one would need data labelled with the decisions (classification, regression, detection or segmentation labels, for example) and with the explanations for those decisions, in order to supervise the process of generating explanations. As this is not feasible, mainly due to the inherent difficulties of the subjective nature of this area, these systems need to be able to generate these explanations in an unsupervised fashion.

To try and begin to tackle this problem, and also to explore the category of in-model methods, which is less developed compared to the post-model class, in this work we proposed an in-model joint approach to produce decisions and explanations using CNNs without the need to supervise the explanation component. The developed architecture is composed of a classifier and an explainer, thus producing class predictions and visual explanations. This novel architecture, along with its custom training process and loss function, allows for the classifier to be trained with the produced explanations, thus only focusing on relevant parts of the input image. Similarly, the explainer is trained together with the classifier, thus learning to find the relevant image regions that contribute to the correct classification of said image and, therefore, explain such prediction. This architecture can be used with any classifier architecture, provided the corresponding connections are made and the classifier is retrained within this new scenario. In fact, in this work two classifier architectures were used, a ResNet50 classifier for the NIH-NCI cervical cancer dataset and a VGG-16-based architecture for the remaining datasets.

The explainer is trained without direct supervision, but with indirect supervision through the global loss function that includes a classification and an explanation component. Two strategies were proposed for this last component: an unsupervised and a weakly supervised approach. The first one introduces regularisation terms, the $l1$ penalised norm and total variation, that impose desired characteristics on the explanations, such as sparsity and contiguity, respectively. The weakly

supervised approach works by punishing the explanations outside of interest regions defined by masks created from annotations of other tasks, namely detection or segmentation tasks. The first approach does not require any further annotations besides the class labels, but it involves tuning two hyperparameters, while the weakly supervised strategy reduces this tuning effort by at least half.

To test the proposed architecture a synthetic data generation tool with automatic annotation was also developed, in addition to the real datasets used. This tool allows producing images containing simple polygons, in a way that is both versatile and highly customisable. These simple images do not require expert knowledge to compare their desired explanations with the ones produced by the proposed architecture, which greatly speeds up the evaluation process and helps debug the model.

The results obtained show that this architecture is able to produce visual explanations, as well as decisions, while not degrading classification accuracy. The explanations are not only interpretable, but also complete, i.e., are able to describe the system's internals accurately. When compared to state of the art methods, the proposed joint approach produces better visual explanations on synthetic data that, in fact, only highlight relevant image regions for the outputted predictions. Especially with regards to negative instances, this method considers the whole image, except the non-target objects, an explanation, which aligns with the intuition that, in the absence of a target object, the whole image is the explanation for why no target is present.

Furthermore, the proposed architecture was tested on a real dataset, the NIH-NCI cervical cancer dataset. The obtained results show that, when no misclassification occurs, the explanations highlight regions of the image consistent with *a priori* knowledge of visual features associated with cancerous cases, such as morphology, colour and contour changes of the cervix tissue. However, the identification of the highlighted regions in misclassification examples is also important to better understand why the misclassification occurred and debug the model. In fact, it was observed that the architecture is sensible to lighting conditions and reflections that might appear in the images due to their collection process, which provided important insights into the need to better preprocess the dataset.

In conclusion, in this work we proposed an **in-model joint approach** that, for the reasons stated above, is able to produce meaningful and reasonable explanations for its generated predictions, without sacrificing predictive performance. Furthermore, this architecture can be **used with different classifier architectures on real data and does not require further annotations**, being capable of producing explanations in an **unsupervised or weakly supervised** way. **Therefore, we conclude that all the proposed research questions were addressed and proven feasible**.

However, research in this area is still in its infancy and the proposed approach can be improved and further explored. Examples of some future research directions include:

- Application to more complex classification problems including multi-class classification.
- Comparison between the usage of finer grained masks, such as segmentation masks, versus detection masks in the weakly supervised approach.

- More robust evaluation of the produced explanations, especially in the cervical cancer application, by consulting more experts.
- Evaluation of the proposed architecture in relation to possible transformations of the input data.
- Inclusion of more preprocessing steps, mainly in the cervical cancer case, such as the removal of specular reflections.
- Combination of the proposed unsupervised and weakly supervised approaches.
- Design of a loss function that incorporates characteristics at the population level, such as the promotion of explanation clusters similar to the decision clusters, instead of only instance level aspects.
- Extension of the architecture to include multimodal inputs, such as additional tabular information included in the NIH-NCI cervical cancer dataset (age, HPV status, etc.).
- Extension of the architecture to also produce textual explanations.

Appendix A

Extra Files and Submitted Papers

A.1 Synthetic Data Generation Tool Configurable Parameters

- **folder**='train': directory where the dataset files are to be stored/imported from
- **config_file**=None: JSON configuration file where parameters reside if not None. If None, parameters are passed as class constructor arguments
- **nr_images**=100: number of generated images
- **polygon**=[-1, None]: target polygon used to generate annotations given by its parameters p and q (if p equals -1, corresponding to a circle, then q is irrelevant)
- **outside_polygon**=None: polygon to place around target polygon
- **background_colour**=255: background image colour in RGB
- **img_height**=224: image height
- **img_width**=224: image width
- **nr_channels**=3: number of colour channels
- **nr_shapes**=20: number of polygons per generated image
- **nr_tries**=100: number of tries before the algorithm gives up trying to fit the polygon inside the image
- **rad_min**=224/32: minimum possible radius for the polygon's outer circumference
- **rad_max**=224/16: maximum possible radius for the polygon's outer circumference
- **overlap**=False: overlap between polygons of the same image if True, no overlap between every two polygons if False
- **occlusion**=False: occlusion of polygons on image borders if True, no occlusion if False

- **rotation**=True: random rotation of polygons if True, no rotation if False
- **noise**=False: Gaussian noise addition to image if True
- **min_nr_vertices**=3: minimum number of vertices
- **max_nr_vertices**=13: maximum number of vertices
- **min_nr_shapes**=1: minimum number of polygons per image
- **max_nr_shapes**=20: maximum number of polygons per image
- **simplified**=False: simplified version of the dataset (only triangles and circles)
- **no_circles**=False: do not draw circles (if simplified mode is set, then this parameter will be ignored)
- **poly_colour**=False: colour for the target polygon in RGB
- **start_index**=0: start index for image naming (useful when one wants to add more images to an existing dataset)

A.2 XML Annotation File

As depicted in figure A.1, each annotation file includes the folder and file where the image is located, as well as the image's width, height and depth. It also contains the target polygon's p and q values, followed by the bounding box coordinates (xmin, ymin, xmax, ymax) of every instance of the target polygon, as well as the number of target polygons present in the image.

```
<annotation>
  <folder>simplified_no_colour</folder>
  <filename>0.png</filename>
  <size>
    <width>224</width>
    <height>224</height>
    <depth>3</depth>
  </size>
  <polygon>
    <p>3.0</p>
    <q>1.0</q>
  </polygon>
  <bndbox0>
    <xmin>20</xmin>
    <ymin>149</ymin>
    <xmax>38</xmax>
    <ymax>167</ymax>
  </bndbox0>
  <exists>1</exists>
</annotation>
```

Figure A.1: Example XML annotation file in Pascal VOC format.

A.3 IbPRIA2019 Accepted Paper (Honourable Mention Winner)

Towards a Joint Approach to Produce Decisions and Explanations Using CNNs *

Isabel Rio-Torto[1(✉)], Kelwin Fernandes[3], and Luís F. Teixeira[1,2]

[1] Faculdade de Engenharia da Universidade do Porto, Porto, Portugal
[2] INESC TEC, Porto, Portugal
[3] NILG.AI, Porto, Portugal
{icrtto}@gmail.com, {kelwin}@nilg.ai, {luisft}@fe.up.pt

Abstract. Convolutional Neural Networks, as well as other deep learning methods, have shown remarkable performance on tasks like classification and detection. However, these models largely remain black-boxes. With the widespread use of such networks in real-world scenarios and with the growing demand of the right to explanation, especially in highly-regulated areas like medicine and criminal justice, generating accurate predictions is no longer enough. Machine learning models have to be explainable, i.e., understandable to humans, which entails being able to present the reasons behind their decisions. While most of the literature focuses on post-model methods, we propose an in-model CNN architecture, composed by an explainer and a classifier. The model is trained end-to-end, with the classifier taking as input not only images from the dataset but also the explainer's resulting explanation, thus allowing for the classifier to focus on the relevant areas of such explanation. We also developed a synthetic dataset generation framework, that allows for automatic annotation and creation of easy-to-understand images that do not require the knowledge of an expert to be explained. Promising results were obtained, especially when using L1 regularisation, validating the potential of the proposed architecture and further encouraging research to improve the proposed architecture's explainability and performance.

Keywords: Explainable AI · Explainability · Interpretability · Deep Learning · Convolutional Neural Networks

1 Introduction

Deep learning changed the machine learning paradigm in recent years, significantly improving performance on tasks like classification, sometimes even outperforming humans. Due to the achieved outstanding predictive capability, deep learning methods have since been employed in tackling other problems, such as detection and segmentation, surpassing the performance of classical machine learning models also on these tasks.

* This work was partially funded by NILG.AI.

Despite this overwhelming dominance, deep learning models, and in particular convolutional neural networks (CNNs), are still considered black-box models, i.e., models whose reasons for the outputted decisions cannot be understood at a human-level. However, with the growing ubiquitousness of deep learning systems, especially in highly regulated areas such as medicine, criminal justice or financial markets [6], an increasing need for these models to output explanations in addition to decisions is arising. Moreover, the new General Data Protection Regulation (GDPR) includes policies on the right to explanation [5], thus increasing this need for explainable deep learning systems that can operate within this legal framework.

The research community has rapidly taken an interest in this topic, proposing several methods that try to meet this explainability requirement. Nevertheless, the field is still lacking a unified formal definition or possible evaluation metrics. The terms explainability and interpretability are often used interchangeably in the literature. In this work, we adopt the definition proposed by *L.H. Gilpin et al.* [4]. The authors loosely define interpretability as the process of understanding the model's internals and describing them in a way that is understandable to humans, while explainability goes beyond that. Briefly, an explainable model is one that can summarise the reasons for its behaviour or the causes of its decisions. In fact, for a model to be explainable, it needs to be interpretable, but also complete, i.e., to describe the system's internals accurately. As such, explainable models are interpretable by default, while the reverse may not be true. Therefore, a good explanation should be able to balance the interpretability-completeness trade-off, because the more accurate an explanation, the less interpretable it is to humans; for example, an entirely complete explanation of a neural network would consist of all the operations, parameters and hyperparameters of such network, rendering it uninterpretable. Conversely, the most interpretable description is often incomplete.

The majority of the literature focuses only on interpretability, and more specifically on post-model or post-hoc interpretability methods, i.e. methods that are applied after the model is trained. Examples range from proxy methods that approximate the original network model [9] to methods that output visual cues representing what the network is focusing on to make its decisions, such as Sensitivity Analysis and Saliency Maps [8], SmoothGrad [10], DeConvNet [17] or Layer-Wise Relevance Propagation [2].

While a few works focus on in-model approaches, in which interpretability is taken into account while building the model, these are for the most part application oriented. Some work has been developed trying to make predictions based only on patches of the input images, which limits the interpretability of the classifier. Although such in-model methods exist for models such as CNNs [11,12], these are still not considered intrinsically interpretable, making this category still dominated by classic methods like decision trees.

In this work, we propose a preliminary end-to-end in-model approach, based on an explainer+classifier architecture. This architecture outputs not only a class label, but also a visual explanation of such decision. The classifier takes

as input an image, as well as the output of the explainer, i.e. it is trained using the explanation. Therefore, the classifier focuses on the regions the explainer deemed relevant and does not take into account regions where the explanation for that class is not present. This approach aligns with the intuition that, when explaining a decision, for example, whether or not image X is an image of a car, humans tend to distinguish what is an object and what is not, and then proceed to look for patterns in the region where the object is in order to classify it correctly. Conversely, sometimes humans cannot tell if an object belongs to some class, but can tell which regions of the image do not contain said class.

We also propose a synthetic dataset generation framework, allowing for automatic image generation and annotation. The generated images consist of simple polygons, therefore easily explainable by humans, which allows for a qualitative and quantitative evaluation of the produced explanations without the need of expert knowledge, necessary in most real-world datasets.

2 Methodology

2.1 Proposed Architecture

We propose a model consisting of an explainer and a classifier, as depicted in Figure 1. Figure 2 depicts a detailed diagram of the proposed architecture, which is a concretisation of the aforementioned model. The explainer (top row) outputs an image, which we call the explanation, with the same spatial dimensions as the input image. It is composed of a downsampling and an upsampling path. The downsampling path is a simple convolution-convolution-pooling scheme. The upsampling path follows a convolution-convolution-deconvolution scheme, where the first convolution operation is applied to the sum of the previous layer's output with the corresponding convolutional layer in the downsampling path. These connections allow for the successive layer to learn a more precise output. Also, batch normalisation is applied to the last layer of the explainer.

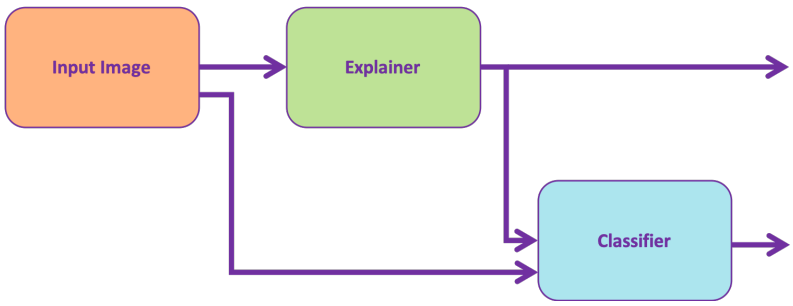


Fig. 1: Block diagram of the proposed explainer+classifier model.

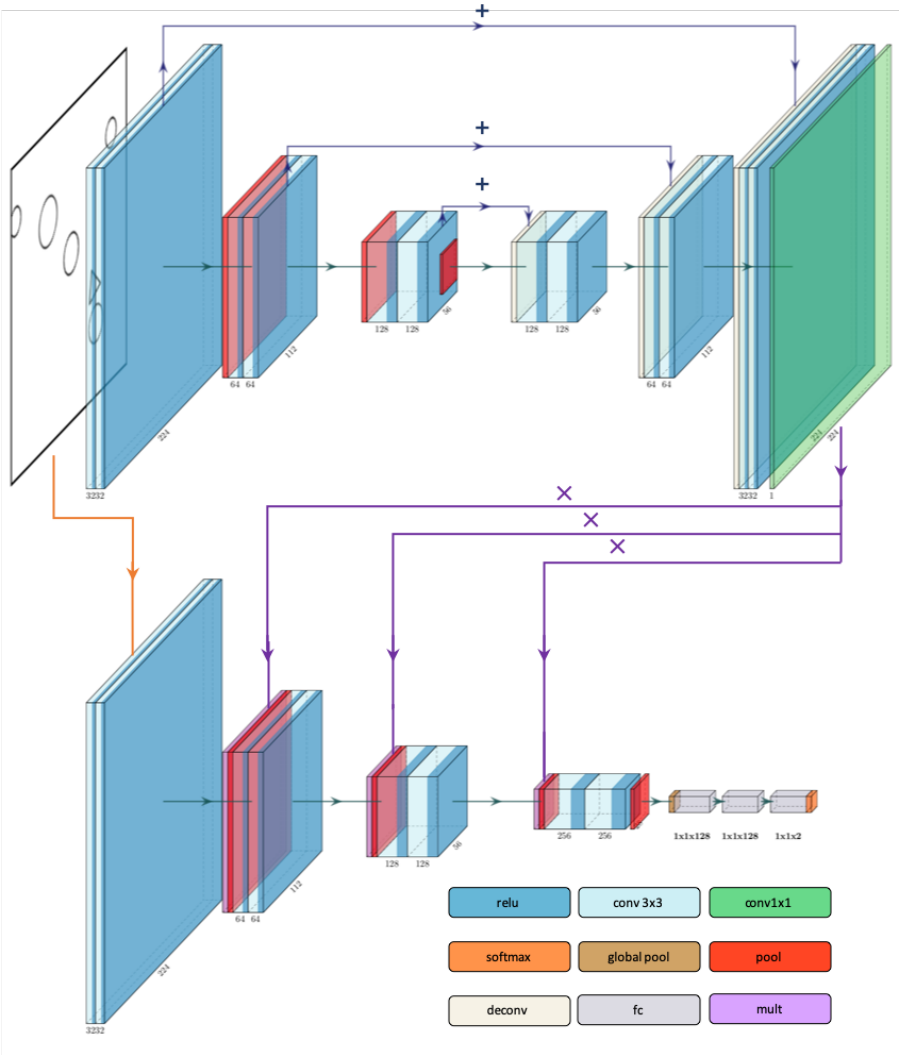


Fig. 2: Diagram of the proposed architecture. It is composed by an explainer (top row) and a classifier (bottom row). The sums correspond to the simple addition of the outputs of the respective layers. The multiplications involve the concatenation and resizing of the explainer's output before computing the element-wise multiplication with the involved classifier layer. This architecture outputs a class label, as well as an explanation for that decision.

The classifier (bottom row) is inspired by the VGG architecture [13], having 4 consecutive convolution-convolution-pool stages, ending with 2 fully connected layers, followed by a softmax layer. However, an important modification to the original VGG is made: each pooling layer takes as input the multiplication of the output of its preceding layer with the output of the explainer. This way, the classifier is trained using the outputted explanation, allowing for it to focus on the relevant parts of the input image and to discard regions where the explanation for the class being predicted is not present.

In both classifier and explainer, 3x3 kernels are used in the convolutional layers. All pooling operations resort to max pooling, downsampling by a factor of 2. The number of filters starts at 32 for the first stage and afterwards increases as a power of 2 according to its stage level.

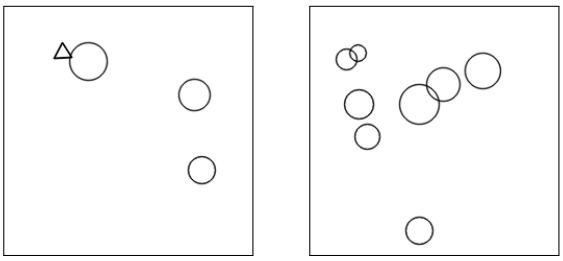
Training involves three steps. First, only the classifier is trained, taking a "white image" as explanation, meaning that the initial explanation is the whole image. Then the explainer is trained with the classifier frozen, outputting an explanation that highlights relevant areas to the classification task. Finally, the whole architecture is fine-tuned end to end.

While the explainer is trained unsupervised, the classifier is trained using the categorical cross-entropy loss. The Keras Adadelta default optimizer was used, which employs an adaptive learning rate based on a moving window of gradient updates [16].

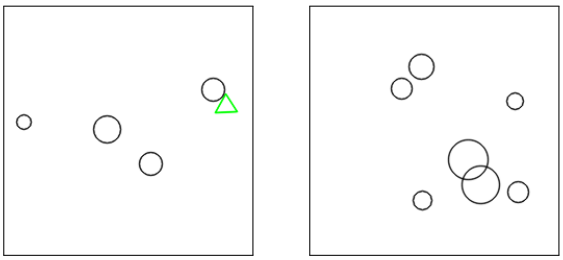
2.2 Synthetic Dataset

For the experimental assessment of the proposed architecture, a synthetic dataset generation framework was developed. The use of a synthetic dataset entails numerous advantages, such as:

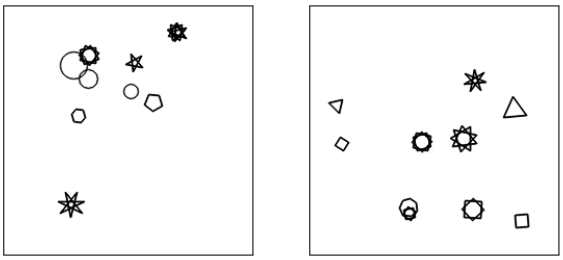
- the ability to generate as many images as one needs
- the possibility of defining the number of instances for each class
- automatic annotation for different problems ranging from classification to detection
- definition of custom-made characteristics like overlap, occlusion, object type, object colour, image dimensions, etc



(a) Images from a simple dataset without colour cues



(b) Images from a simple dataset with colour cues



(c) Images from a complex dataset without colour cues

Fig. 3: Example images of 3 generated datasets. For each row, the left column illustrates an example of the positive class, while the right column illustrates the negative class. On the first two rows the target polygon is a triangle, while on the third row it is a 5 pointed star.

This framework generates images consisting of several polygons, such as triangles, circles and stars. Examples of such images can be observed in Figure 3. Moreover, for each image, an XML annotation file in PASCAL VOC format is created, containing information on how many target polygons exist in the image and their respective bounding boxes. The developed code is available at `https://github.com/icrto/xML`.

2.3 Experiments

The synthetic dataset used in the experiments consists of 1000 224x224 RGB images, each containing a triangle (target polygon) and a variable number of circles. For each image it is only taken into account the presence or absence of the target polygon, thus making these experiments binary classification tasks. The positive class has 618 instances, while the remaining 382 instances belong to the negative class. The data was split into a 75-25 training-validation partition. Each of the three training phases involved 50 epochs with 50 steps each, with a batch size of 8. The classifier was evaluated in terms of its accuracy and the explainer was qualitatively evaluated, by means of human visual inspection.

All experiments were conducted on the Google Collaboratory environment and involved applying to the explainer different L1 regularisation factors, ranging

from 10^{-8} to 10^{-4}. Without regularisation, one can obtain a degraded solution, in which everything is considered an explanation. Therefore, L1 regularisation is employed, so that only a small part of the whole image constitutes an explanation.

A qualitative and quantitative comparison of the proposed architecture with various methods available in the iNNvestigate toolbox [1] is also made. This toolbox aims to facilitate the comparison of reference implementations of post-model interpretability methods, by providing a common interface and out-of-the-box implementation of various analysis methods. The toolbox is, then, used to compare the proposed architecture with methods like SmoothGrad [14], DeconvNet [17], Guided Backprop [15], Deep Taylor Decomposition [7] or Layer-Wise Relevance Propagation (LRP) [2]. In the proposed architecture, the explainer is the component that produces a visual representation of the reasoning behind the classifier's decisions, just like the analysers available in the iNNvestigate toolbox. As such, these analysis methods are applied only to the classifier of the proposed architecture, in order to compare only the explanation generators, i.e. the proposed architecture's explainer and the different analysis methods. Since these methods are applied after the model is trained, we started by training the classifier on the simple dataset without colour cues. Then, the various analysis methods are applied to the trained classifier and their generated visual explanations are compared to the ones outputted by the explainer trained in the previous experiments with 10^{-6} L1 regularisation factor. Furthermore, the classifier's accuracy with and without explainer are also compared. The obtained results are described in section 3.

2.4 Experiments on real datasets

Experiments were also conducted on a real dataset, available at `https://github.com/rgeirhos/texture-vs-shape`. This dataset was created in the context of the work developed by *Geirhos et al* [3], where the authors validate that Imagenet-trained CNNs are biased towards texture. In order to validate this hypothesis, the authors propose a cue conflict experiment in which style transfer is employed, introducing texture in the Imagenet images. This dataset contains 16 classes, with 80 images each. The proposed architecture was trained on this dataset without any regularisation. Results for this experiment are shown in section 3.

3 Results and Discussion

For all of the following images it is worth noting that the colour code ranges from purple to yellow, where yellow represents higher pixel values. The left column corresponds to the original image, the middle column to the outputted explanation and the right column to the explanation after an absolute threshold of 0.75 is applied.

Figure 4 constitutes examples of the obtained results for simple datasets with and without a target polygon of different colour. Both images are the result of training without any kind of regularisation. For such simple datasets, it is expected that the explanation focuses on the target polygon, rendering the rest of the image as irrelevant for the predicted class.

While on the dataset with colour cues the resulting explanation consists only of the target polygon, as expected, in the slightly more complex dataset without colour cues the whole image is considered an explanation, which corroborates the need for regularising the explainer output. As stated in section 2.3, we use an L1 regularisation factor, because it allows for the selection of the relevant parts of the explanation, ensuring that only a small part of the image is in fact the explanation of the classifier's decision. Thus, this regularisation ensures that the explanation is not only interpretable, but also complete, as desired.

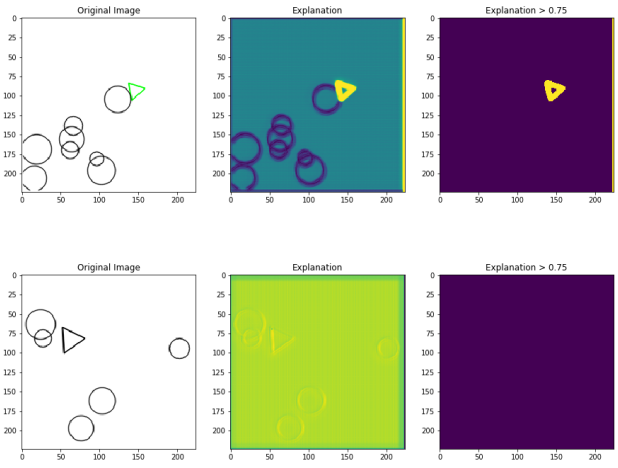


Fig. 4: Positive instance and respective explanation. These results were obtained without any kind of regularisation of the explainer's output while training on a simple dataset without (top) and with (bottom) colour cues.

Figure 5 is the result of the experiments with different regularisation factors, namely 10^{-8}, 10^{-6} and 10^{-4}, on the dataset without colour cues. With a factor of 10^{-8}, not only the target polygon is considered relevant, as well as the circles, which may imply that such a small regularisation is still not enough to limit the relevant parts of the explanation. In fact, increasing L1 to 10^{-6}, produces much better results, with the target polygon clearly highlighted. Finally, increasing L1 a bit further, to 10^{-4}, proved to be too much regularisation, causing the explanation to "disappear".

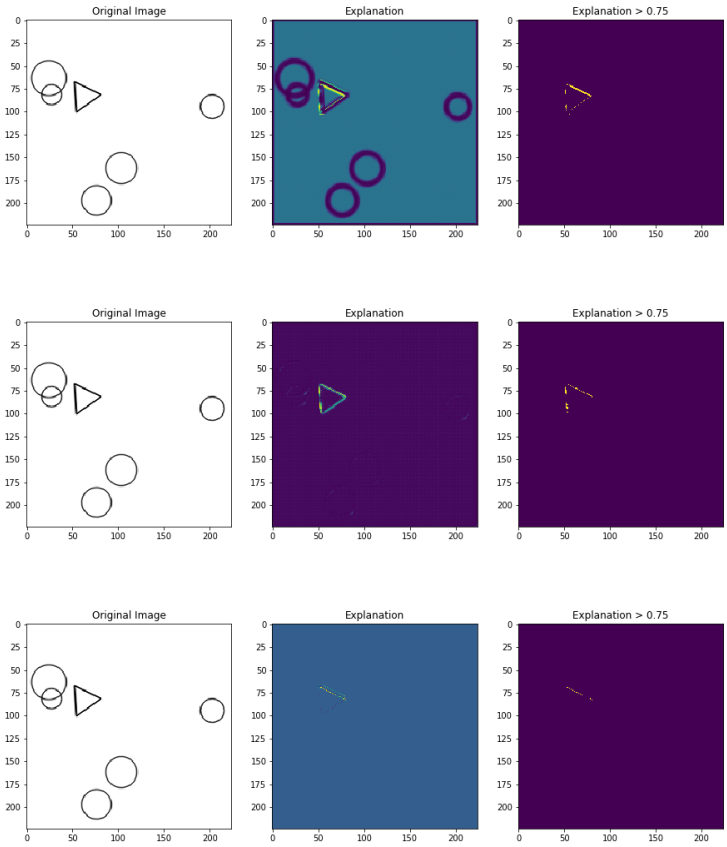


Fig. 5: Positive instance and respective explanation. These results were obtained with 10^{-8} (top), 10^{-6} (middle) and 10^{-4} (bottom) L1 regularisation of the explainer's output while training on a simple dataset without colour cues.

Moreover, the proposed architecture was compared to several other methods available in the iNNvestigate framework [1]. As can be seen in figure 6, the majority of the methods are unable to produce reasonable explanations for the chosen dataset, highlighting corners of the image, for example, while the proposed architecture is able to only highlight the relevant regions for the classifier's decision (see figure 5 middle). Furthermore, training only the classifier, as was done when applying the iNNvestigate toolbox's analysis methods, yields accuracies close to 62%, while the accuracy of the proposed architecture reaches 100%, as illustrated in Figure 7. For this dataset, the proposed network not only produces explanations alongside with predictions, as well as improves accuracy, by forcing the classifier to focus only on relevant parts of the image.

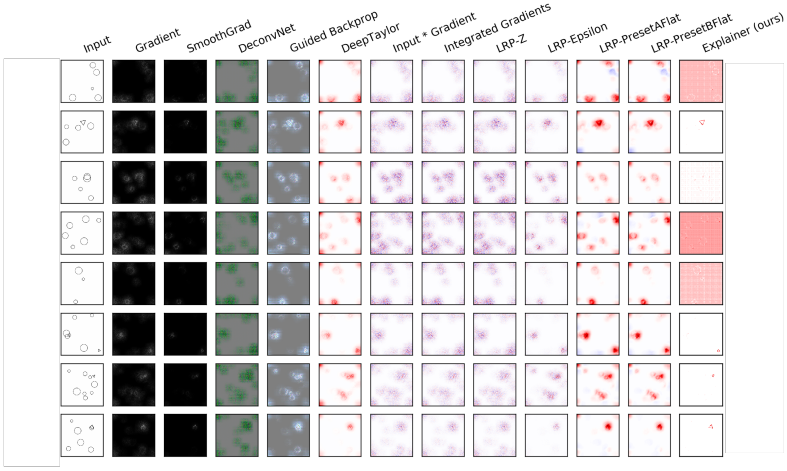


Fig. 6: Results of the application of 10 analysis methods available on the iNNvestigate toolbox [1] to the proposed classifier and comparison with the proposed end-to-end architecture. The color map of the right column's images was adjusted to help visualization due to the small size of each image and for easier comparison.

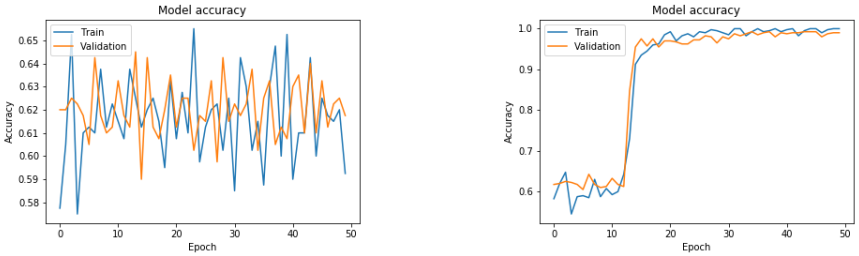


Fig. 7: Evolution of classifier accuracy per training epoch for the same classifier trained alone (left) and within the proposed explainer-classifier architecture (right).

Finally, Figure 8 depicts the obtained results of the experiment on the cue conflict dataset. One can see that the generated explanations are oriented towards semantic components of the objects. For example, for the bottle case the explanations focus more on the neck of the bottle and on its label. In the car example, the explanation highlights the car's bumper and in the bicycle case, the handles and the seat are highlighted, while in the chair example the chair's legs are highlighted. It is worth noting that although the resulting explanations highlight different semantic components of the objects, they do not appear connected to each other (for example, the handle and the seat of the bicycle). This

result hints that improving the quality of these explanations can be made by ensuring that explanations are sparse, i.e., cover a smaller part of the whole image, and also connected.

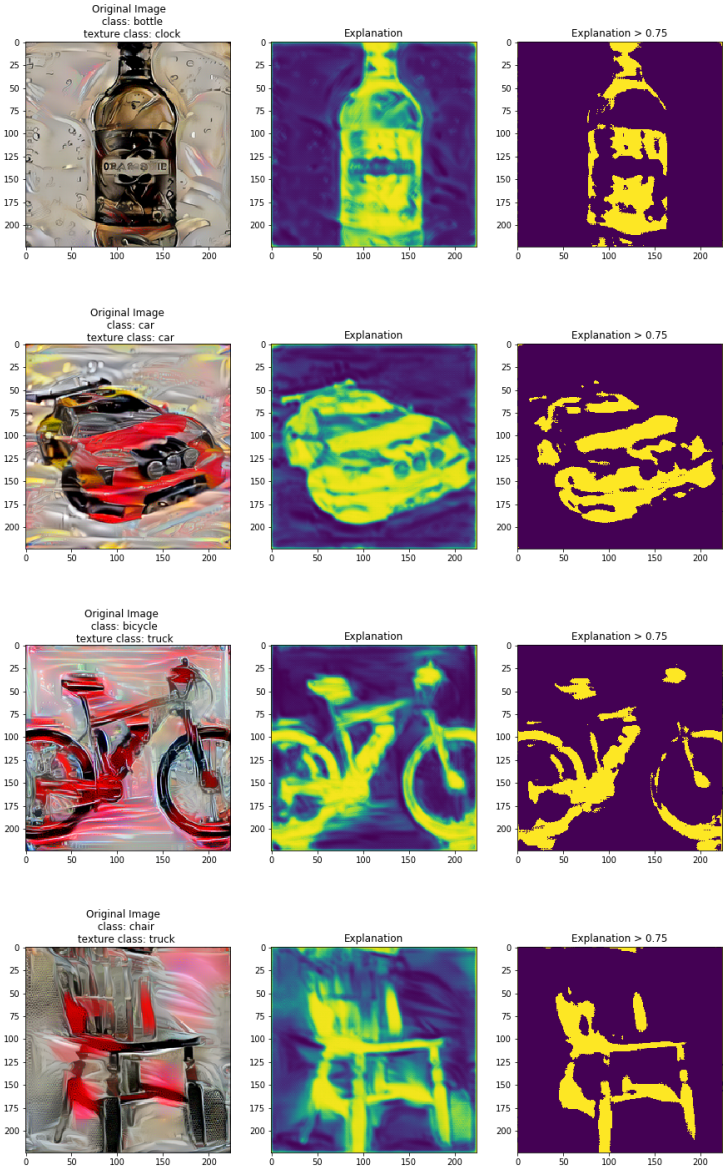


Fig. 8: Results of the cue conflict experiment.

4 Conclusion

We propose a preliminary in-model joint approach to produce decisions and explanations using CNNs, capable of producing not only interpretable explanations, but also complete ones, i.e., explanations that are able to describe the system's internals accurately. We also developed a synthetic dataset generation framework with automatic annotation.

The proposed architecture was tested with a simple generated synthetic dataset, for which explanations are intuitive and do not need to employ expert knowledge. Results show the potential of the proposed architecture, especially when compared to existing methods and when adding L1 regularisation. These also hint at the need for regularisation in order to better balance the interpretability-completeness trade off. As such, future research will study the effect of adding total variation regularisation as a way of making explanations sparse. Also, we will explore the possible advantages of supervising the explanations, as well as develop a proper annotation scheme and evaluation metrics for such task.

References

1. Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K.T., Montavon, G., Samek, W., Müller, K.R., Dähne, S., Kindermans, P.J.: iNNvestigate neural networks! (2018)
2. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS One (2015). https://doi.org/10.1371/journal.pone.0130140
3. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness (nov 2018), http://arxiv.org/abs/1811.12231
4. Gilpin, L.H., Bau, D., Yuan, B.Z., Bajwa, A., Specter, M., Kagal, L.: Explaining explanations: An overview of interpretability of machine learning. Proc. - 2018 IEEE 5th Int. Conf. Data Sci. Adv. Anal. DSAA 2018 pp. 80–89 (2019). https://doi.org/10.1109/DSAA.2018.00018
5. Goodman, B., Flaxman, S.: European Union regulations on algorithmic decision-making and a "right to explanation" (jun 2016). https://doi.org/10.1609/aimag.v38i3.2741, https://arxiv.org/abs/1606.08813
6. Lipton, Z.C.: The Mythos of Model Interpretability (2016). https://doi.org/10.1145/3233231
7. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recognit. (2017). https://doi.org/10.1016/j.patcog.2016.11.008
8. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digit. Signal Process. **73**, 1–15 (feb 2018). https://doi.org/10.1016/J.DSP.2017.10.011, https://www.sciencedirect.com/science/article/pii/S1051200417302385
9. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why Should I Trust You?": Explaining the Predictions of Any Classifier (feb 2016), http://arxiv.org/abs/1602.04938

10. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.r.: Evaluating the Visualization of What a Deep Neural Network Has Learned. IEEE Trans. Neural Networks Learn. Syst. **28**(11), 2660–2673 (2017)
11. Silva, W., Fernandes, K., Cardoso, J.S.: How to produce complementary explanations using an ensemble model. In: 2019 International Joint Conference on Neural Networks (IJCNN) (2019)
12. Silva, W., Fernandes, K., Cardoso, M.J., Cardoso, J.S.: Towards Complementary Explanations Using Deep Neural Networks (2018). https://doi.org/10.1007/978-3-030-02628-8$_1$5
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
14. Smilkov, D., Thorat, N., Kim, B., Vi, F.: SmoothGrad : removing noise by adding noise (2017)
15. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for Simplicity: The All Convolutional Net (dec 2014), `https://arxiv.org/abs/1412.6806`
16. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
17. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) (2014). https://doi.org/10.1007/978-3-319-10590-1$_5$3

References

[1] Michael Van Lent, William Fisher, and Michael Mancuso. An explainable artificial intelligence system for small-unit tactical behavior. In *Proceedings of the national conference on artificial intelligence*, pages 900–907. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[2] David Gunning. Explainable artificial intelligence (xai) - Program Update. *Defense Advanced Research Projects Agency (DARPA)*, 2, Nov 2017.

[3] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *DeepLearning*. MIT Press, 2016. www.deeplearningbook.org, [Accessed: 20th September 2019].

[5] Dhairya Parikh. Learning paradigms in machine learning. https://medium.com/datadriveninvestor/learning-paradigms-in-machine-learning-146ebf8b5943, Jul 2018. [Accessed: 20th September 2019].

[6] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017. doi:10.1093/nsr/nwx106.

[7] Fahad Lateef and Yassine Ruichek. Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, pages 321–348. doi:10.1016/j.neucom.2019.02.003.

[8] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for Interpretable Machine Learning. arXiv:1808.00033.

[9] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943. doi:10.1007/BF02478259.

[10] Ana Neves, Ignacio Gonzalez, John Leander, and Raid Karoumi. A new approach to damage detection in bridges using machine learning. pages 73–84, 01 2018. doi:10.1007/978-3-319-67443-8_5.

[11] Jayesh Bapu Ahire. The artificial neural networks handbook: Part 4. https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e, Nov 2018. [Accessed: 20th September 2019].

[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[13] Marvin Minsky and Seymour Papert. Perceptron: an introduction to computational geometry. *The MIT Press, Cambridge, expanded edition*, 19(88):2, 1969.

[14] Favio Vázquez. A "weird" introduction to deep learning. https://towardsdatascience.com/a-weird-introduction-to-deep-learning-7828803693b0, Aug 2018. [Accessed: 20th September 2019].

[15] Arden Dertat. Applied deep learning - part 1: Artificial neural networks. https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6, Oct 2017. [Accessed: 20th September 2019].

[16] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. arXiv:1412.6980.

[18] Matthew D Zeiler. Adadelta: an adaptive learning rate method, 2012. arXiv:1212.5701.

[19] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate, 2019. arXiv:1902.09843.

[20] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012. arXiv:1207.0580.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015. URL: http://dl.acm.org/citation.cfm?id=3045118.3045167.

[22] David H Hubel. Single unit activity in striate cortex of unrestrained cats. *The Journal of physiology*, 147(2):226–238, 1959.

[23] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[24] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[25] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[26] Convolutional neural network. https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html, [Accessed: 20th September 2019.

[27] Convolutional neural networks (cnns / convnets). http://cs231n.github.io/convolutional-networks/, [Accessed: 20th September 2019].

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2014. arXiv:1409.1556.

[31] Vgg16 - convolutional network for classification and detection, Nov 2018. https://neurohive.io/en/popular-networks/vgg16/, [Accessed: 20th September 2019].

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] Siddharth Das. Cnn architectures: Lenet, alexnet, vgg, googlenet, resnet and more ...., Jul 2018. https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5, [Accessed: 20th September 2019].

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[36] 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.

[37] Wilson Silva, Kelwin Fernandes, Maria João Cardoso, and Jaime S. Cardoso. Towards complementary explanations using deep neural networks. In *Understanding and Interpreting Machine Learning in Medical Image Computing Applications - First International Workshops MLCN 2018, DLF 2018, and iMIMIC 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16-20, 2018, Proceedings*, pages 133–140, 2018.

[38] Artur Andrzejak, Felix Langner, and Silvestre Zabala. Interpretable models from distributed data via merging of decision trees. In *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 1–9. IEEE, 2013.

[39] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 623–631, New York, NY, USA, 2013. ACM. doi:10.1145/2487575.2487579.

[40] Zachary C. Lipton. The Mythos of Model Interpretability. jun 2016. arXiv:1606.03490.

[41] Finale Doshi-Velez and Been Kim. Towards A Rigorous Science of Interpretable Machine Learning. 2017. arXiv:1702.08608.

[42] Been Kim and Finale Doshi-Velez. Interpretable machine learning: The fuss, the concrete and the questions. *ICML Tutorial on Interpretable Machine Learning*, 2017.

[43] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, feb 2018.

[44] Leila Arras, Franziska Horn, Grégoire Montavon, Klaus Robert Müller, and Wojciech Samek. "What is relevant in a text document?": An interpretable machine learning approach. *PLoS ONE*, 12(8):1–23, aug 2017. arXiv:1612.07843.

[45] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89, Oct 2018. doi:10.1109/DSAA.2018.00018.

[46] Bernease Herman. The promise and peril of human evaluation for model interpretability. 2017. arXiv:1711.07414.

[47] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. Machine Learning Interpretability: A Survey on Methods and Metrics. *Electronics*, (8):832. doi:10.3390/electronics8080832.

[48] Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1 – 38, 2019. doi:10.1016/j.artint.2018.07.007.

[49] Tae Wam Kim. Explainable artificial intelligence (xai), the goodness criteria and the grasp-ability test. 2018. arXiv:1810.09598.

[50] Finale Doshi-Velez, Mason Kortz, Ryan Budish, Christopher Bavitz, Samuel J. Gershman, David O'Brien, Stuart Shieber, Jim Waldo, David Weinberger, and Alexandra Wood. Accountability of AI Under the Law: The Role of Explanation. 2017. arXiv:1711.01134.

[51] Raymond S Nickerson. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of general psychology*, 2(2):175–220, 1998.

[52] Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. Examples are not Enough, Learn to Criticize! Criticism for Interpretability. In *Advances in Neural Information Processing Systems*, pages 2280–2288, 2016.

[53] Bryce Goodman and Seth Flaxman. European Union regulations on algorithmic decision-making and a "right to explanation". jun 2016. doi:10.1609/aimag.v38i3.2741.

[54] Andreas Holzinger, Chris Biemann, Constantinos S. Pattichis, and Douglas B. Kell. What do we need to build explainable AI systems for the medical domain? (Ml):1–28, 2017. arXiv:1712.09923.

[55] Julia Angwin, Jeff Larson, Lauren Kirchner, and Surya Mattu. Machine Bias, 2016. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, [Accessed: 20th September 2019].

[56] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 07-12-June-2015:427–436, 2015. doi:10.1109/CVPR.2015.7298640.

[57] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. pages 1–10, 2013. arXiv:1312.6199.

[58] Seyed Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, 2017-January:86–94, 2017. doi:10.1109/CVPR.2017.17.

[59] Nicolas Papernot, Patrick Mcdaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *Proc. - 2016 IEEE Eur. Symp. Secur. Privacy, EURO S P 2016*, pages 372–387, 2016. doi:10.1109/EuroSP.2016.36.

[60] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning Attack on Neural Networks. (February), 2018. doi:10.14722/ndss.2018.23291.

[61] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models. aug 2017. arXiv:1708.08296.

[62] Matt McFarland. Uber self-driving car kills pedestrian in Arizona - Roadshow. www.cnet.com/roadshow/news/uber-autonomous-car-crash-arizona-tempe/, [Accessed: 20th September 2019].

[63] Jacob Haspiel, Na Du, Jill Meyerson, Lionel P. Robert, Dawn Tilbury, X. Jessie Yang, and Anuj K. Pradhan. Explanations and Expectations: Trust Building in Automated Vehicles. *ACM/IEEE Int. Conf. Human-Robot Interact.*, (Dc):119–120, 2018. doi:10.1145/3173386.3177057.

[64] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. pages 1–9, 2016. arXiv:1604.07316.

[65] Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. Interpretable Deep Models for ICU Outcome Prediction. *AMIA ... Annu. Symp. proceedings. AMIA Symp.*, 2016:371–380, 2016.

[66] Angela Stark. FDA permits marketing of artificial intelligence-based device to detect certain diabetes-related eye problems. www.fda.gov/news-events/press-announcements/fda-permits-marketing-artificial-intelligence-based-device-detect-certain-diabetes-related-eye, [Accessed: 20th September 2019], 2018.

[67] Alun Preece, Dan Harborne, Dave Braines, Richard Tomsett, and Supriyo Chakraborty. Stakeholders in Explainable AI. 2018. arXiv:1810.00184.

[68] Mauricio Reyes. Interpretability methodologies for machine learning in medical imaging. https://github.com/visum-summerschool/visum-2019/blob/master/july11_interpretability/Invited_VISUM_Interpretability_Reyes_2019_smallsize.pdf, July 2019. [Accessed: 20th September 2019], VISUM Summer School 2019.

[69] Christoph Molnar. *Interpretable Machine Learning*. 2018. https://christophm.github.io/interpretable-ml-book/, [Accessed: 20th September 2019].

[70] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, pages 369–370. Springer-Verlag New York, second edition, 2009.

[71] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. feb 2016. arXiv:1602.04938.

[72] Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, pages 457–473. Springer, 2016.

[73] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pages 818–833. Springer International Publishing, 2014.

[74] 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*, 2015. arXiv:1606.04155.

[75] 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*, pages 2921–2929, 2016.

[76] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153, International Convention Centre, Sydney, Australia, 2017. PMLR.

[77] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, Oct 2017. doi:10.1109/ICCV.2017.74.

[78] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. 2017. arXiv:1703.01365.

[79] Daniel Smilkov, Nikhil Thorat, Been Kim, and Fernanda Vi. SmoothGrad : removing noise by adding noise. 2017. arXiv:1706.03825v1.

[80] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[81] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pages 2673–2682, 2018.

[82] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. 2017. arXiv:1703.03717.

[83] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable Convolutional Neural Networks. oct 2017. arXiv:1710.00935.

[84] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[85] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[86] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[87] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gpdr. *Harv. JL & Tech.*, 31:841, 2017.

[88] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. 2018. arXiv:1802.05668.

[89] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-robert Müller. Evaluating the Visualization of What a Deep Neural Network Has Learned. *IEEE Trans. Neural Networks Learn. Syst.*, 28(11):2660–2673, 2017.

[90] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. A unified view of gradient-based attribution methods for deep neural networks. *NIPS Workshop on Interpreting, Explaining and Visualizing Deep Learning*, 11 2017.

[91] Daniel W Apley. Visualizing the effects of predictor variables in black box supervised learning models. 2016. arXiv:1612.08468.

[92] Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015.

[93] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-Agnostic Interpretability of Machine Learning. 2016. arXiv:1606.05386.

[94] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, 2018.

[95] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013. arXiv:1312.6034.

[96] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. iNNvestigate neural networks! 2018. arXiv:1808.04260.

[97] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[98] Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: PatternNet and PatternAttribution. 2017. arXiv:1705.05598.

[99] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. 2014. arXiv:1412.6806.

[100] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. 2016. arXiv:1605.01713.

[101] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65(C):211–222, May 2017.

[102] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org, 2017.

[103] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. 2017. arXiv:1711.06104.

[104] Wojciech Samek. Interpreting Deep Neural Networks and their Predictions, 2018. www.iphome.hhi.de/samek/pdf/CERN2018.pdf, [Accessed: 20th September 2019].

[105] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9505–9515, 2018.

[106] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (Un)reliability of saliency methods. 2017. arXiv:1711.00867.

[107] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[108] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. 2014. arXiv:1412.6856.

[109] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016.

[110] Ruth Fong and Andrea Vedaldi. Net2Vec: Quantifying and Explaining how Concepts are Encoded by Filters in Deep Neural Networks. 2018. arXiv:1801.03454.

[111] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. 2018. arXiv:1803.03635.

[112] Song-chun Zhang Quan-shi and Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, jan 2018.

[113] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[114] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 842–850, 2015.

[115] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances in Neural Information Processing Systems 29*, pages 289–297. Curran Associates, Inc., 2016.

[116] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[117] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016.

[118] Isabel Rio-Torto, Kelwin Fernandes, and Luís F. Teixeira. Towards a Joint Approach to Produce Decisions and Explanations Using CNNs. In *9th Iberian Conference on Pattern Recognition and Image Analysis*, pages 3–15. Springer International Publishing, 2019.

[119] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1721–1730, New York, NY, USA, 2015. ACM. doi:10.1145/2783258.2788613.

[120] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[121] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, Nov 1987. doi:10.1007/BF00058680.

[122] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

[123] Wilson Silva, Kelwin Fernandes, and Jaime S. Cardoso. How to produce complementary explanations using an ensemble model. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.

[124] Sérgio Pereira, Raphael Meier, Richard McKinley, Roland Wiest, Victor Alves, Carlos A. Silva, and Mauricio Reyes. Enhancing interpretability of automatically extracted machine learning features: application to a RBM-Random Forest system on brain lesion segmentation. *Med. Image Anal.*, 44:228–244, 2018. doi:10.1016/j.media.2017.12.009.

[125] Pedro M. Ferreira, Filipe Marques, Jaime S. Cardoso, and Ana Rebelo. Physiological Inspired Deep Neural Networks for Emotion Recognition. *IEEE Access*, 2018.

[126] Robert Geirhos, Carlos R. Medina Temme, Jonas Rauber, Heiko H. Schütt, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks. In *Advances in Neural Information Processing Systems*, number NeurIPS 2018, pages 7538–7550, 2018.

[127] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. nov 2018. arXiv:1811.12231.

[128] Rolando Herrero, Mark H Schiffman, Concepción Bratti, Allan Hildesheim, Ileana Balmaceda, Mark E Sherman, Mitchell Greenberg, Fernando Cárdenas, Víctor Gómez, Kay Helgesen, et al. Design and methods of a population-based natural history study of cervical neoplasia in a rural province of costa rica: the guanacaste project. *Revista Panamericana de Salud Pública*, 1:362–375, 1997.

[129] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[130] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[131] Robert P Kauffman, Stephen J Griffin, Jon D Lund, and Paul E Tullar. Current recommendations for cervical cancer screening: do they render the annual pelvic examination obsolete? *Medical Principles and Practice*, 22(4):313–322, 2013.

[132] Dezhao Song, Edward Kim, Xiaolei Huang, Joseph Patruno, Héctor Muñoz-Avila, Jeff Heflin, L Rodney Long, and Sameer Antani. Multimodal entity coreference for cervical dysplasia diagnosis. *IEEE transactions on medical imaging*, 34(1):229–245, 2014.