FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Explainability in Medical Imaging
Student:	Bc. Adam Skluzáček
Supervisor:	Ing. Jakub Žitný
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

Machine learning applications in healthcare are increasingly dependent on the possibility to interpret or explain the underlying models. This is to strengthen confidence when classifying diseases or segmenting imaging data, but also to improve trust by the medical personnel. There are different approaches to explain models and only some of them are useful for medical imaging data. What is more, many of the explainability (XAI) techniques are hard to evaluate and compare. In this thesis, focus on the state-of-the-art XAI methods and evaluate them in the context of medical imaging.

Electronically approved by Ing. Karel Klouda, Ph.D. on 10 February 2022 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Explainability in Medical Imaging

Bc. Adam Skluzáček

Department of Applied Mathematics
Supervisor: Ing. Jakub Žitný

May 5, 2022

Acknowledgements

I would like to express my gratitude to my supervisor, Ing. Jakub Žitný for his guidance and encouragement throughout the process of writing this thesis. The same goes to my loved ones who kept supporting me throughout the many struggles of writing this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 5, 2022 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Skluzáček, Adam. *Explainability in Medical Imaging.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

Abstrakt

Zvyšující se složitost moderních modelů strojového učení z nich udělala neprůhledné černé skříňky, což komplikuje jejich nasazení v kritických oborech jako například zdravotnictví. Tato práce zkoumá obor vysvětlitelné umělé inteligence z pohledu lékařského snímkování a detailně popisuje několik vhodně zvolených nejmodernějších vysvětlovacích metod. Tyto vysvětlovací metody jsou následně vyhodnoceny na modelech Resnet50 a Vision transformer natrénovaných na úloze detekce onemocnění covid-19 na základě rentgenových snímků hrudníku.

Klíčová slova Vysvětlitelnost, interpretabilita, vysvětlitelná umělá inteligence, XAI, lékařské snímkování

Abstract

The increasing complexity of the state-of-the-art machine learning models has caused them to become opaque black boxes. This hinders their deployment in critical domains such as healthcare. This thesis studies the field of explainable artifical intelligence in the context of medical imaging. It selects and detailedly describes various state-of-the-art explanation methods. The explanation methods are used to explain and evaluated on ResNet50 and Vision transformer models trained for the task of covid-19 pneumonia detection from chest X-ray scans.

Keywords Explainability, Interpretability, Explainable Artificial Intelligence, XAI, medical imaging

Contents

List of Figures

List of Tables

Introduction

The field of artificial intelligence has witnessed a dramatic rise of machine learning models that arise from neural network architectures. Thanks to the increasing computational power and dataset sizes such models are becoming more complex. On one hand, the increasing complexity of deep neural networks drives their performance forward which motivates their deployment in practice to the point that they are becoming ubiquitous in our everyday lives. On the other hand the models are becoming more black box which hinders their deployment in critical domains such as healthcare.

This has lead to a recent rapid rise of interest in the field of explainable AI that aims to produce methods that uncover the reasoning of complex models. Such explanations are valuable for doctors who can validate whether or not the model has learned reasonable features and is safe to be used as a guide in performing diagnoses. They are also valuable for the machine learning practitioners for debugging and evaluating their models.

While various explanation methods were introduced in the recent years an insufficient attention was paid to how the explanations should be evaluated in terms of their faithfulness to the model being explained. The lack of available ground truth makes such evaluation difficult which has lead the researchers to often rely on nonquantitative visual inspection which is an inadequate form of evaluation.

This thesis aim to research and describe the state-of-the-art explanation methods that are suitable in the context of medical imaging. The second part of the research is dedicated to quantitative evaluation of the explanation's faithfulness to the model being explained.

In the theoretical part 1, section 1.1 briefly introduces the concepts from machine learning and neural networks that are relevant for the further description of the individual explanation methods. Section 1.2 provides a detailed description of 9 selected state-of-the-art explanation methods. Section 1.3 describes various methods for evaluating the explanation's faithfulness.

In the experimental part 2, various convolutional neural networks and a vision transformer are trained on the task of covid-19 pneumonia detection from chest X-ray images. The corresponding models are subsequently explained by all the described explanation methods and the explanation's faithfulness is evaluated through various experiments in section 2.4.

Chapter 1

Theoretical Part

1.1 Machine learning

Machine learning classifier is a function $f : \mathbb{R}^n \mapsto \mathbb{R}^{|C|}$, where $\mathbb{R}^n$ is n-dimensional input space and $\mathbb{R}^{|C|}$ is output space, where $C = \{c_1, c_1, \ldots, c_m\}$ denote the set of classes. The predicted class is generally given as the argument of the maxima in the output vector. Alternatively a classifier can directly return the predicted class: $f : \mathbb{R}^n \mapsto C$. A special case of classification is binary classification, where $C = \{0, 1\}$.

1.1.1 Perceptron

An example of a binary classifier is a simple perceptron illustrated in figure 1.1.

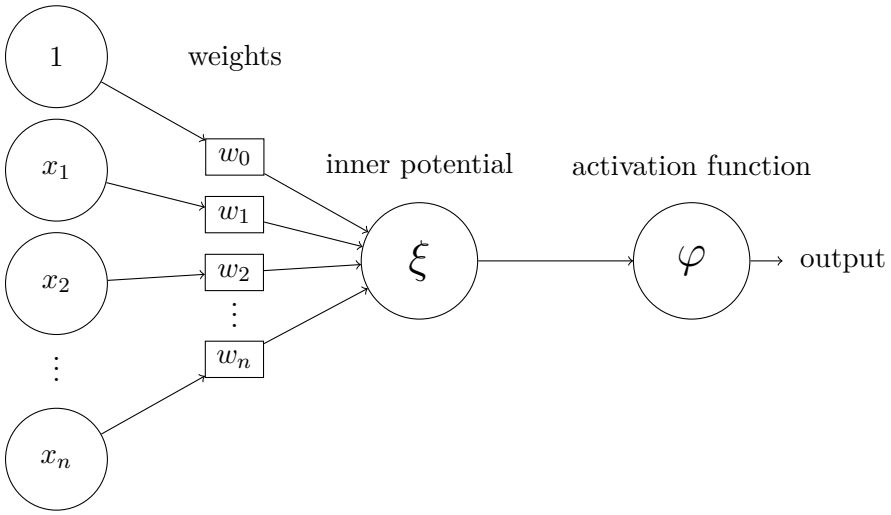


Figure 1.1: Architecture of a simple (single-layer) perceptron classifier

Where ξ denotes the inner potential given as the weighted sum of inputs:

$$\xi = w_0 + \sum_{i=1}^{n} w_i x_i \tag{1.1}$$

and φ denotes an activation function which is in case of a simple perceptron given as the threshold function:

$$\varphi(\xi) = \begin{cases} 1 & \xi \geq 1 \\ 0 & \xi < 1 \end{cases} .$$

1.1.2 Multi-layer perceptron

The multi-layer perceptron, also called the feedforward neural network consists of simple perceptrons organized in layers and interconnected in a way that the outputs of one layer of perceptrons are used as inputs into the next layer [1]. An example architecture of a multi-layer perceptron with one hidden layer and two output neurons (perceptrons) for binary classification is given in figure 1.2.

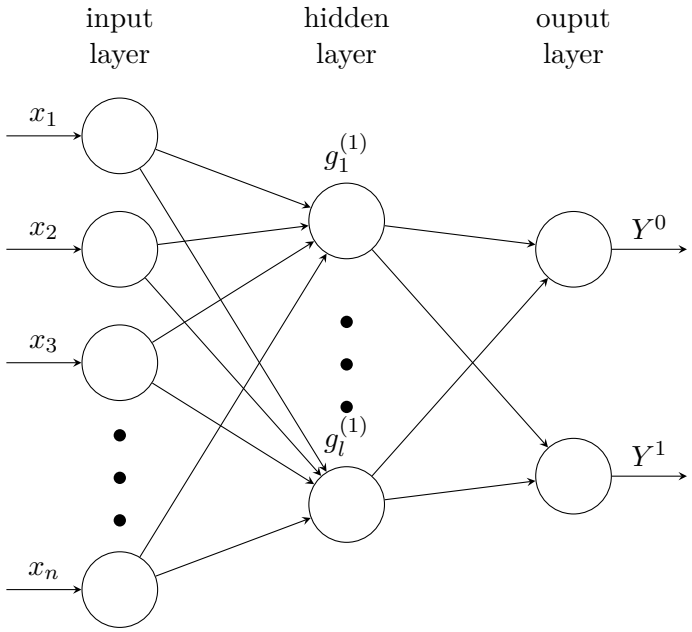


Figure 1.2: Example of a multi-layer perceptron (feedforward neural network) architecture with a single hidden layer

The activation functions used in the multi-layer perceptron are generally not the same as for the simple perceptron. A popular choice is the Rectified linear unit activation function ($ReLU$) defined as:

$$ReLU(\xi) = max(0, \xi). \tag{1.2}$$

In order for the model to directly output the probabilities of the given classes, the softmax function is often added after the last layer. Softmax function for class i is given as:

$$softmax(Y^1, Y^2, \ldots, Y^{|C|})_i = \frac{e^{Y^i}}{\sum_{c=1}^{|C|} e^{Y^c}}, \tag{1.3}$$

where $Y^1, Y^2, \ldots, Y^{|C|}$ denote the outputs of the neurons in the last layer.

Some of the explanation methods described in section 1.2 use the pre--softmax outputs to compute the explanations. To avoid ambiguity the following notation is used throughout the thesis: f denotes the post-softmax output (probabilities) of a classifier, f^c denotes the post-softmax output for a class of interest c, Y denotes the pre-softmax output of a classifier and Y^c denote the pre-softmax output for a class of interest c.

1.1.3 Supervised training

As illustrated in figure 1.1 each perceptron (neuron) has associated weights w_i for each input feature x_i. In supervised learning, these weights are trained (approximated) using a set of annotated pairs of inputs and target outputs (x, y) called the training dataset.

The training is performed by an optimization algorithm such as stochastic gradient descent (SGD) [1] or Adam [2] which aims to minimize a certain loss function over the whole training dataset.

A typical loss function used for training a classifier is the categorical cross-entropy loss defined as:

$$L(y, f(x)) = \sum_{c=1}^{|C|} \mathbb{1}_{c=y} log f^c(x), \tag{1.4}$$

where $f^c(x)$ denotes the probability of class c predicted by the model f and (x, y) is an input-output pair from the training data.

The training of neural networks is described in more detail in the Deep Learning book by Goodfellow, Bengio and Courville [1].

1.1.4 Convolutional neural networks

Convolutional neural networks (CNNs) are specialized neural network architectures used for processing data that has a grid-like topology, such as ECG signals, X-ray scan images or MRI scans. Convolutional neural networks leverage sparse connectivity and parameter sharing which makes them much more efficient on such data compared to the classic feedforward neural networks.

A typical convolutional layer consists of three steps: convolution, application of nonlinearity (activation function) and pooling. Convolution and pooling are described in the following subsections. In some literature (and

implementations), the convolution, nonlinearity application and pooling are considered as individual layers, however their meaning is the same [1].

It has been observed by many researchers that creating a deep network that consists of such constructed convolutional layers allows the network to learn high-level features which are then used as an input for a fully connected layer that performs the final classification [3].

1.1.4.1 Convolution

In convolutional neural networks, the convolution operation corresponds to application of a convolutional kernel (filter) over the whole input. The kernel is applied by obtaining its dot product with the input at each spatial location as is illustrated in figure 1.3. By making the kernel smaller than the input a corresponding neuron does not require the outputs of all neurons in the previous layer but only of those necessary to compute the dot product. This property is called sparse connectivity. Moreover, the kernels' weights (parameters) are shared among the neurons which is again in contrast with the feedforward neural network, where each neuron has its own set of parameters [1].

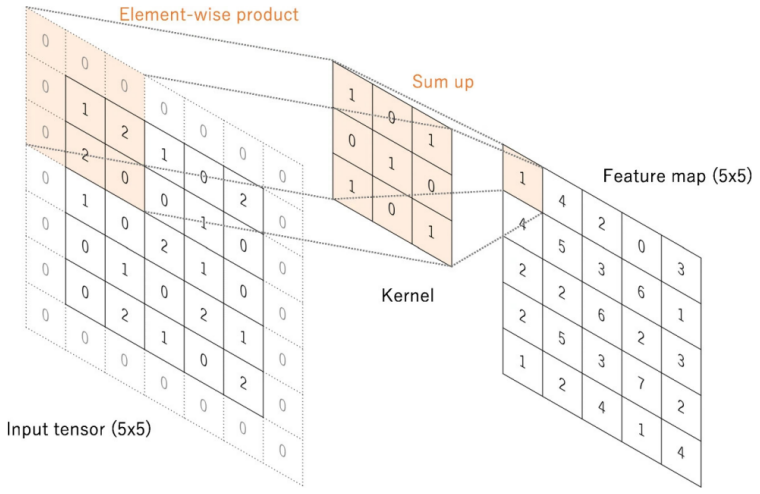


Figure 1.3: Illustration of the convolution operation, where a 3×3 filter (kernel) is applied on a two dimensional input. The result of convolving over the whole input is reffered to as feature map.

1.1.4.2 Pooling

Pooling operation aim to down-sample its input by aggregating input values at certain spatial locations. The most common aggregation choice is taking

the maximum value among the considered spatial locations as illustrated in figure 1.4.

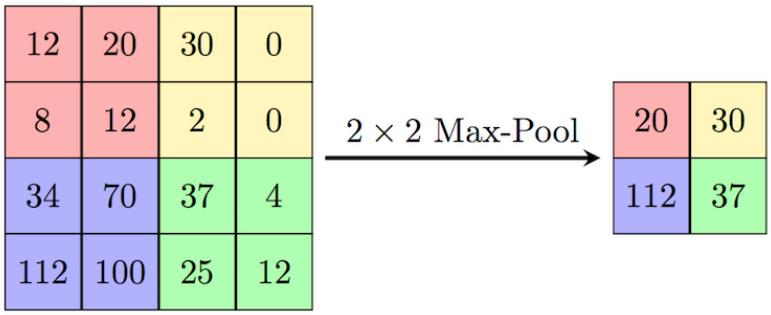


Figure 1.4: Illustration of the max-pooling operation.

The pooling operation makes the internal representation to some extent invariant to translations. This means that the presence of a given feature is more imporant than its exact location in the image. While this property has been criticized by some researchers, the use of pooling has been proven to be valuable in practice [4].

1.1.4.3 Architectures

This subsection briefly introduces the two CNN architectures that are trained in the experimental part of this thesis.

ResNet50 [5] Prior CNN architectures such as VGG [6] increased their accuracy mainly by adding more layers. The authors of ResNet observed that at some point adding more layers to the model degrades it's performance rather than increase it due to the vanishing gradient problem. The authors tackle this problem by introducing skip connections, which makes the output of a certain layer flow not only to the directy subsequent layer but also to some more distant layer. The ResNet50 architecture is illustrated in figure 1.5.

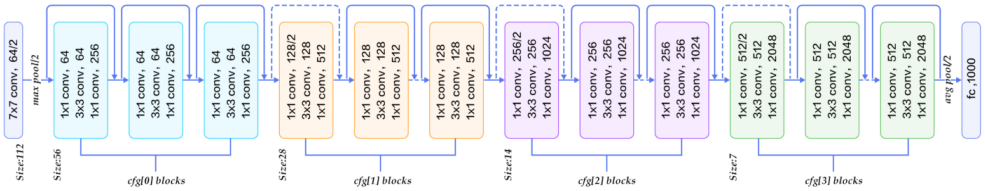


Figure 1.5: Illustration of the ResNet50 architecture.

DenseNet121 [7] The DenseNet architecture takes the skip connection concept even further. In Densenet121 multiple convolutional layers are grouped into 4 dense blocks, where the inputs of these dense blocks are given as out-

puts of all prior dense blocks in the network. The DenseNet121 architecture is illustrated in figure 1.6.

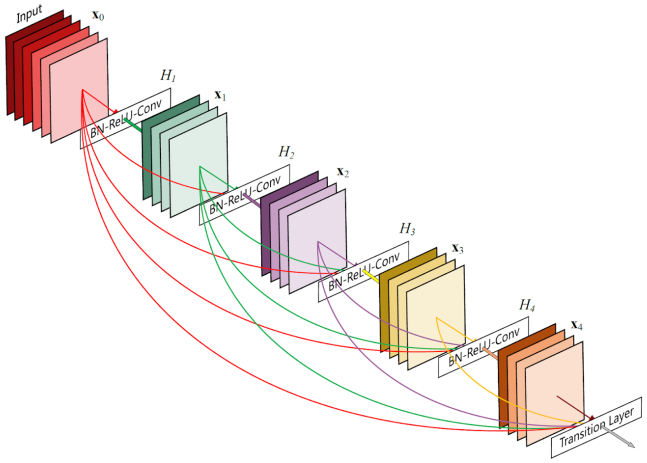


Figure 1.6: Illustration of the DenseNet121 architecture.

1.1.5 Vision transformer

Transformer is a neural network architecture proposed in [8] which quickly became the state-of-the-art for natural language processing tasks. Its main contribution is the utilization of the concept of self-attention and attention in general.

Let $X \in \mathbb{R}^{n \times d}$ denote a matrix where each feature of an input $x \in \mathbb{R}^n$ was embedded into a d dimensional space. Let $W^Q \in \mathbb{R}^{d \times d_q}, W^K \in \mathbb{R}^{d \times d_k}$ and $W^V \in \mathbb{R}^{d \times d_v}$ such that $d_q = d_k$ denote the querries, keys and values weight matrices. The self-attention is given as:

$$softmax\left(\frac{QK^\intercal}{\sqrt{d_q}}\right)V, \tag{1.5}$$

where $K = XW^K, Q = XW^Q$ and $V = XW^V$ are referred to as the key, querry and value matricies.

The interpretation of the self-attention is that it captures how salient the embedded features are with each other. This makes the self-attention interesting not only in the context of improving performance of a model but also to uncover the feature interactions the model has learned through the weight matricies W^Q, W^K and W^V.

In the context of images the equation 1.5 is impractical given that the $QK^\intercal$ matrix is of size $n \times n$, where number of features n is equal to the number of pixels in an image. Dosovitskiy et al. propose a solution to this by not using the individual pixels as features but rather splitting the input image into fixed size patches and using them as features instead.

The proposed model is called the Vision transformer and its overview is given in figure 1.7. The important part to note is that the class embedding is added to the patch embeddings. This means that the self-attention is computed not only between the feature embeddings but also between the class embedding and feature embeddings. As a result the self-attention could be intuitively utilized to produce an explanation that is implicitly true to the model.

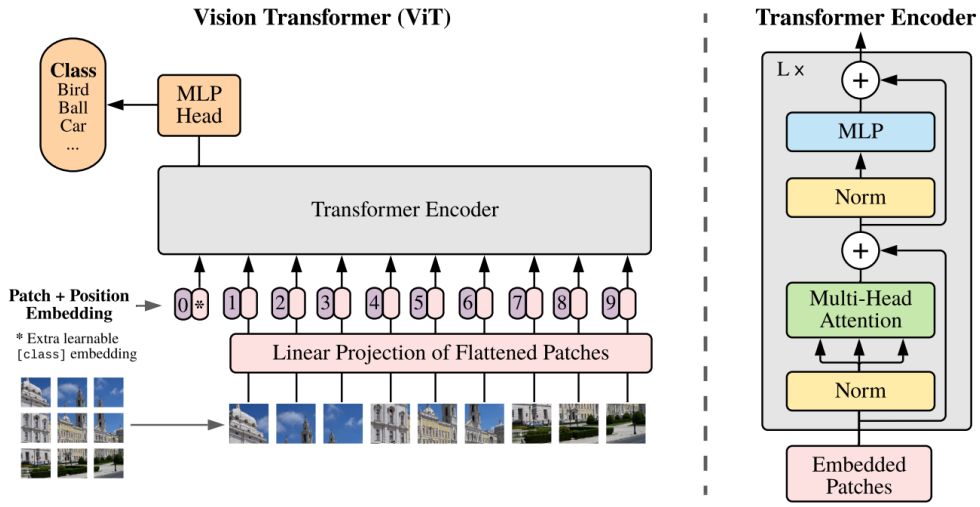


Figure 1.7: Overview of the Vision transformer model.

For more detail, please referr to the original Visual transformer paper [9].

1.2 Explanation methods

This sections describes majority of the research done in this thesis. Subsection 1.2.1 describes the taxonomy of explanation methods. Subsection 1.2.2 discuss various solutions to a common dilemma in the field of explainable AI which is how to simulate missing features for models that require complete inputs.

Further subsections present detailed descriptions of the state-of-the-art explanation methods for the task of diagnostic imaging. The selection was derived through an extensive research of the up to date XAI publications. Subection 1.2.3.1 describes the LIME method that explains a certain prediction of an arbitrary model by trainng a locally faithful surrogate model. Subsection 1.2.4.1 introduces various explanation methods that are based on

concepts from the coalition game theory. Subsection 1.2.5 describes various gradient based explanation methods. Subsection 1.2.6 introduces a final category of explanation methods that aim to explain a neural network through restricting the flow between its layers.

1.2.1 Taxonomy

Explanation methods are primarily divided based on the scope of their explanations as described in subsection 1.2.1.1 and by the variety of models they can be used to explain as described in subsection 1.2.1.2.

Explanation methods can be further divided based on the form in which the explanations are presented. This thesis focuses solely on the feature importance (or attribution) explanations that show what influence did each input feature have on the model's prediction. In the context of medical imaging the input features correspond to individual or multiple pixels in the input image. In images, the feature importances are therefore often referred to as pixel importances (or attributions). Pixel importances organized in the same shape as the input images are then referred to as saliency maps [10].

1.2.1.1 Explanation's scope

Explanation methods are divided based on the scope of their explanations to global and local.

Global explanation methods aim to explain an entire model at once. In other words they aim to produce an explanation that would explain the decision making of the given model on its whole input space. While such explanations are desirable as they could provide new valuable insight in tasks where the machine learning models surprass the human-level performance, they are mostly unattainable in practice [10].

Local explanation methods aim to explain the decision making of a model only for a single prediction. The idea is that focusing on a single prediction of the model can make the explanation process feasible even for models that are too complex to be explained globally.

1.2.1.2 Model specificity

Explanation methods are divided based on their specificity to model-agnostic and model-specific.

Model-agnostic explanation methods can be used to explain an arbitrary model as they treat the model as a black box and study it only through its inputs and corresponding outputs. The downside to their flexibility is their higher computational complexity.

Model-specific explanation methods are designed to explain a certain class of models by utilizing the information in model's internals (such as

the backpropagated gradients) which makes them computationally more efficient. Moreover, in the context of convolutional neural networks, the access to the network's feature maps can allow the explanation method to uncover the learned higher-level features.

1.2.2 Simulating missingness in images

An important concept in the explainable AI is simulating the missingness of given features (pixels). It is an analogy to the human reasoning through counterfactual thinking, where a person assignings importance to certain events by imagining the change of outcome in cases when the particular event did not occur [11]. Similar intuition is used by various explanation methods as well as explanation evaluation methods. Hiding a certain part of an input and observing the effect it has on the model's output should give a hint on how important the given part of the input is to the model. However, most of the machine learning models and specifically neural networks generally can't make a prediction of an incomplete input. The missingness therefore has to be approximated by replacing the pixels that are supposed to be missing with values that hide the information in the corresponding pixels while keeping the input image complete.

First publications in XAI that dealt with the concept of missingness suggested to simply zero out the features that are supposed to be missing. This approach is generally successful in hiding the information in the particular features, depending on the task domain. Subsequent publications have recognized that by using this approach the inputs with zeroed out features come from a different distribution than the original inputs on which the model was trained. The distribution shift complicates the assessment of the hidden features' importance as it is unclear how to evaluate whether the change in model's prediction was caused by hiding an important part of the input or by the shift in distribution [12]. This has led to introduction of multiple alternative methods that try to simulate missingness while minimizing the distribution shift. The current methods used in the image domain can be divided into two categories – baseline images and imputation methods, both are presented in the following subsections.

1.2.2.1 Baseline image

Baseline image $z \in \mathbb{R}^{w \times h \times d}$ is an image of the same size as the input image of interest $x \in \mathbb{R}^{w \times h \times d}$, where each pixel $z_{i,j,k}$ simulates the missingness of pixel $x_{i,j,k}$ in the corresponding spatial location (i, j, k). Let $m \in \{0, 1\}^{w \times h \times d}$ denote a binary mask, where $m_{i,j,k} = 1$ indicate that pixel at (i, j, k) should be missing from the input image and $m_{i,j,k} = 0$ otherwise. The image $x' \in \mathbb{R}^{w \times h \times d}$ that simulates x with the corresponding pixels missing is

then given pixel-wise as:

$$x'_{i,j,k} = \begin{cases} x_{i,j,k} & m_{i,j,k} = 0 \\ z_{i,j,k} & m_{i,j,k} = 1 \,. \end{cases} \tag{1.6}$$

Popular baseline image choices are introduced below. Visualisation of the effect of different baseline images in the chest X-ray domain is shown in figure 1.8.

Black color image: Missing pixels are simply simulated by zeroing out:

$$z_{i,j,k} = 0. \tag{1.7}$$

Mean color image: Whether the black pixel can be considered as non-informative depends on the image domain. In some domains the mean color of x can be a more suitable alternative:

$$z_{i,j} = \frac{1}{wh} \sum_{a}^{w} \sum_{b}^{h} x_{a,b}. \tag{1.8}$$

Both black and mean color images are constant color baselines which means they are inherently unable to "hide" pixels of the same color as the baseline. As a result, by using constant color baseline the pixels that are of the same color will be always deemed as unimportant [13]. The constant color baselines are therefore suitable only in domains, where the corresponding color is guaranteed to be uninformative.

Random image: Arguably the most uninformative domain-invariant value is random noise. The pixels of random image baseline are either sampled from uniform distribution:

$$z_{i,j,k} \sim \mathcal{U}(0, 1), \tag{1.9}$$

or from normal distribution:

$$z_{i,j,k} \sim \mathcal{N}(\mu_x, \sigma_x^2), \tag{1.10}$$

where μ_x and σ_x^2 denote the mean and variance of pixels in x.

As can be seen in rows 2-4 in figure 1.8 all three baseline images mentioned above seem to be hiding the information in the corresponding pixels sufficiently well. However it is also easy to see that these methods make the modified images x' significantly different compared to the original image distribution. The following methods aim to hide the pixels information while minimizing the differences in the distributions of x and x'.

Sampled image: The baseline image is drawn from the same distribution as the input image. The underlying distribution is generally not available but can be approximated with the distribution $\mathcal{D}_{\text{train}}$ that was used for training the model:

$$z \sim \mathcal{D}_{\text{train}} \tag{1.11}$$

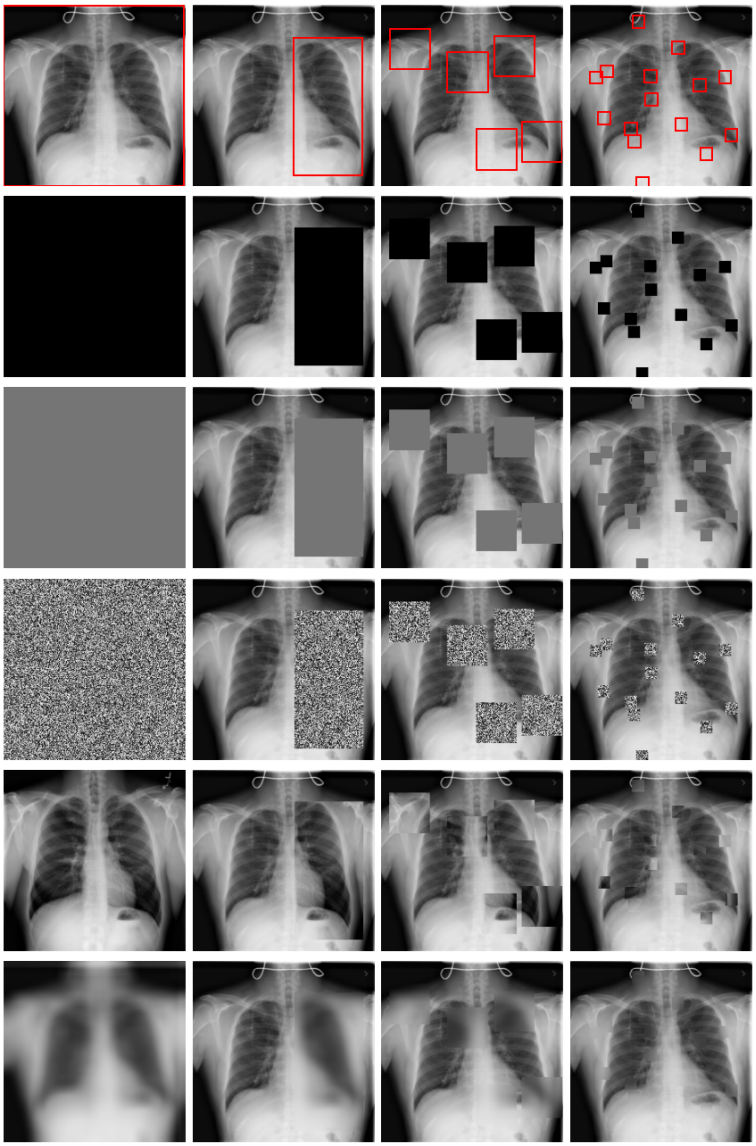


Figure 1.8: The effect of hiding various parts of the chest X-ray image by different baseline images. **Top**: original chest X-ray image and various highlighted segments that are supposed to be missing. Then top to bottom: black baseline, mean color baseline, random uniform baseline, blurred baseline.

Blurred image: The baseline image is obtained by blurring the input image by an arbitrary *blur* function (such as Gaussian blur):

$$z = blur(x) \tag{1.12}$$

The blurred image is a baseline that fits very well with the human intuition of missingness. When computing a blurred image each pixel is given as a

weighted average of its neighborhood pixels. This means that for homogenous parts of the image, the input image x and baseline z will be similar and a similar drawback as in the constant color baselines occurs. As a result the blurred baseline is biased towards highlighting high-frequency parts of the image [13].

1.2.2.2 Imputation

A common machine learning problem is dealing with missing data in the collected datasets [14]. Due to the ubiquity of this problem, many methods that aim to impute the missing data based on the available data have been introduced over the years. While most of the XAI publications utilize baselines to simulate missingness, an alternative approach is to delete the pixels that are supposed to be missing and impute them by some imputation method based on the remaining pixels. In theory, the imputation methods that utilize the information of the remaining neighborhood pixels can produce modified inputs that are closer to the original image distribution [15]. On the other hand, it may be difficult to assess how much of the deleted information is imputed back and how much is really hidden from the model. This issue could be relevant when using some complex imputation technique, such as utilizing the generative adversarial networks [16]. An overview of two imputation methods that have been utilized in the XAI literature is given below and example of their imputations in chest X-ray image is given in figure 1.9.

Noisy linear imputation: In this method, the missing pixel is imputed by the linear combination of its neighbors. The assumption is that the neighborhood image pixels are highly correlated and therefore a missing pixel can be approximated by a weighted mean of its neighborhood pixels:

$$\begin{aligned} x'_{i,j} = {} & w_d \left(x'_{i+1,j} + x'_{i-1,j} + x'_{i,j+1} + x'_{i,j-1}\right) \\ & + w_i \left(x'_{i+1,j+1} + x'_{i-1,j+1} + x'_{i+1,j-1} + x'_{i-1,j-1}\right) \\ & + \eta\, \varepsilon, \end{aligned} \tag{1.13}$$

where w_d and w_i denotes weights for the direct and indirect neighbor pixels respectively and η denote the magnitude of added noise $\varepsilon \sim \mathcal{N}(0, 1)$. Rong et al. suggest $w_d = \frac{1}{6}, w_i = \frac{1}{12}$ and $\eta = 0.1$ [15]. This imputation method is used in an explanation evaluation method described in subsection 1.3.2.2.

Telea inpaint: Telea is a more advanced inpaint technique introduced in [17] and is described here only in highlevel. The method inpaints a given region by gradually imputing the missing pixels on its boundary, making its way into the center of the region. The inputation is again done by the known pixels in the neighborhood. Lundberg et al. propose to use this technique in their explanation method described in subsection 1.2.4 [18].

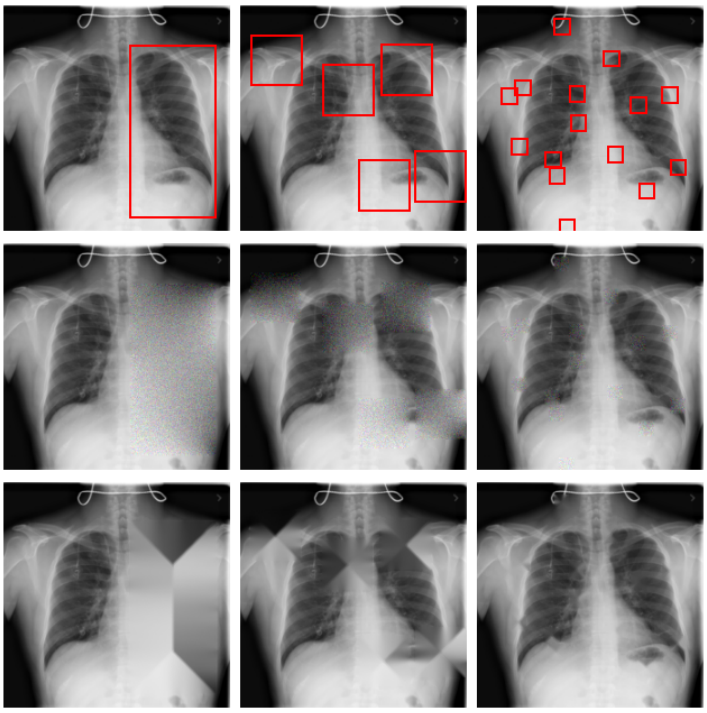


Figure 1.9: The effect of imputing various parts of the chest X-ray image by different imputing methods. **Top**: original chest X-ray image and various highlighted segments that are deleted. Then top to bottom: noisy linear imputation, telea inpaint.

1.2.3 Surrogate model based methods

Arguably the most straightforward method to globally explain an arbitrary model is through an interpretable surrogate model. Surrogate model g is trained to reproduce predictions of a model f. Surrogate models have been originally used in deep learning to speed up the inference of a model by reducing the complexity of the surrogate model. This is possible as training of the model f is done on a limited number of training examples which often requires the model to have more parameters than is necessary for the task to achieve high generalization [1]. However, for training the surrogate model g an arbitrary number of training examples are available as the model f is used to annotate the data. The assumption is that when g is interpretable and sufficiently accurate in reproducing predictions of f then g can serve as a global explanation of f. In practice, this is rarely the case as interpretable models such as logistic regression lack the necessary capacity to approximate complex models such as deep neural networks on the whole input space despite the unlimited training examples [10].

1.2.3.1 LIME

Local Interpretable Model-agnostic Explanations (LIME) introduced by Ribeiro et al. in [19] is a framework for providing a local explanation of an arbitrary model through a locally faithful surrogate model. Despite being originally introduced as a general framework, the majority of publications refer to LIME as a specific version of LIME that trains a sparse linear model as the local surrogate. The local faithfulness is achieved by weighting the training samples for the surrogate model proportionately to their distance from the input of interest as is illustrated in figure 1.10.

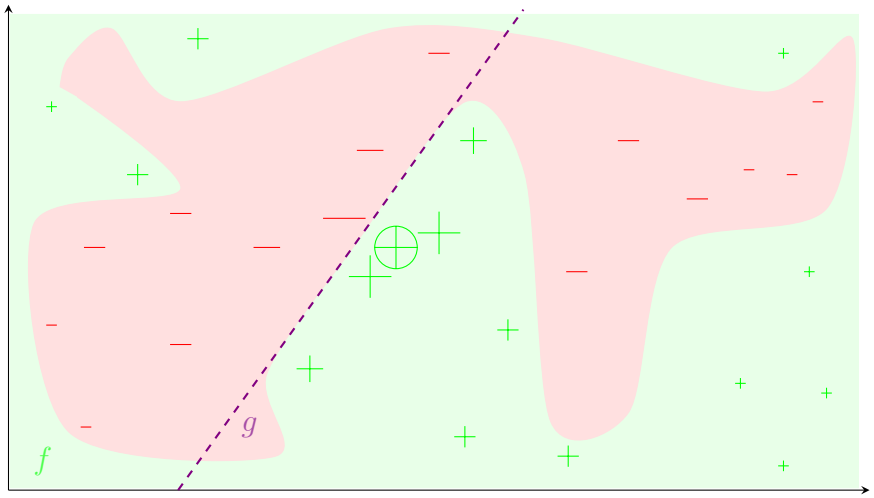


Figure 1.10: Illustration of Local Interpretable Model-agnostic Explanation (LIME). The background represents a hypothetical decision boundary of a model f being explained, where the green color represents subregion of the input space where the class of interest c is predicted by f and the red color represents subregion of the input space where a different class is predicted by f. The green marks represent the training samples for the local surrogate model with marker sizes indicating their weights. The violet line represents the linear local surrogate model g.

Formally, let $f : \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}^{|C|}$ denote the model being explained and $f^c : \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}$ denote model that returns prediction only for the class of interest c. For an input of interest $x \in \mathbb{R}^{w \times h \times d}$ let $\pi_x : \mathbb{R}^n \mapsto \mathbb{R}$ denote the proximity measure between x and another instance from $\mathbb{R}^n$. Ribeiro et al. suggest π_x as exponential kernel defined on some distance function $D : \mathbb{R}^{w \times h \times d} \times \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}$:

$$\pi_x(z) = exp(\frac{-D(x,z)^2}{\omega^2}), \tag{1.14}$$

where ω is the kernel width, on default set as $\omega = 0.75\sqrt{d}$, where d denotes the

number of features used for the surrogate model. Choice of distance function D is domain specific and for images L2 distance is suggested:

$$D(x, z) = \sum_{i=0}^{w} \sum_{j=0}^{h} \sum_{k=0}^{d} (x_{i,j,k} - z_{i,j,k})^2, \tag{1.15}$$

where $x_{i,j,k}$ and $z_{i,j,k}$ denote a pixels at spatial location (i, j, k) in image x and z respectively.

The proximity measure $\pi_x(z)$ is used to weight the training samples $z \in \mathcal{Z}_x$ for the surrogate model annotated by the model f. Samples z are constructed from the input of interest x by replacing random features of x with features drawn from normal distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu, \sigma^2 \in \mathbb{R}^{w \times h \times d}$ denote means and variances of pixels at each spatial location that can be estimated from the training examples used for the model f.

In the image domain Ribeiro et al. suggest using superpixels (groupings of multiple pixels) as features for the linear model instead of the individual image pixels. The input image of interest x is therefore segmented into d superpixels via a segmentation algorithm such as Quickshift [20] used by Ribeiro et al. Let x' denote the segmented input x and z' denote a segmented training sample z. Note that superpixels are now treated as individual features, therefore when constructing the training samples z all pixels in the corresponding segment have to be either taken from x or drawn from the normal distribution $\mathcal{N}(\mu, \sigma^2)$. The linear surrogate model g_c for the model f and class of interest c is then given as:

$$g_c(x') = \sum_{i=0}^{d} w_i x'_i, \tag{1.16}$$

where w_i denotes the weight of segment x'_i and therefore the importance of all the pixels in this segment. Locally weighted square loss with L2 regularization $\mathcal{L}$ is defined as:

$$\mathcal{L}(f^c, g_c, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x (f^c(z) - g_c(z'))^2 + \lambda \|w\|_2^2. \tag{1.17}$$

And the weight vector $w \in \mathbb{R}^d$ is then obtained through minimizing $\mathcal{L}$:

$$w = \arg\min_{w} \mathcal{L}(f^c, g_c, \pi_x), \tag{1.18}$$

which is generally done by stochastic gradient descent or any other popular optimizer such as Adam. The LIME method is summarized in Algorithm 1.

Algorithm 1 LIME

Require: Model f, class of interest c, input of interest x
Require: Number of segments d, number of samples for training M
Require: Means and variances μ, σ^2 of all input pixels (exact or estimated)

$x' \leftarrow$ Quickshift(x) ▷ segmentation
$\mathcal{Z} \leftarrow \{\}$
for $m \in \{1, 2, \ldots, M\}$ **do** ▷ training data sampling
    $z' \leftarrow \{\}$
    for $i \in \{1, 2, \ldots, d\}$ **do**
        if Bernoulli$(0.5) == 1$ **then** $z'_i \leftarrow x'_i$
        else $z'_i \leftarrow \mathcal{N}(\mu_i, \sigma_i^2)$
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{z'_i\}$
$w \leftarrow \arg\min_w \mathcal{L}(f^c, g_c, \pi_x)$
return w

1.2.4 Game theory based methods

Numerous popular local explanation methods are based on concepts from game theory specifically on coalition games and Shapley values [21]. This includes model-agnostic methods such as estimating Shapley values via adaptive sampling [22], KernelSHAP which is a combination of Shapley values and LIME [18] and model-specific methods such as DeepSHAP [18] used for explaining deep neural networks.

1.2.4.1 Shapley values

Formally, a coalition game is defined as a tuple $\langle N, v \rangle$, where N is a finite set of players and $v : 2^{|N|} \mapsto \mathbb{R}$ is a characteristic function that maps every subset of players $S \subseteq N$ (coalition) to a number representing the coalition's collective payout and satisfies $v(\emptyset) = 0$ for an empty set $\emptyset$. Shapley value $\phi_i(N, v)$ is defined as the average marginal contribution of the player i over all permutations of the coalition:

$$\phi_i(N, v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!}(v(S \cup \{i\}) - v(S)). \tag{1.19}$$

where $|N|$ and $|S|$ denote the number of elements in N and S respectively. The averaging over all coalitions is necessary to correctly assign the contributions to players with overlapping skillsets.

Shapley values as defined in (1.19) are the only solution to the problem of fair distribution of the payout that satisfy the following axioms as proven in [21] by Lloyd Shapley:

Efficiency axiom: The sum of the Shapley values of all players is equal to the collective payout, so the whole payout is distributed among the players:

$$\sum_{i \in P} \phi_i(v) = v(N) \tag{1.20}$$

Symmetry axiom: If two players increase the payout of every coalition by the same amount, then their Shapley values are equal:

$$\forall i, j \in N : \forall S \subset N \backslash \{i, j\} \;\; v(S \cup \{i\}) = v(S \cup \{j\}) \Rightarrow \phi_i(v) = \phi_j(v) \tag{1.21}$$

Dummy axiom: If a player does not increase the payout of any coalition, then the player's Shapley value is equal to zero:

$$\forall i \in N : \forall S \subset N \;\; v(S \cup \{i\}) = v(S) \Rightarrow \phi_i(v) = 0 \tag{1.22}$$

Linearity axiom: If a characteristic functions of two games are linearly combined, then the corresponding Shapley values can be decomposed in the same way:

$$\forall \langle N, \alpha v + \beta w \rangle \, \forall i \in N : \phi_i(\alpha v + \beta w) = \alpha \phi_i(v) + \beta \phi_i(w), \tag{1.23}$$

where α, β are arbitrary real numbers.

The above axioms are a major contributor to the popularity of Shapley value based explanation methods as they are built on solid theory. Some researchers also suggest that the established theory makes Shapley value based explanations particularly compelling in law regulated fields such as healthcare or banking [10, 23].

1.2.4.2 Estimating Shapley values of input features

The intuition of leveraging Shapley values for construction of a saliency map explanation of an arbitrary model's prediction is straightforward: the individual input features (pixels or superpixels) are considered to be the players, the model's prediction is the features' payout and Shapley values then represent the feature importance scores.

Formally, let $f : \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}^{|C|}$ denote the model to be explained and $f^c : \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}$ denote its prediction only for the class of interest c. Let $x \in \mathbb{R}^{w \times h \times d}$ denote the input image of interest. x represents the set of n players, where $n = whd$ is the number of pixels in the image. For better clarity and without any loss of generalization let the players (pixels of x) be denoted in the vectorized (one-dimensional) form $N = \{1, 2, \ldots, n\}$. Let $\mathbb{E}[f^c]$ denote the expected prediction for class c of the model being explained and the collective payout of all players is then given as $f^c(x) - \mathbb{E}[f^c]$. The goal is to fairly divide this collective payout among the pixels based on their contribution to obtain a saliency map. However, the equation 1.19 can not

be directly used to calculate pixel's importance for the two reasons described below.

First, the equation performs a sum over all possible subsets of players, where the number of possible subsets grows exponentially with the number of players. This makes the solution computationally infeasible in most domains, especially in the medical imaging given the typical resolution of the images. Strumbelj and Kononenko [22] proposed to approximate the equation 1.19 by sampling the following equivalent formulation:

$$\phi_i(N, v) = \frac{1}{|N|!} \sum_{\psi \in \Psi(N)} (v(Pre^i(\psi) \cup \{i\}) - v(Pre^i(\psi)), \tag{1.24}$$

where $Pre^i(\psi)$ denotes the set of all players that precede the player i in permutation ψ.

Second, the characteristic function is a function of subsets of players. However, ML models generally can not make predictions based only on a subset of input features as most of them can not handle missing data. Various baseline choices to simulate missingness of input pixels were presented in Subsection 1.2.2. In the context of Shapley values estimation, Strumbelj and Kononenko suggest simulating missingness by random sampling the features that are supposed to be missing from the training data distribution under the assumption of feature independence [22]. However, image pixels tend to be highly correlated in their respective neighborhoods [15]. Lludenberg et al. propose to simulate missing pixels by blurring those pixels or erasing them and then inpainting them back by an inpaint method such as Telea [17] [18]. To not lose any generality, let $z \in \mathbb{R}^{w \times h \times d}$ represent a baseline image which pixels simulate missingness of the corresponding pixels in x. z could for example be an image randomly drawn from the training data or a blurred version of the image x. Let z be also denoted by the vectorized index. For a permutation $\psi \in \Psi(N)$ of vectorized pixel indexes N, an input image x and a missingness simulating image z, let $x_{+1}, x_{-1} \in \mathbb{R}^{w \times h \times d}$ denote two input instances constructed as:

$$\begin{aligned} x_{+i} &= (x_{\psi^{-1}(1)}, x_{\psi^{-1}(2)}, \ldots, x_{\psi^{-1}(i)}, z_{\psi^{-1}(i+1)}, \ldots, z_{\psi^{-1}(n)}), \\ x_{-i} &= (x_{\psi^{-1}(1)}, x_{\psi^{-1}(2)}, \ldots, z_{\psi^{-1}(i)}, z_{\psi^{-1}(i+1)}, \ldots, z_{\psi^{-1}(n)}). \end{aligned} \tag{1.25}$$

Such constructed instances x_{+1} and x_{-1} simulate terms $Pre^i(\psi) \cup \{i\}$ and $Pre^i(\psi)$ respectively from equation 1.24. Instead of removing the pixels that precede the pixel i in permutation ψ, they are replaced by pixels from z. x_{+1} and x_{-1} are therefore complete images that can serve as inputs for the model f.

Let $\phi_i(f^c, x)$ denote the Shapley value of pixel i for the model f, class c and input x. The approximation $\widehat{\phi}_i(f^c, x)$ by Monte Carlo sampling is given as:

$$\widehat{\phi}_i(f^c, x) = \frac{1}{M} \sum_{\substack{m=1 \\ \psi \in \Psi(N)}}^{M} (f^c(x_{+i}) - f^c(x_{-i})), \tag{1.26}$$

where M is a hyperparameter that determines the number of samples used in the approximation.

Strumbelj and Kononenko [22] showed, that when z is sampled from the training data $\mathcal{D}$, then $\widehat{\phi}_i(f^c, x)$ is a consistent and unbiased estimator of $\phi_i(f^c, x)$ and is approximately normally distributed:

$$\widehat{\phi}_i(f, x) \approx \mathcal{N}(\phi_i(f, x), \frac{\sigma_i^2}{M_i}), \tag{1.27}$$

where σ_i^2 denotes the variance of pixel i in the image distribution and M_i denotes the number of samples used for calculating $\widehat{\phi}_i(f, x)$. From (1.27) it follows that:

$$\widehat{\phi}_i(f, x) - \phi_i(f, x) \approx \mathcal{N}(0, \frac{\sigma_i^2}{M_i}). \tag{1.28}$$

This implies that the number of samples M_i needed to accurately approximate $\phi_i(f, x)$ depends solely on the pixel's i variance σ_i^2. As pixels at different spatial positions are likely to have different variances, Strumbelj and Kononenko suggest an adaptive sampling method to be used instead of the pseudo-random sampling in equation 1.26. First, as the true variances σ_i^2 are generally not know, they are first approximated, for example by the Knuth's incremental algorithm [24], and denoted as $\widehat{\sigma}_i^2$. Two hyperparameters M_{min} and M_{max} are then chosen, where M_{min} represents the minimum number of samples drawn for each pixel and M_{max} represents the maximum number of samples drawn for the whole image. After each pixel is sampled M_{min} times, the next pixel j to be sampled is chosen as:

$$j = \arg\max_j (\frac{\sigma_j^2}{m_j} - \frac{\sigma_j^2}{m_j + 1}), \tag{1.29}$$

where m_j denotes the number of samples for the feature j drawn so far. The sampling stops after drawing M_{max} samples. This adaptive sampling strategy minimizes the following squared loss:

$$\sum_{i=1}^{n} (\widehat{\phi}_i(f, x) - \phi_i(f, x))^2. \tag{1.30}$$

The Shapley values estimation method as described in [22] is summarized in Algorithm 2.

Algorithm 2 Shapley values estimation via adaptive sampling

Require: Model f, class of interest c, input image x
Require: Baseline image distribution $\mathcal{D}$, pixels' variance vector σ^2 (exact or estimated)
Require: Minimum number of samples for each pixel M_{min} and maximum number of samples for all the pixels M_{max}

for $i = 1$ to n **do** ▷ Initialization
  $m_i \leftarrow 0$
  $\phi_i \leftarrow 0$ ▷ $\widehat{\phi}_i(f, x)$
while $\sum_{i=1}^{n} m_i < M_{max}$ **do**
  if $\exists i : m_i < M_{min}$ **then**
    select pixel j to be sampled s.t. $m_j < M_{min}$
  else
    select pixel j to be sampled s.t. $j = \arg\max_j (\frac{\sigma_j^2}{m_j} - \frac{\sigma_j^2}{m_j+1})$
  $\psi \leftarrow \Psi(N)$
  $z \leftarrow \mathcal{D}$
  $x_{+i} \leftarrow (x_{\psi^{-1}(1)}, x_{\psi^{-1}(2)}, \ldots, x_{\psi^{-1}(i)}, z_{\psi^{-1}(i+1)}, \ldots, z_{\psi^{-1}(n)})$
  $x_{-i} \leftarrow (x_{\psi^{-1}(1)}, x_{\psi^{-1}(2)}, \ldots, z_{\psi^{-1}(i)}, z_{\psi^{-1}(i+1)}, \ldots, z_{\psi^{-1}(n)})$
  $\phi_i \leftarrow \phi_i + (f(x_{+i}) - f(x_{-i}))$
  $m_j \leftarrow m_j + 1$
for $i = 1$ to n **do**
  $\phi_i \leftarrow \frac{\phi_i}{m_i}$
return ϕ ▷ Vector of Shapley values

1.2.4.3 Owen values

Lundberg et al. [18] propose and implement various other model-agnostic and model-specific methods to approximate the Shapley values. One model--agnostic method that is particularly interesting in the context of computer vision is based on **Owen values** which are a generalization of Shapley values. In Owen values, the players are considered to be forming sub-coalitions that are non-overlapping and indivisible and the coalitions are formed as subsets of those sub-coalitions. The sub-coalitions can therefore be viewed as individual players.

Formally for a coalition game $\langle N, v \rangle$ and a partition $C = \{C_1, \ldots, C_m\}$ such that $C_k \cap C_j = \emptyset$ for each $k \neq j$ and $\bigcup_{t=1}^{m} C_l = N$, the Owen value $\varphi_i(N, C, v)$ of C_i is defined as:

$$\varphi_i(N, C, v) = \frac{1}{|C|!} \sum_{\psi \in \Psi(C)} (v(Pre^{C_i}(\psi) \cup \{C_i\}) - v(Pre^{C_i}(\psi)). \tag{1.31}$$

Note that for $C = \{N\}$ the (1.31) becomes equal to (1.24) and thus $\varphi_i(N, C, v) = = \phi_i(N, v)$ [25]. The further derivations of using Owen values to explain an arbitrary model are equivalent to using Shapley values.

Owen values can therefore be used to calculate importances of groupings of pixels (superpixels) instead of individual pixels as in Shapley values. This results in two major advantages for the tradeoff of lower resolution saliency map. First, given the typical resolution of a medical image it is generally computationally infeasible to accurately approximate importances of each pixel individually. Grouping pixels into superpixels (Lundberg et al. use a simple grid) dramatically reduces the number of features and therefore reduces the computational cost. Secondly, Strumbelj and Kononenko impose a feature independence assumption in their derivations, which is generally violated in the context of images as especially neighborhood pixels tend to be highly correlated. Grouping neighborhood pixels into superpixels helps to partially account for such pixel correlations [18].

1.2.4.4 Other Shapley value based explanation methods

Other model-agnostic methods proposed and implemented by Lundberg include KernelSHAP, that uses the LIME framework introduced in the subsection 1.2.3.1 to estimate the Shapley values by choosing an appropriate loss function and proximity measure. Numerous model-specific methods are also proposed such as DeepSHAP for neural networks that utilizes a gradient based explanation method DeepLIFT [26].

Frye et al. [27] propose Asymmetric Shapley values. The symmetry axiom (1.21) implies that Shapley values uniformly distribute the feature (pixel) importance over identically informative features. Frye et al. argue that when redundancies exist a sparse explanation should be desired instead. They propose to relax the symmetry axiom and incorporate domain specific causal knowledge via a causal diagram. Similar approach for incorporating causality into the Shapley values is presented in [28].

Subsection 1.2.5.1 describes a gradient based explanation method called Integrated gradients that approximates Aumann-Shapley values which are an extension of Shapley values for non-atomic games (coalition games with infinite number of players) [29, 30].

1.2.5 Gradient based methods

Majority of the model-specific methods for explaining convolutional neural networks via saliency maps utilize the gradient of the model's output with respect to the input or feature maps at a given layer [10].

Consider a simple example of a linear score model for class c:

$$Y^c(I) = w_c^\intercal x + b_c, \tag{1.32}$$

where x is the input image in a flattened (one-dimensional) form, w_c is the weight vector and b_c is the bias. As the output is simply a linear combination of the input pixels, the pixels' importances are given by the weight vector, which can be trivially obtained as the gradient of output Y^c with respect to the input image x.

In contrast, convolutional neural networks are highly non-linear functions of the input, therefore the same logic doesn't directly apply. The output of a CNN can however be approximated with a linear function in the neighborhood of x by computing the first (or higher) order Taylor expansion:

$$Y^c(I) \approx w^\intercal x + b, \tag{1.33}$$

where w is the gradient of class score with respect to the input pixels:

$$w = \left.\frac{\partial Y^c}{\partial x}\right|_{x_0=x}, \tag{1.34}$$

which can be computed via a single backward pass through the network [31]. This baseline gradient based explanation method is often referred to in literature as **vanilla gradient**. For the final explanation the vanilla gradient is often combined with the input image:

$$x \odot \left.\frac{\partial Y^c}{\partial x}\right|_{x_0=x}, \tag{1.35}$$

where $\odot$ denote the element-wise product.

Subsections below introduce a variety of state-of-the-art gradient based explanation methods. Subsection 1.2.5.1 describes Integrated gradients that addresses some of the limitations of previously introduced gradient based methods by building axiomatic explanations similar to Shapley value based methods. A further advancement by incorporating bias-gradients – FullGrad is presented in subsection 1.2.5.2. Subsections 1.2.5.3 and 1.2.5.4 introduce a family of CAM methods that don't propagate pixels' attributions all the way to the input but only to the activation maps in the last convolutional layer. Subsection 1.2.5.5 then propose to combine the CAM explanations with some other gradient based method that propagates the attributions all the way to the input to obtain a high resolution saliency map.

1.2.5.1 Integrated gradients

Integrated Gradients (IG) is a gradient based local feature attribution explanation method designed to satisfy two axioms – Sensitivity and Implementation invariance [29]. These axioms are deemed as fundamental by the authors Sundararajan et al., however, they show that prior gradient based explanation methods such as the Vanilla gradient, Layer-wise relevance propagation [32] or DeepLIFT [26] violate at least one of the two axioms.

Integrated gradients utilize a baseline image that simulate missingness of particular pixels in the image to calculate the pixel attributions. Various baseline choices are discussed in Subsection 1.2.2. The authors of IG suggest using a black image as the baseline.

The desired axioms for any explanation via attribution method are defined by Sundararajan et al. [29] as:

Sensitivity axiom: "An attribution method satisfies Sensitivity if for every input and baseline that differ in one feature but have different predictions then the differing feature should be given a non-zero attribution." Moreover, the authors include the **Dummy axiom** as a complement to the Sensitivity axiom: "If the function implemented by the deep network does not depend (mathematically) on some variable, then the attribution to that variable is always zero." This is equivalent to the Dummy axiom (1.21) from Shapley values.

Implementation invariance axiom: "Two networks are functionally equivalent if their outputs are equal for all inputs, despite having very different implementations. Attribution methods should satisfy Implementation invariance, i.e., the attributions are always identical for two functionally equivalent networks."

One of the reasons why these axioms are violated by simpler gradient based methods is the *ReLU* activation function that is widely adopted in deep learning [1]. $ReLU(x) = max(0, x)$ serves as a threshold function which can cause some features of the input to have zero gradient despite having non-zero importance [26, 29].

Formally, let $f : \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}^{|C|}$ denote the model being explained and $f^c : \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}$ denote model f that returns prediction only for the class of interest c. Further, let $x \in \mathbb{R}^{w \times h \times d}$ denote the input image of interest, $z \in \mathbb{R}^{h \times w \times d}$ denote the baseline image and let the pixels in both x, z be indexed by $\{1, 2, \ldots, n\}$, where $n = whd$ as if they were vectorized (one-dimensional). Integrated gradients are a special case of Path methods that aggregate gradients along an arbitrary monotonic path between two points in the same space [33]. Let $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n) : [0, 1] \mapsto \mathbb{R}^{w \times h \times d}$ be a smooth function specifying a path in $\mathbb{R}^{w \times h \times d}$ from the baseline z to the input x. Path integrated gradients along the ith pixel for an input x and model f^c is

defined as:

$$PG_i^{\lambda}(x) = \int_{\alpha=0}^{1} \frac{\partial f^c(\gamma(\alpha))}{\partial \gamma_i(\alpha)} \frac{\partial \gamma_i(\alpha)}{\partial \alpha} d\alpha. \tag{1.36}$$

Integrated gradients define path γ between z and x as a straight line:

$$\gamma(\alpha) = z + \alpha\,(x - z) \text{ for } \alpha \in [0, 1]. \tag{1.37}$$

The attribution $IG_i(x)$ of ith pixel in the input x for the model f^c is then computed by the integrated gradients as:

$$IG_i(x) = (x_i - z_i) \int_{\alpha=0}^{1} \frac{\partial f^c(z + \alpha\,(x - z))}{\partial x_i} d\alpha, \tag{1.38}$$

which is in practice approximated via a Riemann sum with m steps:

$$\widehat{IG}_i(x) = (x_i - z_i) \sum_{k=1}^{m} \frac{\partial f^c(z + \frac{k}{m}\,(x - z))}{\partial x_i} \frac{1}{m}. \tag{1.39}$$

The integration gradients in fact correspond to Aumann-Shapley values which are an extension of Shapley values, described in Subsection 1.2.4.1, for non-atomic games which are coalition games with infinite number of players [30]. This implies that apart of the sensitivity, dummy and implementation invariance axioms, the integrated gradients also satisfy the two following axioms:

Completeness axiom: If f^c is differentiable almost everywhere then:

$$\sum_{i=1}^{n} IG_i(x) = f^c(x_i) - f^c(z_i), \tag{1.40}$$

which is equivalent to the Efficiency axiom (1.20) in Shapley values.

Linearity axiom: Let f be a neural network, that is composed as a linear combination of two other neural networks $f = \alpha f_1 + \beta f_2$, where $\alpha, \beta \in \mathbb{R}$. The attributions $IG_i^f(x)$ for the model f are then given as $IG_i^f(x) = \alpha IG_i^{f_1}(x) + \beta IG_i^{f_2}(x)$. This is equivalent to the Linearity axiom (1.23) in Shapley values.

Given the satisfied axioms, the integrated gradients method can be viewed as an efficient, model-specific method to approximate the Shapley values. The method is summarized in Algorithm 3

Algorithm 3 Integrated gradients

Require: Model f, class of interest c, input of interest x, baseline x'
Require: Number of approximation steps m for each pixel
$\widehat{IG} \leftarrow \{\}$
for $i \in \{1, 2, \ldots, n\}$ **do** ▷ for each pixel
  $IG_i \leftarrow 0$
  for $k \in \{1, 2, \ldots, m\}$ **do**
    $\widehat{IG}_i \leftarrow \widehat{IG}_i + \frac{\partial f^c(x' + \frac{k}{m}(x - x'))}{\partial x_i}$
  $\widehat{IG} \leftarrow \widehat{IG} \cup \{\frac{(x_i - x'_i)\,\widehat{IG}_i}{m}\}$
return $\widehat{IG}$

1.2.5.2 FullGrad

FullGrad is an explanation method introduced by Srinivas in [34] that combines the input (vanilla) gradient with bias gradients of each activation map in a convolutional neural network. Similarly as Integrated gradients, the method is motivated by an axiom that is deemed as desariable by the author and violated by the prior gradient based explanation methods.

Weak dependence on inputs axiom: Consider a piecewise-linear model f:

$$f(x) = \begin{cases} w_1^\intercal x + b_1 & x \in \mathcal{U}_1 \\ \ldots & \\ w_n^\intercal x + b_n & x \in \mathcal{U}_n \end{cases}$$

where $\mathcal{U}_i$ are open connected sets. Then all inputs x from the same set $\mathcal{U}_i$ should be assigned the same pixels' importances as it depends only on the parameters w_i and b_i. Srinivas shows that prior gradient based methods such as the Integrated gradients do not satisfy this axiom which causes them to produce counterintuitive explanations. The author argues that this is due to the exclusion of the bias term, which causes a compounding effect as the neural networks have bias a term for every neuron.

Let $Y^c : \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}$ denote the logit (pre-softmax) output of a convolutional neural network $f : \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}^{|C|}$ for the class of interest c. Further, let l be the number of convolutional layers in f and let $A_i^1, A_i^2, \ldots, A_i^{d_i} \in \mathbb{R}^{w_i \times h_i}$ denote the activation maps (output) of convolutional layer i where w_i, h_i, d_i denote the corresponding dimensions. The FullGrad attributions $FG_f(x) \in \mathbb{R}^{w \times h \times d}$ for each pixel in the input of interest x are then given as:

$$FG_f(x) = \frac{\partial Y^c}{\partial x} \odot x + \sum_{i=1}^{l} \sum_{k=1}^{d_i} \psi(\frac{\partial Y^c}{\partial A_i^k} \odot b), \tag{1.41}$$

where ψ is an upscaling function that is necessary as the activation maps $A_i^k \in \mathbb{R}^{w_i \times h_i \times d_i}$ are generally of lower resolution than input image $x \in \mathbb{R}^{w \times h \times d}$. The FullGrad method is summarized in Algorithm 5.

Algorithm 4 FullGrad

Require: Model f, class of interest c, input of interest x
Require: Upscaling function ψ
$input\text{-}grad \leftarrow \frac{\partial Y^c}{\partial x} \odot x$ ▷ vanilla gradient $\odot$ input
$bias\text{-}grad \leftarrow 0$
for $i \in \{1, 2, \ldots, l\}$ **do** ▷ for each convolutional layer in f
    $A_i \leftarrow f_i(x)$ ▷ activation maps as output of f's layer i
    for $k \in \{1, 2, \ldots, d_i\}$ **do** ▷ for each activation map
        $bias\text{-}grad \leftarrow bias\text{-}grad + \psi(\frac{\partial Y^c}{\partial A_i^k} \odot b)$
$FG_f \leftarrow input\text{-}grad + bias\text{-}grad$
return FG_f

1.2.5.3 Grad-CAM

Gradient-weighted class activation mapping (Grad-CAM) is an explanation method introduced by Selvaraju et al. [35] designed specifically for convolutional neural networks, where the gradient of model's output is not back-propagated all the way back to the input image, but to the (usually) last convolutional layer.

Many researchers in the past have shown that convolutional units in the later layers of CNNs implicitly learn to localize objects in the images despite only being trained for whole-image classification [36]. This phenomenon has been initially leveraged to build object detection models without the requirement of detailedly annotated images with bounding boxes [37, 38] and more recently used in the context of explainable AI.

Grad-CAM method is an improved version of **CAM** (class activation mapping) method introduced in [36]. CAM is based on the activation maps in the last convolutional layer that indicate where the particular convolutional filter has detected the corresponding learned abstract feature of an input image. By identifying which activation maps were important for predicting a class of interest a saliency map can be produced as an upscaled aggregation of the relevant activation maps.

Formally, let $A^1, A^2, \ldots, A^l \in \mathbb{R}^{w_A \times h_A}$ denote the l activation maps in the last convolutional layer of the model $f : \mathbb{R}^{h \times w \times d} \mapsto \mathbb{R}^{|C|}$ being explained and let A_{ij}^k denote the spatial location (i, j) in the kth activation map. In CAM all the layers following the last convolutional layer (generally fully connected layers) are replaced by the global average pooling (GAP) layer and the logit (pre-softmax) classification score $Y^c : \mathbb{R}^{h \times w \times d} \mapsto \mathbb{R}$ for class c is given as

a linear combination of the global average pooled activation maps A^k:

$$Y^c = \sum_k w_k^c \underbrace{\frac{1}{Z} \sum_i \sum_j}_{\text{GAP}} A_{ij}^k, \tag{1.42}$$

where w_k^c denote the corresponding weights that are estimated by training a linear classifier for each class c on the same targets as the original model being explained and Z is the number of pixels in the activation map A^k. The resulting explanation in form of a class activation map M^c for class c is defined in each spatial location (i, j) as:

$$M_{ij}^c = \sum_k w_k^c A_{ij}^k. \tag{1.43}$$

Since $Y^c = \sum_i \sum_j M_{ij}^c$, the M_{ij}^c directly correlates with the importance of the activation at the spatial location (i, j) for the class of interest c. The visual overview of CAM is given in figure 1.11.

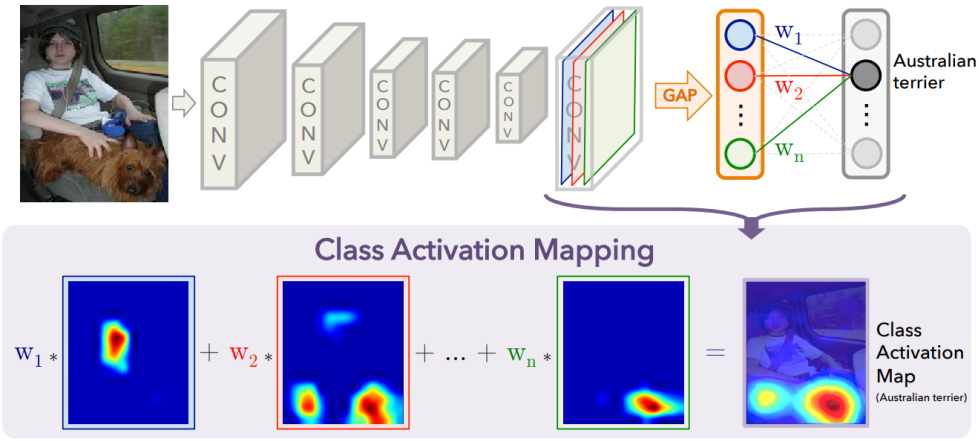


Figure 1.11: Visual overview of the CAM method. **Top:** example input image with the modified CNN architecture, where the layers following the last convolutional layer are replaced by the global average pooling layer. **Bottom:** example activation maps and their linear combination resulting in the class activation map for the class of interest (Australian terrier).

Grad-CAM method addresses the main apparent limitation of CAM – the need of training an additional linear classifier in order to produce explanations by utilizing gradient of the class score with respect to the activation maps. Let F^k denote the global average pooled activation map A^k:

$$F^k = \frac{1}{Z} \sum_i \sum_j A_{ij}^k. \tag{1.44}$$

This allows to rewrite the equation 1.42 as:

$$Y^c = \sum_k w_k^c F^k. \tag{1.45}$$

Taking the gradient of Y^c with respect to F^k gives:

$$\frac{\partial Y^c}{\partial F^k} = \frac{\frac{\partial Y^c}{\partial A_{ij}^k}}{\frac{\partial F^k}{\partial A_{ij}^k}}, \tag{1.46}$$

where $\frac{\partial Y^k}{\partial F^k} = w_k^c$ from (1.45) and $\frac{\partial F^k}{\partial A_{ij}^k} = \frac{1}{Z}$ from (1.44). Substituting both into (1.46) gives:

$$w_k^c = Z \frac{\partial Y^c}{\partial A_{ij}^k}. \tag{1.47}$$

Summing both sides of (1.47) over all spatial locations (i, j) gives:

$$\sum_i \sum_j w_k^c = \sum_i \sum_j Z \frac{\partial Y^c}{\partial A_{ij}^k}. \tag{1.48}$$

As Z and w_k^c do not depend on (i, j) and Z is the number of pixels in the activation map ($Z = \sum_i \sum_j 1$) the above equation can be rewritten as:

$$Z w_k^c = Z \sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^k}, \tag{1.49}$$

from which:

$$w_k^c = \sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^k}. \tag{1.50}$$

This result show that the weights w_k^c can be computed via the gradients of class scores with respect to the activation maps in the last convolutional layer and it is not necessary to train a linear classifier to obtain class activation maps [35].

In the Grad-CAM method the computed weights are further normalized:

$$w_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^k}, \tag{1.51}$$

and the linear combination of the activation maps are followed by *ReLU* function so that only features that have a positive influence on predicting the class of interest c are highlighted in the resulting class activation map:

$$M^c = ReLU(\sum_k w_k^c A^k). \tag{1.52}$$

Algorithm 5 GradCAM

Require: Model f, class of interest c, input of interest x

$w^c \leftarrow \{\}$
for $k \in \{1, 2, \ldots, l\}$ **do**
    $w_k^c \leftarrow 0$
    for $i \in \{1, 2, \ldots, w_A\}$ **do**
        for $j \in \{1, 2, \ldots, h_A\}$ **do**
            $w_k^c \leftarrow w_k^c + \frac{\partial Y^c}{\partial A_{ij}^k}$
    $w^c \leftarrow w^c \cup \frac{1}{Z} w_k^c$ ▷ $Z = w_A h_A$
$M^c \leftarrow \{0, 0, \ldots, 0\}$
for $k \in \{1, 2, \ldots, l\}$ **do**
    $M^c \leftarrow M^c + w_k^c A^k$
$M^c \leftarrow ReLU(M^c)$
return M^c

1.2.5.4 Grad-CAM++

Grad-CAM++ is a generalization of the Grad-CAM method introduced by Chattopadhyay et al. in [39]. Its authors argue that Grad-CAM often fails to properly highlight the entire object or all the objects in an image with multiple occurrences of the same class.

Equation 1.51 shows that Grad-CAM computes the weight w_k^c of activation map A^k for class c as an unweighted average of all pixel gradients. Treating each pixel equally when computing the importance of an activation map can suppress activation maps with comparatively lesser spatial footprint as is demonstrated in figure 1.12. In this hypothetical example with activation maps A^1, A^2, A^3 Grad-CAM is shown to assign lower weights to activation maps A^2, A^3 due to their smaller spatial footprint, despite their pixels being of the same importance as the pixels in A^1.

Grad-CAM++ solves this problem by introducing pixel-wise weights α_{ij}^{kc} for activation map k, class of interest c and spatial location (i, j). The weights w_k^c are then given as:

$$w_k^c = \sum_i \sum_j \alpha_{ij}^{kc} \, ReLU(\frac{\partial Y^c}{\partial A_{ij}^k}). \tag{1.53}$$

Substituting the weights w_k^c in (1.42) with (1.53) and removing the equal weightage constant $\frac{1}{Z}$ gives the class score Y^c as:

$$Y^c = \sum_k \{\sum_a \sum_b \alpha_{ab}^{kc} \, ReLU(\frac{\partial Y^c}{\partial A_{ab}^k})\}[\sum_i \sum_j A_{ij}^k], \tag{1.54}$$

where (a, b) and (i, j) are the iterators over the same activation map A^k used for more clarity. In the further derivation of α_{ij}^{kc} the $ReLU$ function is omitted

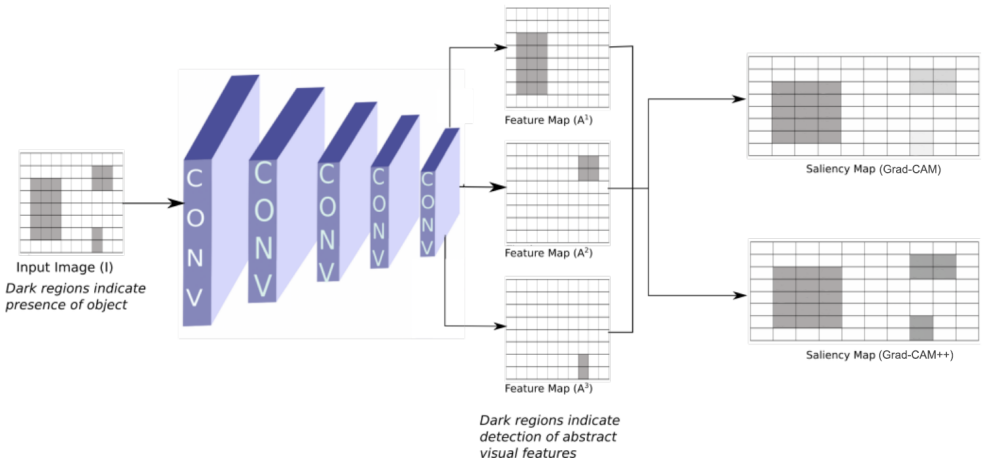


Figure 1.12: A hypothetical example of Grad-CAM's shortcoming that led to the introduction of Grad-CAM++ method. The shade of gray in the grids demonstrates the computed attributions of the corresponding pixels. **Top right:** Grad-CAM's saliency map fails to properly highlight all the important pixels. **Bottom right:** Grad-CAM++ correctly assigns the same importance to pixels from all the activation maps.

without the loss of generality as it only serves as a threshold. Taking partial derivative of both sides of (1.54) with respect to A_{ij}^k gives:

$$\frac{\partial Y^c}{\partial A_{ij}^k} = \sum_a \sum_b \alpha_{ab}^{kc} \frac{\partial Y^c}{\partial A_{ab}^k} + \sum_a \sum_b A_{ab}^k \{\alpha_{ij}^{kc} \frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2}\}. \tag{1.55}$$

Taking a further partial derivative with respect to A_{ij}^k gives:

$$\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2} = 2\alpha_{ij}^{kc} \frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k \{\alpha_{ij}^{kc} \frac{\partial^3 Y^c}{(\partial A_{ij}^k)^3}\}. \tag{1.56}$$

Rearranging the equation gives the solution to calculate the pixel-wise weights α_{ij}^{kc}:

$$\alpha_{ij}^{kc} = \frac{\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2}}{2\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k \{\frac{\partial^3 Y^c}{(\partial A_{ij}^k)^3}\}}. \tag{1.57}$$

The Grad-CAM++ activation map weights w_k^c are then given as:

$$w_k^c = \sum_i \sum_j [\frac{\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2}}{2\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k \{\frac{\partial^3 Y^c}{(\partial A_{ij}^k)^3}\}}] \, ReLU(\frac{\partial Y^c}{\partial A_{ij}^k}). \tag{1.58}$$

And the final class activation map for class c is again given by the linear combination of all activation maps followed by *ReLU* function [39]:

$$M^c = ReLU(\sum_k w_k^c A^k). \tag{1.59}$$

1.2.5.5 Guided Grad-CAM/Grad-CAM++

Both Grad-CAM and Grad-CAM++ methods introduced in the previous subsections compute attributions of spatial positions in activation maps of the last convolutional layer. The convolution and pooling layers gradually shrink the input representation in the CNN. This causes the saliency maps M^c computed by Grad-CAM or Grad-CAM++ to be of substantially lower resolution than the input and therefore the M^c has to be upscaled for visualisation of the explanation. The upscaled saliency map inherently lacks the ability to highlight fine-grained details that might be desirable in some contexts. To solve this, Selvaraju et al. [35] propose to combine the upscaled saliency map M^c with saliency map from another method that computes the attribution on a pixel level. Baseline proposition was to combine Grad-CAM/Grad-CAM++ explanation with the vanilla gradient and the input image:

$$\textit{Guided-}M^c = x \odot \frac{\partial Y^c}{\partial x} \odot \textit{upscale}(M^c), \tag{1.60}$$

where $\odot$ denote the element-wise product and *upscale* denotes an upscaling function to the corresponding dimensions. Alternatively a more complex method such as the Integrated gradients can be used to "guide" the Grad-CAM explanation.

1.2.6 Information bottleneck method

Information Bottlenecks for Attributions (IBA) [40] is a model-specific local explanation method for convolutional neural networks that aim to identify important spatial locations at a selected convolutional layer by injecting noise to it's input to mask out uninformative regions. Noise injection is a popular regularization technique used in deep learning to improve generalization and robustness where the noise is applied either to the input or to arbitrary layers' parameters during training [1]. Schulz et al. propose to leverage this technique for computing a saliency map by injecting noise into a layer in a trained CNN.

Let $f : \mathbb{R}^{w \times h \times d} \mapsto \mathbb{R}^{|C|}$ denote a convolutional neural network model, $X \in \mathbb{R}^{w \times h \times d}$ be random variable associated with the input and $Y \in \mathbb{R}^{|C|}$ be random variable associated to the output. In the standard setting the model f uses all information from X to predict Y. Information bottleneck introduces a new variable Z that limits the information from X that can be

used for Y. The goal is for the Z to maximize its mutual information with Y while minimizing its mutual information with X:

$$\max I(Y\,;Z) - \beta I(X\,;Z), \tag{1.61}$$

where $I(\cdot\,;\cdot)$ denotes the mutual information and β is a parameter that controls the amount of information that is allowed to flow through the network.

Let $f_l : \mathbb{R}^{w\times h\times d} \mapsto \mathbb{R}^{w_l\times h_l\times d_l}$ denote the output of lth convolutional layer in f after which the information bottleneck is to be injected. Schulz et al. suggest inserting the bottleneck roughly into the middle of the CNN model. Let $A = f_l(x)$ denote the output activation maps. The information bottleneck is inserted by adding noise to A. The model f is already trained and it is desirable that A preserves the same mean and variance even after the noise is added to it. Therefore the information bottleneck is applied as a linear interpolation between signal A and noise ϵ:

$$Z = \lambda A + (1-\lambda)\epsilon, \tag{1.62}$$

where $\epsilon \sim \mathcal{N}(\mu_A, \sigma_A^2)$, where μ_A, σ_A^2 denote the estimated means and variances of A, and $\lambda \in \mathbb{R}^{w_l\times h_l\times d_l}$ controls how much noise is injected to A, where $\lambda_{i,j,k} \in [0,1]$ at each spatial position (i,j,k). The mutual information $I(A;Z)$ is given as:

$$I(A\,;Z) = \mathbb{E}_A[D_{KL}[P(Z|A)||P(Z)]], \tag{1.63}$$

where $P(Z|A)$ and $P(Z)$ denote the respective probability distributions and $D_{KL}(\cdot||\cdot)$ denotes the Kullback-Leibler divergence. Computing $P(Z)$ requires evaluation of an intractable integral, therefore it is approximated instead. The authors suggest substituting $P(Z)$ by variational approximation $Q(Z) = \mathcal{N}(\mu_A, \sigma_A^2)$ which assumes all dimensions of Z to be independent and normally distributed. The approximation of $I(A\,;Z)$ denoted as $\mathcal{L}_I$ (information loss function) is then given as:

$$\mathcal{L}_I = \mathbb{E}_A[D_{KL}[P(Z|A)||Q(Z)]]. \tag{1.64}$$

The authors show in appendix D of [40] that $\mathcal{L}_1 \geq I(A\,;Z)$ and the approximation in (1.64) actually overestimates (1.63). As the goal of IBA is to minimize $\mathcal{L}_I$ while keeping the classification score high, the following optimization problem is defined:

$$\mathcal{L} = \mathcal{L}_{CE} + \beta\mathcal{L}_I, \tag{1.65}$$

where $\mathcal{L}_{CE}$ denotes the categorical cross-entropy loss and β is a hyperparameter that controls the relative importance between the two loss components. The optimized parameters are the $\lambda \in \mathbb{R}^{w_l\times h_l\times d_l}$ that control how much noise is injected to activation maps R at each spatial position. For each spatial position (i,j,k) it is required that $\lambda_{i,j,k} \in [0,1]$. To avoid any clipping of $\lambda_{i,j,k}$

during optimization a reparametrization is performed: $\lambda_{i,j,k} = S(\alpha_{i,j,k})$, where $S(x) = \frac{1}{1+e^{-x}}$ denotes the sigmoid function and α denotes the new parameters to be optimized trough $\mathcal{L}$. α is initialized such that $S(\alpha_{i,j,k})$ is close to 1 and the information bottleneck has initially close to no effect. To increase robustness and smoothness of the saliency map, λ is also blurred by convolution with a fixed Gaussian kernel with hyperparameter σ_s (standard deviation of the Gaussian kernel): $\lambda_{i,j,k} = blur_{\sigma_s}(S(\alpha_{i,j,k}))$. The parameters α are fitted by averaging 10 runs of Adam optimizer with 10 iterations and learning rate of 1. The attribution $IBA_{i,j}$ of spatial position (i, j) at activation maps R is then given by evaluating $D_{KL}[P(Z|A)||Q(Z)]$ per dimension and summing over the channel axis:

$$IBA_{i,j} = \sum_{k=1}^{d_l} D_{KL}[P(Z_{i,j,k}|A_{i,j,k})||Q(Z_{i,j,k})]. \tag{1.66}$$

Lastly, $IBA \in \mathbb{R}^{w_l,h_l}$ is upscaled to $\mathbb{R}^{w,h}$. The information bottlenecks for attribution method is summarized in Algorithm 6

Algorithm 6 Information bottlenecks for attribution

Require: Model f, class of interest c, input of interest x
Require: Layer index l, estimated means and variances μ_R, σ_R^2 of layer's l output activation maps
Require: Loss importance coefficient β, standard deviation σ_s of Gaussian kernel for bluring and upscaling function $upscale(\cdot)$

$\alpha \leftarrow \{0, 0, \ldots, 0\}$
for $i \in \{1, 2, \ldots, 10\}$ **do** ▷ α estimation
  $\alpha_i \leftarrow \arg\min_{\alpha_i} \mathcal{L}_{CE} + \beta\mathcal{L}_I$ ▷ by Adam optimizer
  $\alpha \leftarrow \alpha + \alpha_i \frac{1}{10}$ ▷ averaging over individual α_i
$IBA \leftarrow \{\}$
$a = f_l(x)$
for $i \in \{1, 2, \ldots, w_l\}$ **do**
  for $j \in \{1, 2, \ldots, h_l\}$ **do**
    $IBA_{i,j} \leftarrow \{0, 0, \ldots 0\}$
    for $k \in \{1, 2, \ldots, d_l\}$ **do**
      $\lambda_{i,j,k} = blur_{\sigma_s}(S(\alpha_{i,j,k}))$
      $z_{i,j,k} \leftarrow \lambda_{i,j,k}\, a_{i,j,k} + (1 - \lambda_{i,j,k})\, \epsilon_{i,j,k}$ ▷ $\epsilon \sim \mathcal{N}(\mu_A, \sigma_A^2)$
      $IBA_{i,j} \leftarrow IBA_{i,j} + D_{KL}[p(z_{i,j,k}|a_{i,j,k})||q(z_{i,j,k})]$
    $IBA \leftarrow IBA \cup IBA_{i,j}$
return $upscale(IBA)$

1.3 Faithfulness evaluation of explanation methods

Arguably the most pressing issue in the context of explainable AI is how to evaluate the faithfulness of the explanation methods to the model being explained. Faithfulness is here defined as the ability of the explanation method to provide explanations that accurately capture model's true decision making. In the context of saliency maps, high faithfulness would imply that the pixels highlighted as important were truly the important ones to the model.

The difficulty of faithfulness evaluation stems from the fact that there is no ground truth available. This is in contrast with evaluating machine learning models, where a subset of available data is usually held out and used only for testing.

Some explanation methods, such as SHAP and IG presented in sections 1.2.4.1 and 1.2.5.1 respectively try to bypass the need of faithfulness evaluation by constructing explanations that satisfy various desirable axioms such as sensitivity or completeness. As described in the corresponding sections, the exact solutions to these methods are computationally infeasible in practice and approximations are computed instead. It is unclear whether the axioms are satisfied even by the approximated solution. The faithfulness evaluation is therefore necessary even for the axiomatic explanations especially when they are to be compared with some non-axiomatic explanations.

1.3.1 An argument against visual evaluation

One of the first and most cited frameworks for evaluation of explanations was introduced by Doshi-Velez et al. in [41]. They propose three levels in which the explanation methods can be evaluated:

Application-grounded evaluation: Doshi-Velez et al. argue that the best way to evaluate explanations is with the domain experts and on the task for which the model being explained was trained. In the context of this thesis, this would include presenting the explanations to the doctors while they're performing diagnoses and letting them decide which explanation they deem as the most fitting.

Human-grounded evaluation: In situations when the experiments with domain experts on the real task are too challenging, Doshi-Velez et al. propose to conduct simpler experiments that can be performed by lay humans, instead of the domain experts. In the context of medical imaging it is not clear how the diagnosis task could be simplified to the lay human level.

Functionally-grounded evaluation: Doshi-Velez et al. argue that the functionally grounded evaluation represents the lowest level of explanations evaluation and should only be performed when human experiments are out of reach.

While application-grounded evaluations with domain experts certainly provide value, it is important to make a distinction on what they really evaluate.

Through visual inspection the doctors are able to see whether the presented explanation is consistent with their knowledge, however they can not evaluate whether the explanation is also faithful to the underlying model. The doctor can not decide whether an unreasonably looking explanation indicates an error in the reasoning of the underlying model or an error in the explanation method [42].

Moreover, the deep learning field and AI in general strive to surpass the human-level performance on specific tasks. In visual human-level evaluation the person performing the inspection will implicitly compare the explanation to their own knowledge in the given task. It is difficult to justify the assumption that the model should use the same features in its decision making as the person performing the visual inspection.

To make a clear distinction, the property of how aligned the explanation is with the person's prior knowledge is referred to as plausibility [43]. Any human-based visual evaluation of explanations is therefore evaluating plausibility. The provided arguments imply that it makes very little sense to evaluate plausibility before the faithfulness of the explanations is assessed. In contrast with the Doshi-Velez et al. framework, this puts the functionally-grounded evaluation of faithfulness above the human-based evaluations of plausibility.

Despite the presented arguments, the nonquantitative visual inspection is still the most common and often the only evaluation method used in the XAI publications. Nunes et al. show in their review of 190 studies published prior to 2017 that introduced a XAI related method or tool that only 21% contained "any form of evaluation, apart from toy examples" [44]. Similar study was published by Nauta et al. who showed that more recent papers from 2016–2020 contained some kind of quantitative evaluation in 58% of cases, whereas 33% of the papers still relied exclusively on anecdotal evidence [43]. This shows a clearly increasing incentive to quantitatively evaluate faithfulness of explanation methods.

1.3.2 Ablation test

The probably most common quantitative method to evaluate and compare faithfulness of saliency map explanation methods discussed by XAI researchers [13, 10, 45] is the ablation (perturbation) test. The intuition of this test is simple: pixels in the input images are gradually being hidden starting from those with the highest pixel importance as computed by the explanation methods and the changes of the model's outputs for the images with hidden pixels are observed. In theory, higher the true pixel's importance the bigger change in model's output should be observed. The results are often benchmarked against random explanations.

The ablation test is sensitive to how the the important pixels are hidden in the input image. Various choices and their impacts are described in

section 1.2.2.

1.3.2.1 Remove and retrain

The subsection 1.2.2 also discuss that by approximating missingness of certain pixels in the input images, the distribution of such images is different compared to the distribution on which the model being explained was trained. This complicates the evaluation of the ablation test as it may be unclear whether the change in the model's output was caused by hiding important part of the image or by the shift of the input image distribution.

Hooker et al. propose a solution in form of the remove and retrain (ROAR) framework [12]. Instead of observing changes in the model's outputs for the modified input images, they propose to retrain the same architecture on the corresponding modified inputs and observe the change in classification accuracy of the whole network on the modified inputs. The ROAR framework therefore requires for each explanation method and each ratio of hidden most important features to train the architecture of the model being explained on a newly constructed set of training images with hidden subsets of pixels.

Apart from the obvious limitations of having to perform a large number of retraining steps, Strumfels et al. in [13] point out another possible issue with this framework. Ultimately, the evaluation is performed on the retrained models rather than on the original model being explained. There are no guarantees that the retrained model will learn the same features as the original model except the ones it can't learn because they are hidden in the inputs. Even when the explanation method highlight the truly most important pixels for the original model, the retrained model could learn a similar relationship from the remaining pixels. Strumfels et al. further point out that "this problem fits into a larger discussion about whether or not your attribution method should be "true to the model" or "true to the data""[13].

1.3.2.2 Remove and debias

Remove and debias (ROAD) introduced by Rong et al. [15] is an ablation based framework that addresses the distribution shift problem without the necessity of retraining the original model.

They introduce a concept of Class information leakage which corresponds to the amount of discriminative information that is "leaked" through the binary mask which is used to mask out certain pixels in the image. The effect of Class information leakage is empirically demonstrated by training a classifier only on the binary masks that are generated as a certain ratio of the most important pixels identified through the integrated gradients method. For their task (CIFAR-10 classification), the model trained only on the binary masks was able to reach almost the same accuracy as the original model trained on the full images. In other words, they showed that during the ablation test

where the pixels are hidden using some simple baseline image, the location of the hidden pixels is providing additional information to the model and the change in model's output is therefore inaccurate approximation of the hidden pixel's importance to the model.

They further introduce a concept of Minimally revealing imputation condition which states that the pixels in the image should be hidden in a way that minimizes the Class information leakage. To satisfy this condition Rong et al. introduced a Noisy linear imputation method, introduced in subsection 1.2.2.2.

1.3.3 Model parameter randomization test

Model parameter randomization test is an explanation method evaluation technique introduced by Adebayo et al. that tries to evaluate how dependent is the explanation method on the explained model's parameters [46]. A common observation from nonquantitative visual evaluations of explanations is that some methods produce saliency maps that resemble an output of an edge detector. Edge detectors are designed to highlight high-frequency information in an image and therefore look only on the image itself [10]. This has lead researchers to wonder how much do the explanation methods relly on the input of interest instead of on the model being explained. Implicitly, this can not be evaluated through a nonquantitative visual examination of the explanations.

Adebayo et al. propose an evaluation technique that studies the dependency of an explanation method on the learned parameters of the model being explained. They propose to perform a cascading randomization of the model's parameters, where the model's learned parameters are gradually randomly reinitialized up to a certain layer, starting from the output layer all the way to the input layer. At each step, the partially randomized model's predictions for a given dataset are explained by the evaluated explanation method which yields a set of saliency maps for each input image. The corresponding saliency maps are compared by some similarity measure such as Spearman rank correlation. Higher similarity of such constructed saliency maps then imply lower dependency of the explanation method on the model's learned parameters. Adebayo et al. acknowledge that the architecture of the network itself has to some extent an effect on the derived represtantations and even for randomly initialized networks, those representations tend to be non-trivial. However, the insensibility of explanation method to model's learned parameters is still undesirable in most situations.

Chapter 2

Experimental Part

2.1 Implementation technologies

The experiments are written in **Python** language (version 3.7) [47], which is an object-oriented, high-level programming language. Python is a very popular language among the AI practitioners and it therefore provides a wide range of AI related packages and libraries. This is especially true in the field of explainable AI as majority of the reviewed papers that provided any kind of implementation or experiments used Python.

Jupyter notebooks [48] is a web-based interactive environment for running Python code and is also very popular in the AI community. It is mainly suitable for exploratory experiments where the code consist mainly from library or package calls and require many visualisations, which is the case of most experiments done in this thesis. Most of the attached code is therefore in the jupyter notebook format. However, it is not necessary to install any dependencies locally as all the notebooks were run in the **Google Colab**[1] platform. Google Colab is a cloud computing environment based on the Jupyter Notebooks that allows for experimental notebooks to be seamlessly shared among the practitioners.

NumPy [49] is a Python library used for handling data in forms of multi-dimensional numeric arrays. It provides a high-level interface as well as high efficiency due to its core part implementations being written in C/C++.

Matplotlib [50] is a standard plotting library for visualising data in both native Python data structures and NumPy arrays. Some more complex visualisations were done using Seaborn [51] which is a higher-level visualisation library build upon Matplotlib and Pandas [52].

PyTorch [53] is a machine learning framework that is (alongside TensorFlow) one of the two most popular Python libraries for designing and training deep neural networks. It provides tensor computing with strong acceleration

[1] `https://colab.research.google.com`

via GPUs and wide variety of neural network layers, optimizers, architectures and pre-trained models. The PyTorch implementation of the Vision transformer was obtained on GitHub[2]. Due to some CUDA memory management issues PyTorch was used in a downgraded 1.9 version.

OpenCV [54] is a computer vision library that provide optimized tools for image processing.

SciPy [55] is a scientific computation library built upon NumPy to provide complex operations on multi-dimensional arrays.

2.1.1 Explanation methods implementations

For many explanation methods the original implementations by their respective authors are available and PyTorch compatible, this includes LIME[3], SHAP[4], FullGrad[5] and IBA[6] explanation methods. When available and PyTorch compatible, the original implementations are preferably used. For CAM based methods, including GradCAM and GradCAM++, the PyTorch compatible implementations are provided by Gildenblat and contributors[7]. For Integrated gradients, the Captum implementation was used.

Captum [56] is an explainable AI library built on PyTorch. It provides unified interfaces and implementations of various explanation methods described in this thesis – LIME, SHAP, IG, GradCAM and also various other explanation methods such as DeepLIFT.

Surprisingly all the mentioned implementations also worked for explaining the vision transformer model. For attention visualisation the implementation[8] of paper [57] was used.

2.1.2 Hardware

All computations were run in the Google Cloud cloud computing environment with the highest subscription tier. GPU was utilized in all circumstances in which it could provided a noticeable acceleration (such as model training and inference). While Google Colab does not guarantee any specific computational resources, the following hardware specs were usually provided:

- 8 cores Intel Xeon 2.00 GHz CPU
- Nvidia Tesla V100 GPU with 16 GB memory
- 54 GB of RAM

[2] `https://github.com/rwightman/pytorch-image-models`
[3] `https://github.com/marcotcr/lime`
[4] `https://github.com/slundberg/shap`
[5] `https://github.com/idiap/fullgrad-saliency`
[6] `https://github.com/BioroboticsLab/IBA`
[7] `https://github.com/jacobgil/pytorch-grad-cam`
[8] `https://github.com/hila-chefer/Transformer-Explainability`

2.2 Dataset

The selected dataset for this thesis is the COVIDx dataset which contains chest X-ray images (CXR) for the task of covid-19 pneumonia detection. The COVIDx dataset was constructed for the COVID-Net project [58] and to the best of our knowledge it is the largest publicly available dataset in the given domain. It aggregates data from multiple sources which makes the dataset diverse as it contains 16 352 chest X-ray images (mixture posterior-anterior and anterior-posterior views) of 15 346 patients from more than 51 different countries [59]. Examples are given in figure 2.1

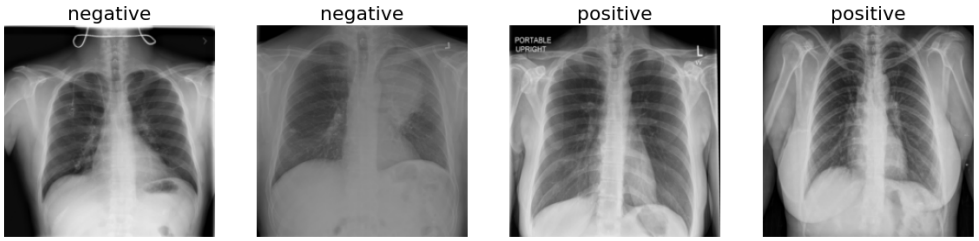


Figure 2.1: Examples of chest X-ray images from the COVIDx dataset

The COVIDx dataset is divided into the training and testing sets by the authors themselves, where the training dataset contains 15 952 images (13 794 negative and 2 158 positive cases) and the testing dataset contains 400 images (200 negative and 200 positive cases). The target classes in the training dataset are therefore unbalanced as is common in the medical datasets.

2.2.1 Preprocessing

The CXR images in the COVIDx dataset vary significantly in their size from the smallest 156×157 pixels image to the largest 3480×4248 pixels image. All CXR images were therefore resized (either downsampled or upsampled) to 224×224 pixels, which is the default input image size of the considered model architectures. As is common in the deep learning preprocessing pipelines, the images were then normalized to $[0, 1]$ pixel values.

Due to the nature of the X-ray acquirement process the resulting X-ray images tend to have low contrast which is undesirable for training of a CNN model. To enhance the images' contrast, the Contrast limited adaptive histogram equalization (CLAHE) is applied [60]. CLAHE is a popular contrast enhancing method often used in the domain of X-ray images. It divides the image into non-overlaping regions, performs histogram equalization for each region individually, clips the number of pixels with the same intensity to mitigate noise amplification and perform bilinear interpolation for the pixels near the region borders to obtain a coherent image.

The example results of this simple preprocessing pipeline are shown in figure 2.2. Note that in the following sections, the input images are visualised without the CLAHE applied.

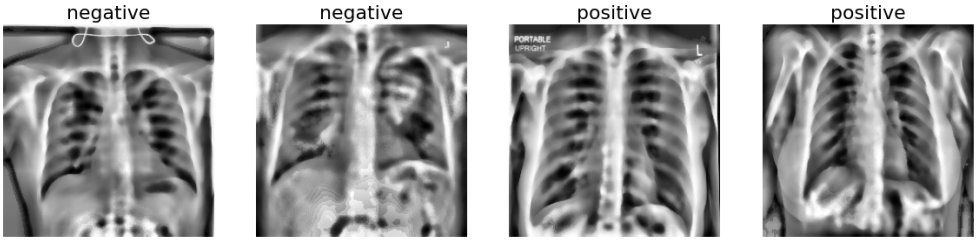


Figure 2.2: Examples of preprocessed chest X-ray images from the COVIDx dataset

2.3 Models

The following three pre-trained CNN models with two different architectures introduced in subsection 1.1.4.3 were fine-tuned on the COVIDx dataset:

ResNet50 model pre-trained on the ImageNet dataset [61] obtained from the PyTorch library.

DenseNet121 model pre-trained on the ImageNet dataset obtained from the PyTorch library.

CheXNet [62] which is a DenseNet121 first trained on the ImageNet and then fine-tuned on ChestX-ray14 dataset [63] which is a large dataset of over 100 thousand CXR images labeled with 14 different lung diseases. The CheXNet weights for the DenseNet121 architecture were obtained from GitHub [9]. To avoid overfitting the models were regularized dropout and early-stopping.

As described in the previous section, the COVIDx dataset is divided into training and testing sets. For the best model selection, the training dataset was further divided into training and validation sets in roughly 80/20 random split. The overall dataset was therefore split to 13 000 training images, 2952 validation images and 400 testing images.

As a loss function, the cross entropy loss was chosen. Given that the target classes are unbalanced, with prevailing negative class, the cross entropy loss was proportionately weighted to balance the classes. Adam with initial learning rate of 0.0001 was used as the optimization algorithm. This is a fairly standard setting for fine-tuning the given architectures.

All models were fine-tuned for 20 epochs which took between 2 and 3 hours of computational time on GPU for each model. The results are shown in Table 2.1.

[9] `https://github.com/zoogzog/chexnet`

CNN model	No class balancing		With class balancing	
	Train accuracy	Val accuracy	Train accuracy	Val accuracy
ResNet50	99.98%	97.69%	99.98%	**97.80%**
DenseNet121	99.00%	96.27%	97.91%	95.56%
CheXNet	**99.99%**	97.22%	99.98%	97.09%

Table 2.1: Fine-tuned models' accuracies on the training and validation datasets. Best results highlighted in bold.

The fine-tuned ResNet50 with class balancing achieved the highest validation accuracy of 97.80%. This model was further evaluated on the testing set, where it achieved **95.0%** accuracy.

Further experiments with more preprocessing techniques, optimizers and regularization choices might yield a model with higher accuracy, however for the purpose of evaluating explanations, the achieved accuracy of the model is sufficient.

2.4 Evaluation of explanations

The following subsections aim to evaluate the faithfulness of explanation methods on the best performing ResNet50 model discussed in the previous section, with the exception of section 2.4.4 that tries to evaluate the explanation methods in general through visualised self-attentions of the Vision transformer.

The explanations of the ResNet50 model were generated for selected 100 positive and 100 negative images from the test set, for which the model predicts the correct class with the highest confidence (highest relative differences between the class outputs). This is done to avoid explaining wrong or unconfident predictions of the model.

The explanations were generated using the implementations discussed in subsection 2.1.1 with the following settings:

LIME: Black imaged was used as the baseline, superpixels were generated by the default Quickshift segmentation algorithm, the local surrogate was trained using 16 000 samples and only the positive feature importances for the predicted class were considered.

SHAP: Partition explainer that uses the Owen values was used as suggested by Lundberg et al. [18] when explaining computer vision models. Telea inpaint was used to impute the hidden pixels and the Owen values were estimated using 1600 iterations. The negative feature importances for the predicted class were again discarded.

IG: The final Integrated gradients saliency map was given as an average of 5 IG explanations with the following baselines: black baseline, mean color baseline, blurred baseline (Gaussian blur with kernel size of 25×25 pixels

Explanation method	Mean computational time [s]
LIME	94.84
SHAP	51.36
Integrated Gradients	10.50
FullGrad	4.18
GradCAM	0.08
GradCAM++	0.09
Guided GradCAM	10.58
Guided GradCAM++	10.59
IBA	0.79

Table 2.2: Mean computational time for producing a single explanation of the ResNet50 by the particular explanation methods. The means were calculated over 200 test set images.

and standard deviation $\sigma = 10$), random baseline (averaged over 10 random baselines drawn from the uniform distribution), sampled baseline (averaged over 10 images sampled from the test data).

FullGrad: No specific setting was set nor is required for this method.

GradCAM and **GradCAM++**: The target layer up to which the gradients are backpropagated was chosen as the last convolutional layer in the model as suggested by the GradCAM's authors in [35].

Guided GradCAM and **Guided GradCAM++**: The Integrated gradient explanations were used to guide the GradCAM and GradCAM++. The guided GradCAM was therefore obtained as a pixel wise multiplication of the GradCAM saliency map and integrated gradients saliency map (analogously for the guided GradCAM++). This should yield better explanations than guiding GradCAM by vanilla gradients as was originally suggested in [35].

IBA: The information bottleneck is introduced roughly into the middle of the network (last convolutional layer in the second block) as suggested by the authors [40].

As is discussed in the following subsection 2.4.1 using only one baseline/imputation method to simulate missingness in the LIME and SHAP methods might not lead to optimal explanations. Table 2.2 shows that due to their model-agnostic nature LIME and SHAP have a significantly higher computational complexity compared to the other methods, which would be further magnified by averaging over multiple baseline/imputation methods.

Figures 2.4 and 2.5 show examples of explanations for various negative and positive test images. To avoid ambiguity the colormaps used for visualising the saliency maps are shown in figure 2.3.

In 2.4a the explanations for the given image show that three of the methods – LIME, SHAP and IBA highlight pixels corresponding to the right lung

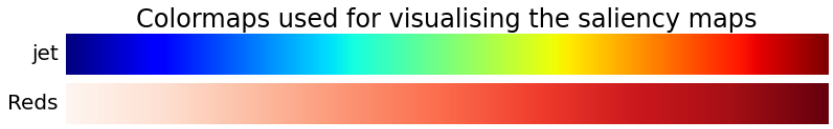


Figure 2.3: Colormaps used for visualising the saliency map explanations. Colors on the left indicate low pixel importance, colors on the right indicate high feature importance. **Top**: used when the saliency map is visualised over the image of interest. **Bottom**: used when the goal is to clearly compare different explanations without the need of visualising them over the image of interest.

of the patient, whereas the other methods highlight mainly the pixels on patient's left shoulder. If a visual evaluation was to be performed, the LIME, SHAP and IBA explanation methods would probably be preferred to other methods as their explanations are in this case more aligned with the intuition i.e. are more plausible. Looking at the other explanations in 2.4b and 2.4c the methods look fairly aligned at highlighting almost everything in the image except the lungs themselves which is in pure contradiction with the intuition. However, the visual inspection can not asses whether the explanation methods are highlighting the wrong pixels as important or whether the model has really learned such unintuitive features. Figures in 2.5 show that explanations of CXR images of positive patients highlight mainly the lungs area which is more inline with the intuition. This phenomenon was observed throughout the whole test set – explanations of negative images tend to highlight the corners of the images, whereas explanations of positive images ten to highlight more of the lungs area.

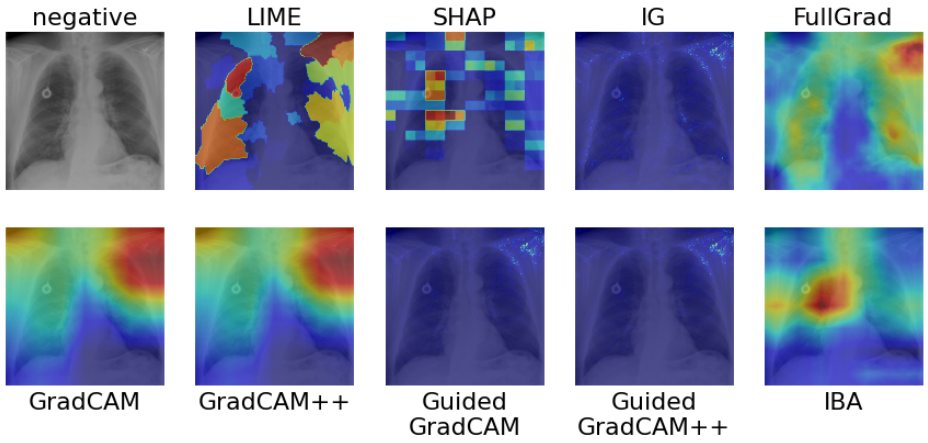
2.4.1 Baseline's influence on Integrated gradients

Section 1.2.2.1 put forward theoretical arguments for the importance of baseline image choice in both explanation and evaluation methods. This section tries to demonstrate it on the integrated gradients explanation method presented in section 1.2.5.1.

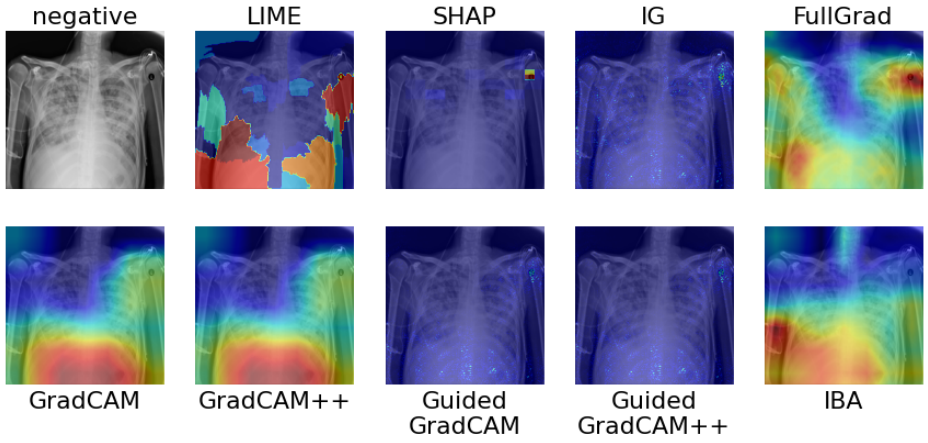
Recall, that integrated gradients compute the attributions by accumulating gradients on linear interpolation between the baseline image z and input image x and then multiply the accumulated gradients with difference of baseline and input values:

$$IG_i(x) = (x_i - z_i) \int_{\alpha=0}^{1} \frac{\partial f^c(z + \alpha\,(x - z))}{\partial x_i} d\alpha\,. \tag{2.1}$$

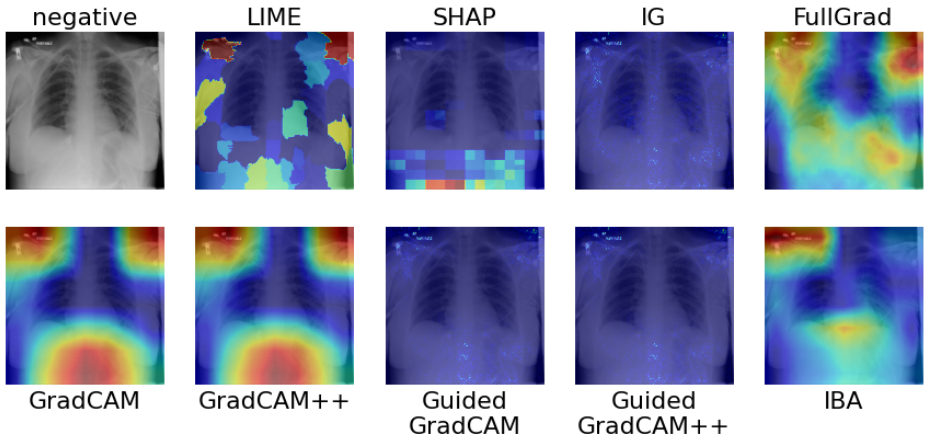
The authors of integrated gradients put very little emphasis on the choice of baseline image and suggest to simply use the black color baseline [29]. The



(a)

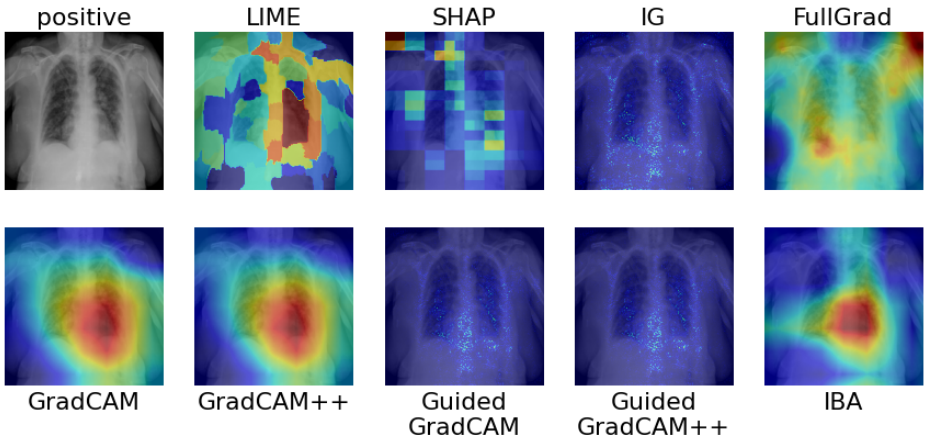


(b)

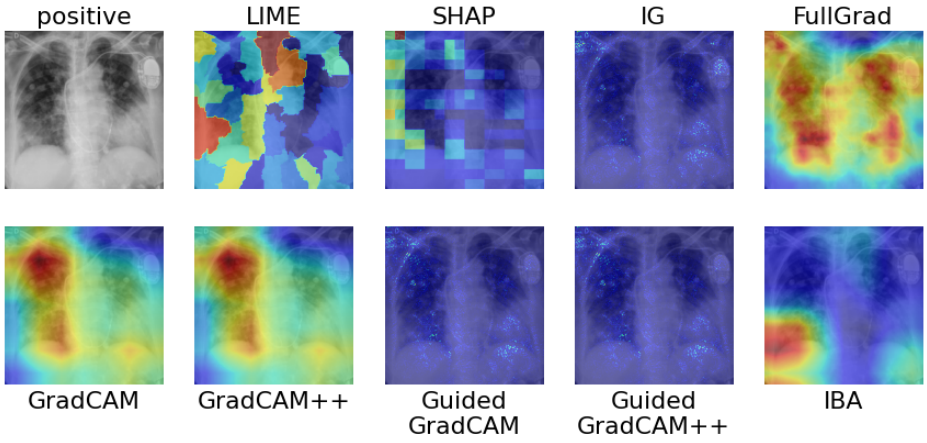


(c)

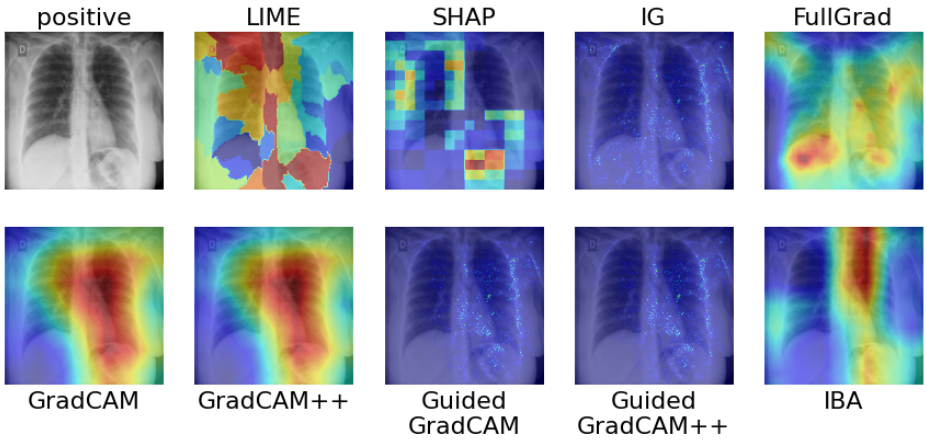
Figure 2.4: Explanations of ResNet50 model's prediction for three input images with negative class



(a)



(b)



(c)

Figure 2.5: Explanations of ResNet50 model's prediction for three input images with positive class

equation above implies that for the input pixels of the same color as baseline $(x_i - z_i) = 0$ the attribution $IG_i(x)$ will always be zero. By using an arbitrary baseline image an assumption is made that pixels for which $x_i = z_i$ should have zero importance. Such assumption is difficult to justify in our domain.

As discussed in [64], one way of minimizing the effect of zeroing out pixel importances for which $x_i = z_i$ is to average attributions over multiple baselines drawn from some distribution (or set) $\mathcal{D}$. $\mathcal{D}$ can be chosen as uniform or normal[10] distribution (random noise baselines), training data set (sampled baselines), set of multiple constant color baselines or a combination.

Figure 2.6 shows example explanations of 4 test data images generated by integrated gradients with black color baseline, mean color baseline, random baseline (drawn from uniform distribution and averaged over 10 iterations), sampled baseline (drawn from the test set and averaged over 10 iterations) and blurred baseline (Gaussian blurred with kernel size of 25×25 pixels and standard deviation $\sigma = 10$).

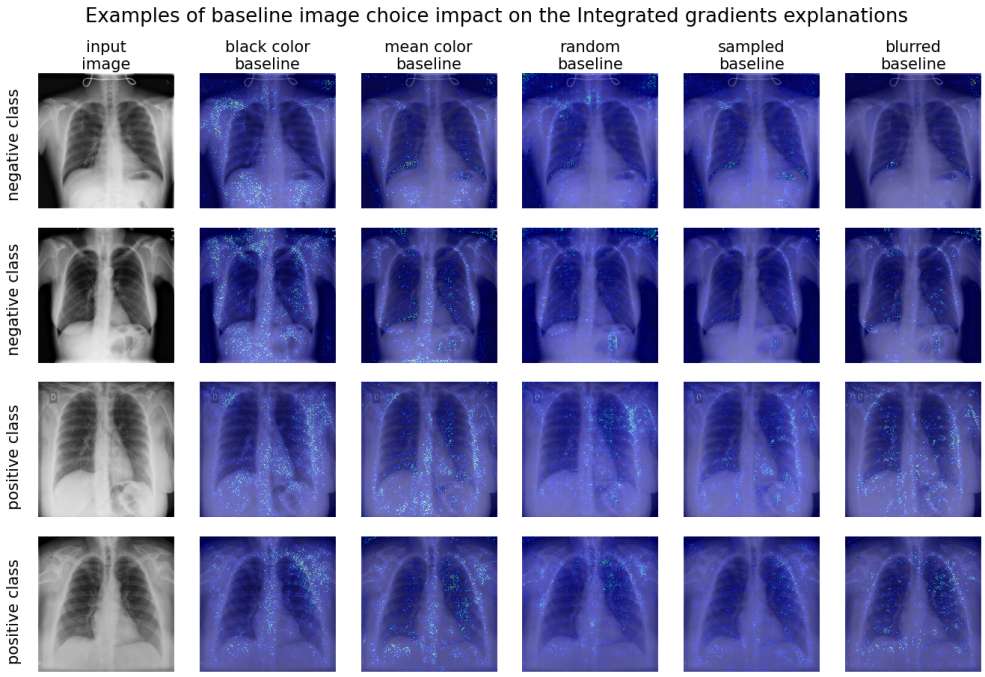


Figure 2.6: Examples of Integrated gradients explanations with different baselines.

[10]When $\mathcal{D}$ is chosen as $\mathcal{N}(0, \sigma^2 I)$ and the gradients are sampled only at the endpoint $\alpha = 0$, then the integrated gradients become almost equivalent to SmoothGrad explanation method introduced in [65].

Visual inspection of the displayed examples show notable differences between explanations that use different baselines. For example, the black color baseline explanations highlight more pixels on the patients body as important compared to the other methods, whereas the blurred baseline highlight more pixels on the edges. These observations are inline with the expectations. The other baseline explanations does not exhibit any noticable patterns.

To quantitatively measure the differences between different baseline explanations two metrics were used – Spearman rank correlation and overlap ratio of the 25% pixels with the highest importance in the explanations. Mean results over the whole test set are given as heatmaps in figure 2.7. Both rank correlations and overlap ratios are surprisingly low (note that mean overlap ratio of two random explanations would be 0.25). The most similar are explanations with mean and blurred baselines, which is expected as the mean baseline is essentially an extreme case of blurring.

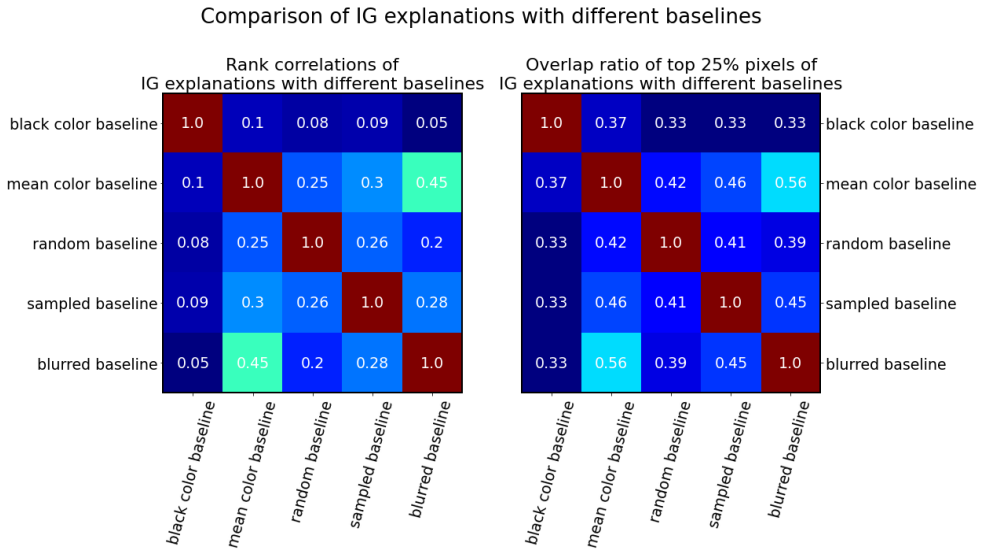


Figure 2.7: Comparison of Integrated gradients explanations with different baselines. **Right**: rank correlation heatmap. **Left**: overlap ratio of 25% most important pixels. Note that random saliency maps would have an overlap of 0.25.

In conclusion, the goal of this section was not to select a baseline choice that is superior to others. It was rather to highlight that baseline's impact on an explanation method can be substantial and more attention should be drawn towards its selection both in research and in practice.

2.4.2 Ablation test (remove and debias)

This section evaluates the explanations of the ResNet50 model through ablation test using the remove and debias framework introduced in section 1.3.2.2. In this experiment, the input image pixels are gradually being hidden starting from the most important ones as computed by the given explanation method. Hidding more relevant pixels in the image should result in bigger change in the model's prediction. The change in the model's prediction is measured as the relative change in the output of true class's logit after hidding a given ratio of pixels. The explanation methods are benchmarked against a random explanation to compare the effect of pixels being hidden in a random order.

Graphs in figure 2.8 show the results of this test. The graph on top visualises the relative change in the target's class logit output as a function of the ratio of top pixels ablated (hidden). Ablation ratios were evaluated in the range of $[0, 0.95]$ with 0.05 steps, where 0 corresponds to no pixels hidden and 1 would correspond to all pixels being hidden. The individual points correspond to individual evaluations. Random explanation is visualised as the gray dotted curve. In this test, lower curve imply better explanation method.

It is clear that all the considered explanation methods performed substantially better than the random explanation. To better visualise how each explanation method performed compared to the random explanation benchmark, the bottom graph shows the sum of differences between the evaluations for the random explanation and the corresponding explanation method. The summation is done up to 0.5 ablation ratio as past that the images become too degraded as more than half of the information is hidden.

The best performing explanation methods in this evaluation were Guided GradCAM and Guided GradCAM++ followed by SHAP.

2.4.3 Model parameter randomization test

This section evaluates the "sanity check" for explanation methods proposed by Adebayo et al. in [46] and described in section 1.3.3. The goal is to evaluate if the explanation method rely on the model's learned parameters by performing cascading randomization of the model's parameters and comparing the the similarity of explanations computed on these models.

The following ResNet50 models were used in this evaluation: the original fine-tuned ResNet50 model, models randomized up to (including) fully connected classification layer and each ResNet50's convolution block and lastly a completely randomized ResNet50 model. To speed up the evaluation, LIME and SHAP hyperparameters (number of samples and number of estimation steps) were lowered compared to the previous sections. An example of the cascading randomization's impact on the explanation methods is visualised in figure 2.9.

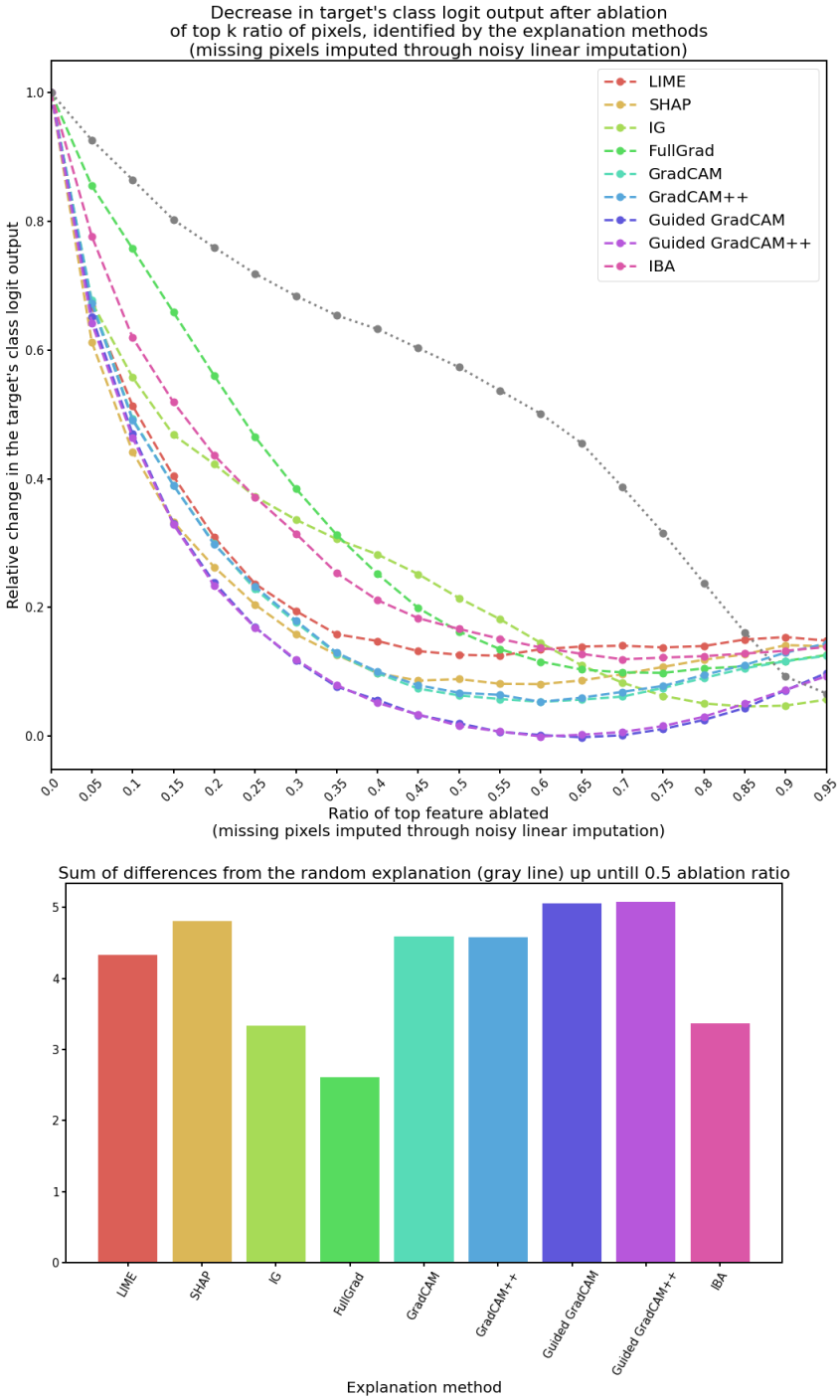


Figure 2.8: Ablation test (remove and debias) results. **Top**: relative change in the target's class logit output as a function of the ratio of top pixels ablated (lower is better). Gray dotted curve correspond to a random explanation. **Bottom**: Sum of differences of a corresponding method from the random benchmark (higher is better).

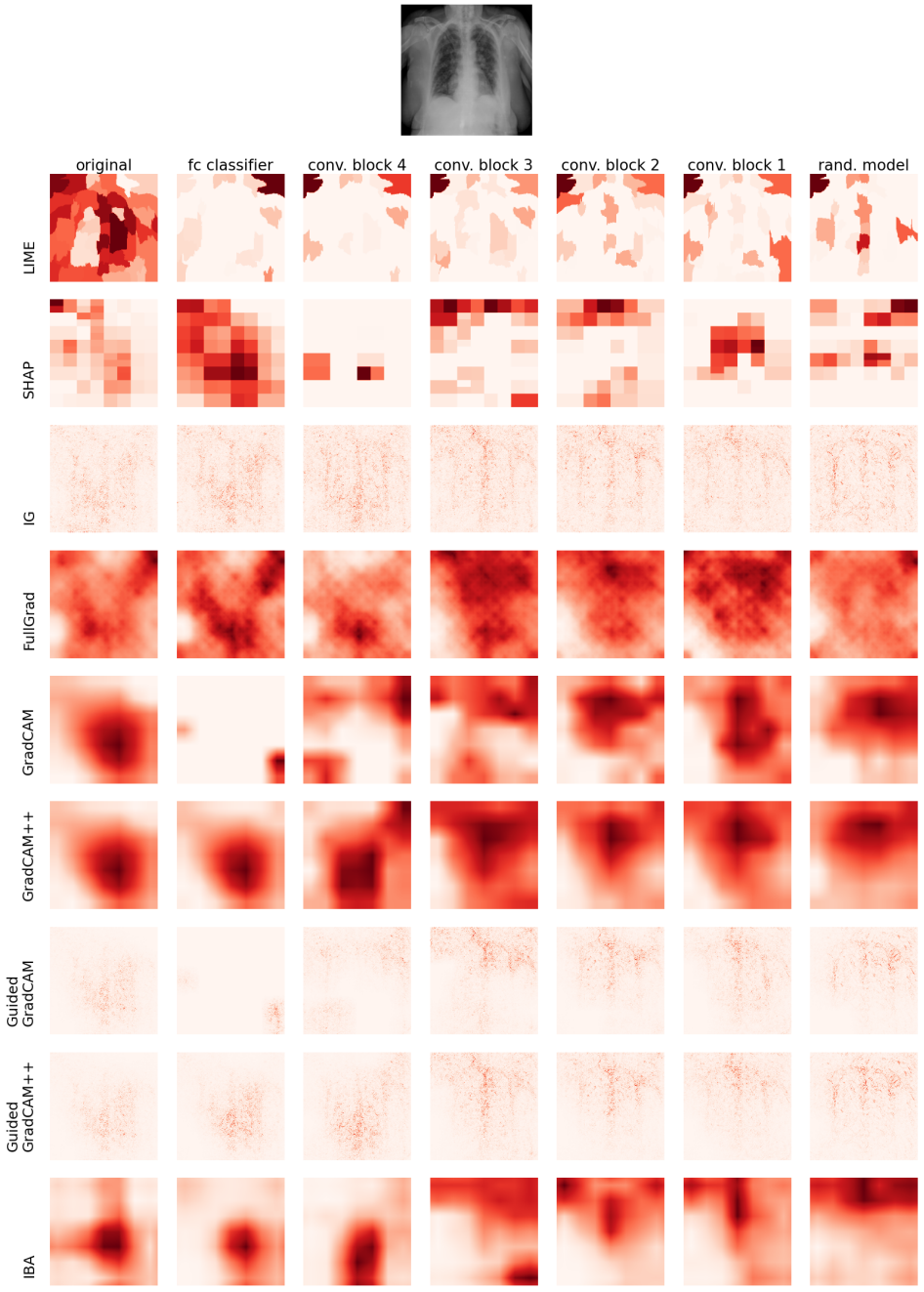


Figure 2.9: Example of the model's parameters randomization effect on the explanation saliency maps. Darker shade of red means higher importance. **Top**: image being explained (negative class). **Columns**: randomization of the Resnet50's parameters up to the given layer. **Rows**: Different explanation methods.

For the quantitative evaluation of explanation's saliency maps similarity the following two similarity metrics were used:

Spearman's rank correlation: For two saliency maps $M_i, M_j \in \mathbb{R}^{w \times h}$ let $R(M_i), R(M_j) \in \mathbb{R}^{w \times h}$ denote their ranked versions, where the pixel importance values are replaced by their respective ranks in the saliency map. The Spearman's rank correlation $r_S \in [-1, 1]$ between saliency maps M_i, M_j is then given as the Pearson correlation coefficient between their ranked version:

$$r_S = \rho_{R(M_i),R(M_j)} = \frac{cov(R(M_i), R(M_j))}{\sigma_{R(M_i)}\sigma_{R(M_j)}}, \tag{2.2}$$

where $cov(R(M_i), R(M_j))$ denotes the covariance and $\sigma_{R(M_i)}$, $\sigma_{R(M_j)}$ denote the corresponding standard deviations [66].

Top $(1-x)$ pixels' overlap ratio: This metric measures how much of the top $(1-x)$ ratio of pixels overlap for two saliency map. Let q_x denote the x quantile of saliency map M and let $B_{q_x}(M) \in \{0,1\}^{w \times h}$ denote a binary mask given at each spatial position (i, j) as:

$$(B_{q_x}(M))_{i,j} = \begin{cases} 1 & (M)_{i,j} \geq q_x \\ 0 & (M)_{i,j} < q_x \end{cases}.$$

Simply put, the binary mask $B_{q_x}(M)$ is non-zero at positions, where the corresponding feature importance in M is among the highest. The overlap ratio of two saliency maps M_i, M_j is then given as:

$$overlap_{q_x}(M_i, M_j) = \frac{|B_{q_x}(M_i) \cap B_{q_x}(M_j)|}{|B_{q_x}(M_i)|}, \tag{2.3}$$

where $|B_{q_x}(M_i) \cap B_{q_x}(M_j)|$ and $|B_{q_x}(M_i)|$ denote the number of non-zero elements in the corresponding binary masks. This metric is used as it may be often more desirable to compare saliency maps with respect to only the most important regions.

The results of this evaluation test are given in figures 2.10 (rank correlation), 2.11 (overlap ratio of top 25% pixels) and 2.12 (overlap ratio of top 10% pixels). At each evaluation step, the randomized model was explained by each explanation method and the explanations were compared to the explanations of the original model. As this test involves randomization of the model's parameters, the evaluations were performed twice. Both means and min/max bounds are visualised on the graphs. The benchmark (gray dotted line) was chosen as 0 for the rank correlation as more the explanation method relies on the learned model's parameters, the less it's explanations of a randomized model should correlate with the explanations of the original model. Similarly for the overlap ratios, the benchmarks are chosen as 0.25 for the overlap ratio of top 25% pixels as uncorrelated explanations would on average overlap by 0.25. Analogously for the overlap ratio of 10% pixels the bench mark was chosen as 0.1. Closer an explanation is to the benchmarks, the more

it depends on the learned parameters of the model and the better it passes this evaluation test. The bottom graphs show the sum of differences from the benchmarks for each explanation method.

The results show that the FullGrad and GradCAM++ methods performed significantly worse in this test compared to the other methods. Interestingly the best performing method was the Guided GradCAM. The wide disparity between the results of GradCAM and GradCAM++ methods is unintuitive as GradCAM++ was introduced as a generalization of GradCAM. This may be caused by the fact that in GradCAM++ additional parameters α^{kc} have to be approximated to compute the saliency map. The approximation of the additional parameters could be implemented in a way that makes it less dependent on the model's parameters and for example more dependent on the initialization.

The graphs on figures 2.11 and 2.12 show that when looking at the overlap ratio metric even the FullGrad and GradCAM++ explanations eventually get close to the benchmarks once a more significant part of the model is randomized.

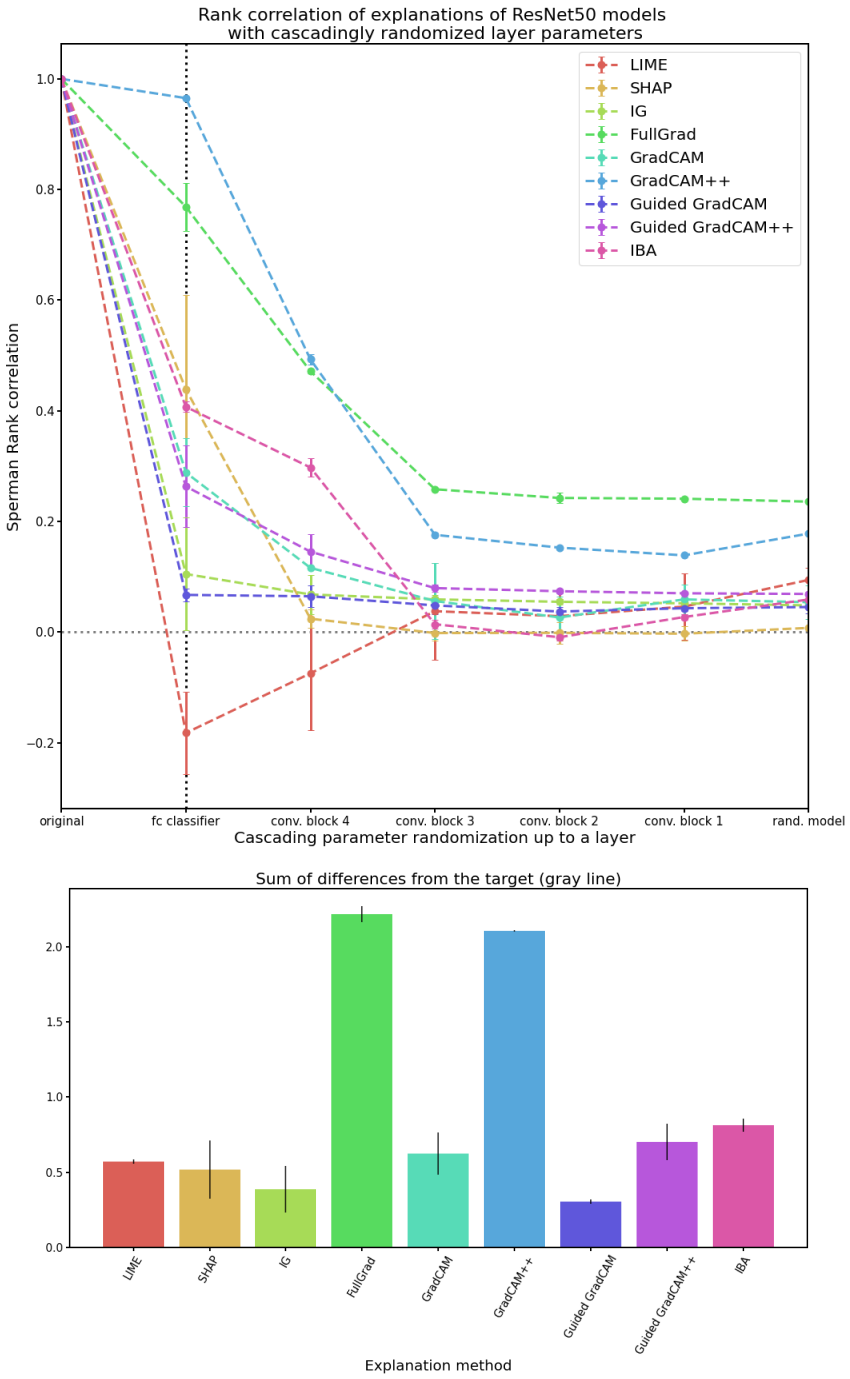


Figure 2.10: Cascading randomization of parameters evaluation (rank correlation). **Top**: rank correlation between original explanation and explanations of models randomized up to (including) layer on y axis. Gray dotted line represents the target rank correlation. **Bottom**: sum of differences from the target rank correlation (lower is better).

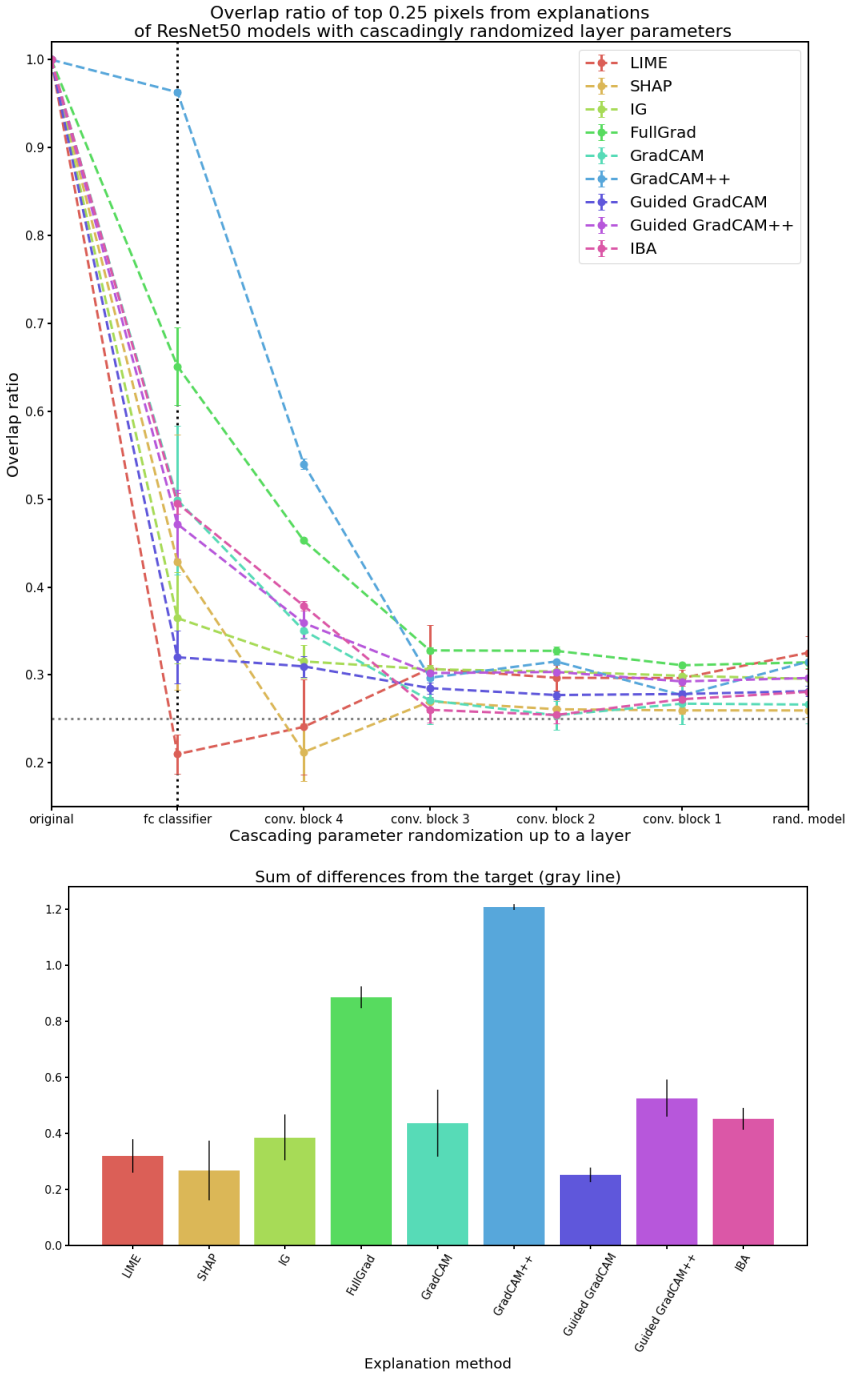


Figure 2.11: Cascading randomization of parameters evaluation. **Top**: Overlap ratio between 25% most important features from the original explanation and 25% most important features from explanations of models randomized up to (including) layer on y axis. **Bottom**: sum of differences from the target overlap ratio (lower is better).

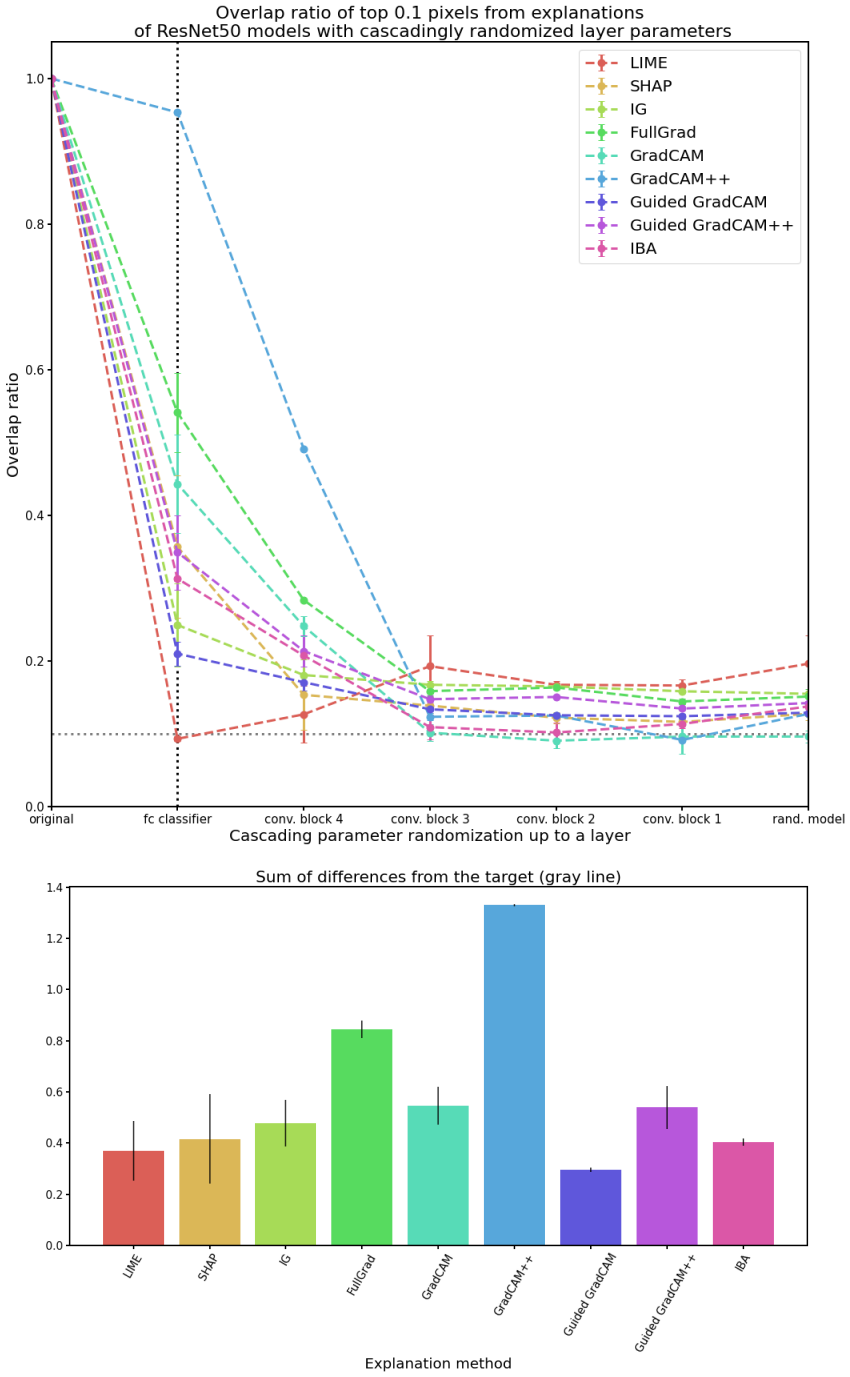


Figure 2.12: Cascading randomization of parameters evaluation. **Top**: Overlap ratio between 10% most important features from the original explanation and 10% most important features from explanations of models randomized up to (including) layer on y axis. **Bottom**: sum of differences from the target overlap ratio (lower is better).

2.4.4 Evaluation through attention

This section aims to evaluate the explanation methods by explaining a visual transformer and comparing the explanations to the visualised attentions obtained through the method described in [57].

The base version of the vision transformer pre-trained on the ImageNet was fine-tuned using the same settings as desribed in section 2.3 for training the CNN models. The resulting vision transformer achieved 90% accuracy on the test set. Similarly as in the previous evaluation, only the correctly classified test set images were explained.

The vision transformer model was explained by all the explanation methods described in this thesis with mostly the same settings as described in section 2.4. The only changes were done for the GradCAM and GradCAM++ methods as they require a target layer up to which the gradients are backpropagated. The authors of the used implementation specifically suggest using the first layer of the vision transformer's last block as the target layer for both GradCAM and GradCAM++. However, when using the suggested settings the GradCAM and GradCAM++ explanations often failed to recognize any important pixels and returned an empty saliency map. This issue was fixed by pushing the target layer closer towards the input layer.

Examples of the vision transformer explanations and visualised attentions are given in figure 2.13. Note that the visualised attentions tend to highlight only a small part of the CXR images.

The similarity of visualised attentions with the explanation methods on the test data are visualised in figures 2.14 and 2.15. Given that the visualised attentions tend to highlight only a small part of the images, the overlap ratio of top 10% pixels might provide more accurate comparison than the rank correlation. However, both rank correlation and overlap ratio metrics imply overall low similarity between the visualised attentions and the explanation methods.

The visualisation of transformer's attention has been considered as an implicitly faithful way of explaining transformers in various publications [8, 9]. While the intuition behind this is correct a recent survey by Khan et al. [67] highlights the fact that "the attention originating in each layer, gets intermixed in the subsequent layers in a complex manner, making it difficult to visualize the relative contribution of input tokens towards final predictions". Whether the use of the current attention visualisation methods explain the transformer in a faithful way seems to be an ongoing debate with some researchers arguing for [68] and others against [69].

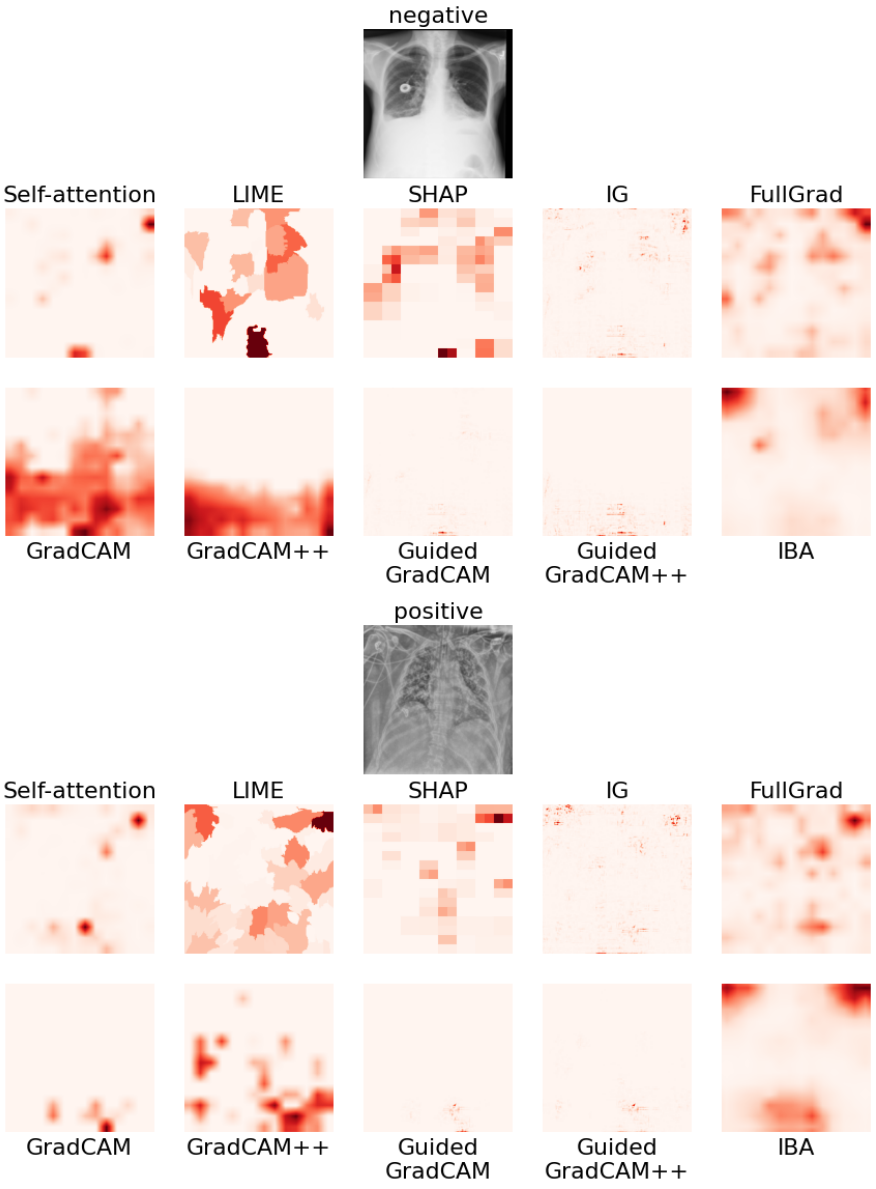


Figure 2.13: Representative examples of visualised self-attention and explanations for the given input images

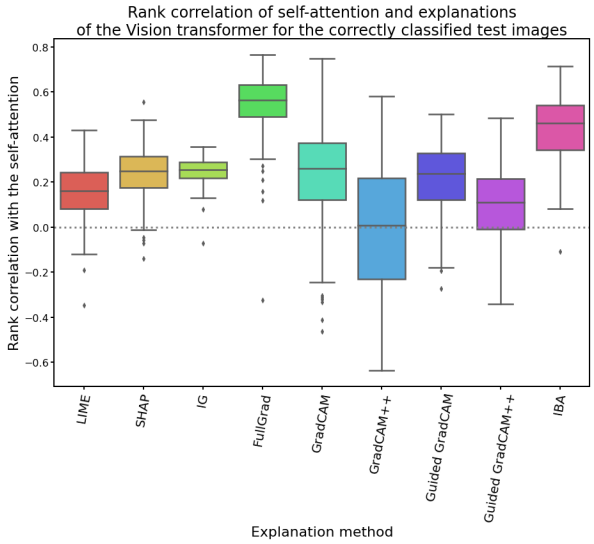


Figure 2.14: Rank correlation of the visualised attentions and the corresponding explanations of the correctly classified test set images.

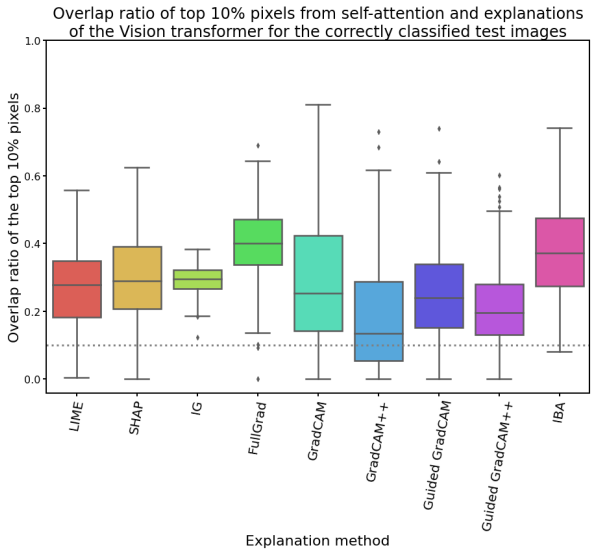


Figure 2.15: Overlap ratio of top 10% pixels of the visualised attentions and the corresponding explanations on the correctly classified test set images.

2.5 Discussion

This section aimed to quantitatively evaluate the explanation methods presented in the theoretical part of this thesis.

Table 2.2 show significantly higher computational complexity of LIME and SHAP methods due to their model-agnostic nature, especially compared to the GradCAM methods. This may discourage the use of model-agnostic explanation methods in some use-cases but not in such critical domains as medical imaging.

Visual inspection of explanations of the ResNet50 model, that achieved 95% accuracy on the test set showed, that according to the explanations the model looks at unintuitive features (pixels in the corners of the images) when when the negative class is predicted and more intuitive features (pixels in the lungs area) when the positive class is predicted.

The ablation test in section 2.4.2 showed that all of the considered explanation methods were significantly better at highlighting important pixels of the image compared to the random explanation benchmark. Section 2.4.3 showed the FullGrad and GradCAM++ to be less dependend on the model's learned parameters compared to the other explanation methods.

For the evaluation test done on the ResNet50 model, Guided GradCAM performed the best overall. Besides the Guided GradCAM, the next best performing methods were the model-agnostic LIME and SHAP which proves their value despite their high computational complexity. A key takeaway is that guiding the GradCAM and GradCAM++ explanations by the integrated gradients explanations yielded improvements in all the considered tests. On the other hand, while the GradCAM++ was designed to be an improved generalization of the GradCAM method, it performed worse or same as GradCAM.

While the evaluations on the ResNet50 provided some valuable information, they allowed for the explanation methods to be mostly just compared to each other and shown to perform significantly better than random explanations. This stems from the fact that the definitive assessment of explanation's faithfulness without having the ground truth is difficult.

To try tackle the problem of missing ground truth a vision transformer model was trained in section 2.4.4. The visualisation of the transformer's attention blocks has been considered as an implicitly faithful way of explaining the transformer model. The quantitative evaluation showed that the explanations provided by the considered explanation methods were substantially different from the visualised attentions. While this result could provide an argument against the discussed explanation methods, there is an ongoing argument whether the current techniques of visualising the attribution blocks are able to correctly assess the relative contribution of input tokens towards the prediction.

Conclusion

This thesis focused on the field of explainable artificial intelligence in the context of medical imaging. The research focused on identifying the suitable state-of-the-art explanation methods and methods to evaluate the faithfulness of their explanations. Special attention was also paid to the concept of simulating missingness in the inputs which is important for both explanation and evaluation methods.

The theoretical part gives a detailed description of the selected explanation methods: LIME, SHAP, Integrated gradients, Full-Gradients, GradCAM, GradCAM++, Guided GradCAM, Guided GradCAM++ and Information bottleneck attributions.

In the experimental part the ResNet50 architecture was trained on the COVIDx dataset for the task of covid-19 pneumonia detection from chest X-ray images and achieved 95% accuracy on the test dataset. The model was subsequently explained by all the described explanation methods. Visual inspection of the explanations showed, that the explanation methods highlight unintuitive pixels when the predicted class is negative and more intuitive pixels when the predicted class is positive. This is an insight to which a machine learning practitioner should pay attention before deploying such model in the critical environment of diagnostic imaging.

The explanations of the ResNet50 model were further quantitatively evaluated in terms of their faithfulness to the model through ablation and model randomization tests. The overall best performing explanation methods in these tests were the Guided GradCAM, SHAP and LIME. The evaluation tests also showed, that guiding the GradCAM and GradCAM++ by the integrated gradients method improved the explanations in all the measured metrics. On the other hand, the GradCAM++ did not yield any improvement over the simpler GradCAM method.

As an attempt to evaluate the explanation methods with respect to the ground truth explanations, a Vision transformer was trained and explained on the COVIDx dataset. The explanations were compared to the visualised

attention blocks of the transformer which should intuitively correspond to the ground truth explanations. Most of the considered explanation methods showed very low similarity with the visualised attentions. However, there is still an ongoing debate whether the current methods for visualising the attention blocks are truly faithful to the model.

The recent work in the field of explainable AI is shown to be promising. However, more focus should be drawn towards better quantitative faithfulness evaluation methods rather than on new explanation methods evaluated solely through a visual inspection. In the end, an insufficiently evaluated explanation method may be counterproductive in providing a false sense of understanding of the model which could have devastating consequences in critical fields such as the medical imaging.

Bibliography

[1] Goodfellow, I. J.; Bengio, Y.; et al. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[2] Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. 2014, doi:10.48550/ARXIV.1412.6980. Available from: https://arxiv.org/abs/1412.6980

[3] Albawi, S.; Abed Mohammed, T.; et al. Understanding of a Convolutional Neural Network. 08 2017, doi:10.1109/ICEngTechnol.2017.8308186.

[4] Sabour, S.; Frosst, N.; et al. Dynamic routing between capsules. *Advances in neural information processing systems*, volume 30, 2017.

[5] He, K.; Zhang, X.; et al. Deep Residual Learning for Image Recognition. 2015, doi:10.48550/ARXIV.1512.03385. Available from: https://arxiv.org/abs/1512.03385

[6] Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2014, doi:10.48550/ARXIV.1409.1556. Available from: https://arxiv.org/abs/1409.1556

[7] Huang, G.; Liu, Z.; et al. Densely Connected Convolutional Networks. 2016, doi:10.48550/ARXIV.1608.06993. Available from: https://arxiv.org/abs/1608.06993

[8] Vaswani, A.; Shazeer, N.; et al. Attention Is All You Need. 2017, doi:10.48550/ARXIV.1706.03762. Available from: https://arxiv.org/abs/1706.03762

[9] Dosovitskiy, A.; Beyer, L.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020, doi:10.48550/ARXIV.2010.11929. Available from: https://arxiv.org/abs/2010.11929

[10] Molnar, C. *Interpretable Machine Learning*. Second edition, 2022. Available from: https://christophm.github.io/interpretable-ml-book

[11] Menzies, P. Counterfactual Theories of Causation. In *Stanford Encyclopedia of Philosophy*, 2008.

[12] Hooker, S.; Erhan, D.; et al. A Benchmark for Interpretability Methods in Deep Neural Networks. 2018, doi:10.48550/ARXIV.1806.10758. Available from: https://arxiv.org/abs/1806.10758

[13] Sturmfels, P.; Lundberg, S.; et al. Visualizing the Impact of Feature Attribution Baselines. *Distill*, 2020, doi:10.23915/distill.00022, https://distill.pub/2020/attribution-baselines.

[14] Marlin, B. *Missing data problems in machine learning*. Dissertation thesis, 2008.

[15] Rong, Y.; Leemann, T.; et al. Evaluating Feature Attribution: An Information-Theoretic Perspective. 2022, doi: 10.48550/ARXIV.2202.00449. Available from: https://arxiv.org/abs/2202.00449

[16] Goodfellow, I. J.; Pouget-Abadie, J.; et al. Generative Adversarial Networks. 2014, doi:10.48550/ARXIV.1406.2661. Available from: https://arxiv.org/abs/1406.2661

[17] Telea, A. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, volume 9, no. 1, 2004: pp. 23–34.

[18] Lundberg, S. M.; Lee, S. A unified approach to interpreting model predictions. *CoRR*, volume abs/1705.07874, 2017, 1705.07874. Available from: http://arxiv.org/abs/1705.07874

[19] Ribeiro, M. T.; Singh, S.; et al. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA: Association for Computing Machinery, 2016, ISBN 9781450342322, p. 1135–1144, doi: 10.1145/2939672.2939778. Available from: https://doi.org/10.1145/2939672.2939778

[20] Vedaldi, A.; Soatto, S. Quick shift and kernel methods for mode seeking. In *European conference on computer vision*, Springer, 2008, pp. 705–718.

[21] Shapley, L. S. *17. A Value for n-Person Games*. Princeton University Press, 2016, pp. 307–318, doi:doi:10.1515/9781400881970-018. Available from: https://doi.org/10.1515/9781400881970-018

[22] Štrumbelj, E.; Kononenko, I. Explaining Prediction Models and Individual Predictions with Feature Contributions. *Knowl. Inf. Syst.*, volume 41, no. 3, dec 2014: p. 647–665, ISSN 0219-1377, doi:10.1007/s10115-013-0679-x. Available from: https://doi.org/10.1007/s10115-013-0679-x

[23] Hall, P. Building Explainable Machine Learning Systems: The Good, the Bad, and the Ugly. 2018, [h2o meetup NYC 2018]. Available from: https://www.youtube.com/watch?v=Q8rTrmqUQsU

[24] Knuth, D. E. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Boston: Addison-Wesley, third edition, 1997, ISBN 0201896842 9780201896848.

[25] Vidal-Puga, J.; Bergantiños, G. An implementation of the Owen value. *Games and Economic Behavior*, volume 44, no. 2, 2003: pp. 412–427.

[26] Shrikumar, A.; Greenside, P.; et al. Learning Important Features through Propagating Activation Differences. ICML'17, JMLR.org, 2017, p. 3145–3153.

[27] Frye, C.; Rowat, C.; et al. Asymmetric Shapley values: incorporating causal knowledge into model-agnostic explainability. 2019, doi:10.48550/ARXIV.1910.06358. Available from: https://arxiv.org/abs/1910.06358

[28] Heskes, T.; Sijben, E.; et al. Causal Shapley Values: Exploiting Causal Knowledge to Explain Individual Predictions of Complex Models. *CoRR*, volume abs/2011.01625, 2020, 2011.01625. Available from: https://arxiv.org/abs/2011.01625

[29] Sundararajan, M.; Taly, A.; et al. Axiomatic Attribution for Deep Networks. *CoRR*, volume abs/1703.01365, 2017, 1703.01365. Available from: http://arxiv.org/abs/1703.01365

[30] Aumann, R. J.; Shapley, L. S. Values of Non-Atomic Games. 1974.

[31] Simonyan, K.; Vedaldi, A.; et al. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[32] Böhle, M.; Eitel, F.; et al. Layer-Wise Relevance Propagation for Explaining Deep Neural Network Decisions in MRI-Based Alzheimer's Disease Classification. *Frontiers in Aging Neuroscience*, volume 11, 2019, ISSN 1663-4365, doi:10.3389/fnagi.2019.00194. Available from: https://www.frontiersin.org/article/10.3389/fnagi.2019.00194

[33] Friedman, E. J. Paths and consistency in additive cost sharing. *International Journal of Game Theory*, volume 32, no. 4, 2004: pp. 501–518.

[34] Srinivas, S. Gradient-based Methods for Deep Model Interpretability. Technical report, EPFL, 2021.

[35] Selvaraju, R. R.; Cogswell, M.; et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[36] Zhou, B.; Khosla, A.; et al. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[37] Zhou, B.; Khosla, A.; et al. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

[38] Bazzani, L.; Bergamo, A.; et al. Self-taught object localization with deep networks. In *2016 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 2016, pp. 1–9.

[39] Chattopadhyay, A.; Sarkar, A.; et al. Grad-CAM++: Generalized Gradient-based Visual Explanations for Deep Convolutional Networks. *CoRR*, volume abs/1710.11063, 2017, 1710.11063. Available from: http://arxiv.org/abs/1710.11063

[40] Schulz, K.; Sixt, L.; et al. Restricting the flow: Information bottlenecks for attribution. *arXiv preprint arXiv:2001.00396*, 2020.

[41] Doshi-Velez, F.; Kim, B. Towards A Rigorous Science of Interpretable Machine Learning. 2017, doi:10.48550/ARXIV.1702.08608. Available from: https://arxiv.org/abs/1702.08608

[42] Gilpin, L. H.; Bau, D.; et al. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, IEEE, 2018, pp. 80–89.

[43] Nauta, M.; Trienes, J.; et al. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. 2022, doi:10.48550/ARXIV.2201.08164. Available from: https://arxiv.org/abs/2201.08164

[44] Nunes, I.; Jannach, D. A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction*, volume 27, no. 3, 2017: pp. 393–444.

[45] Yeh, C.; Hsieh, C.; et al. How Sensitive are Sensitivity-Based Explanations? *CoRR*, volume abs/1901.09392, 2019, 1901.09392. Available from: http://arxiv.org/abs/1901.09392

[46] Adebayo, J.; Gilmer, J.; et al. Sanity Checks for Saliency Maps. *CoRR*, volume abs/1810.03292, 2018, 1810.03292. Available from: http://arxiv.org/abs/1810.03292

[47] Van Rossum, G.; Drake, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN 1441412697.

[48] Kluyver, T.; Ragan-Kelley, B.; et al. Jupyter Notebooks – a publishing format for reproducible computational workflows. 01 2016, ISBN 9781614996484, pp. 87–90.

[49] Oliphant, T. E. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006, ISBN 9781517300074.

[50] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, volume 9, no. 3, 2007: pp. 90–95, doi:10.1109/MCSE.2007.55.

[51] Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software*, volume 6, no. 60, 2021: p. 3021, doi:10.21105/joss.03021. Available from: https://doi.org/10.21105/joss.03021

[52] pandas development team, T. pandas-dev/pandas: Pandas. Zenodo, Feb. 2020, doi:10.5281/zenodo.3509134. Available from: https://doi.org/10.5281/zenodo.3509134

[53] Paszke, A.; Gross, S.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, edited by H. Wallach; H. Larochelle; A. Beygelzimer; F. d'Alché-Buc; E. Fox; R. Garnett, Curran Associates, Inc., 2019, pp. 8024–8035. Available from: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[54] Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[55] Virtanen, P.; Gommers, R.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, volume 17, 2020: pp. 261–272, doi:10.1038/s41592-019-0686-2.

[56] Kokhlikyan, N.; Miglani, V.; et al. Captum: A unified and generic model interpretability library for PyTorch. 2020, 2009.07896.

[57] Chefer, H.; Gur, S.; et al. Transformer Interpretability Beyond Attention Visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 782–791.

[58] Wang, L.; Lin, Z. Q.; et al. COVID-Net: a tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images. *Scientific Reports*, volume 10, no. 1, Nov 2020: p. 19549, ISSN 2045-2322, doi:10.1038/s41598-020-76550-z. Available from: https://doi.org/10.1038/s41598-020-76550-z

[59] Chodounsky, D. *Detection of COVID-19 in X-ray images using Neural Networks. Bachelor's thesis.* Master's thesis, 2021.

[60] Yadav, G.; Maheshwari, S.; et al. Contrast limited adaptive histogram equalization based enhancement for real time video system. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2014, pp. 2392–2397, doi:10.1109/ICACCI.2014.6968381.

[61] Deng, J.; Dong, W.; et al. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[62] Rajpurkar, P.; Irvin, J.; et al. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. 2017, doi:10.48550/ARXIV.1711.05225. Available from: https://arxiv.org/abs/1711.05225

[63] Wang, X.; Peng, Y.; et al. ChestX-ray14: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. 09 2017.

[64] Erion, G.; Janizek, J. D.; et al. Learning Explainable Models Using Attribution Priors. 2020. Available from: https://openreview.net/forum?id=rygPm64tDH

[65] Smilkov, D.; Thorat, N.; et al. SmoothGrad: removing noise by adding noise. *CoRR*, volume abs/1706.03825, 2017, 1706.03825. Available from: http://arxiv.org/abs/1706.03825

[66] *Spearman Rank Correlation Coefficient*. New York, NY: Springer New York, 2008, ISBN 978-0-387-32833-1, pp. 502–505, doi:10.1007/978-0-387-32833-1_379. Available from: https://doi.org/10.1007/978-0-387-32833-1_379

[67] Khan, S.; Naseer, M.; et al. Transformers in vision: A survey. *ACM Computing Surveys (CSUR)*, 2021.

[68] Wiegreffe, S.; Pinter, Y. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*, 2019.

[69] Jain, S.; Wallace, B. C. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

Appendix A

Acronyms

AI Artificial intelligence

ANN Artifical neural network

CAM Class activation mappings

CLAHE Contrast limited adaptive histogram equalization

CNN Convolutional neural network

CXR Chest X-ray

IBA Information bottlenecks attribution

IG Integrated gradients

LIME Local interpretable model-agnostic explanations

ROAD Remove and debias

ROAR Remove and retrain

SHAP Shapley additive explanations

VGG Visual geometry group

ViT Vision transformer

XAI Explainable artifical intelligence

Appendix B

Contents of CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... experiment notebooks
  - `train` ..... training and evaluation of the CNNs and ViT
  - `explain` ..... explaining the ResNet50 and ViT
  - `eval` ..... evaluation of the explanations
- `thesis` ..... the thesis directory
  - `thesis.zip` ..... LaTeX source codes of the thesis
  - `thesis.pdf` ..... the Diploma thesis in PDF format