Wrocław University
of Science and Technology

Faculty of Computer Science and Management
Field of study: **APPLIED COMPUTER SCIENCE**
Specialty: DATA SCIENCE

Master's Thesis

ATTRIBUTION METHODS IN INTERPRETABILITY OF DEEP CONVOLUTIONAL NEURAL NETWORKS

Kemal Erdem

keywords:
explainable artificial intelligence, convolutional neural networks, attribution methods, quantitative research, augmentation methods

short summary:
This thesis focuses on analyzing methods and measures used in Explainable Artificial Intelligence (XAI). Conducted experiments show the effect of augmentation on the interpretability of deep learning models by checking their structural similarity. The obtained results made it possible to compare the robustness of the tested XAI methods. They also provide an inside on the reliability of measures to measure XAI methods.

Supervisor	prof. dr hab. inż. Halina Kwaśnicka
	Title/degree/name and surname

The final evaluation of the thesis

Chairman of the Diploma Examination Committee	dr hab. inż. Paweł Myszkowski	5.5(A+)	.....
	Title/degree/name and surname	grade	signature

For the purposes of archival thesis qualified to:*
a) category A (perpetual files)
b) category BE 50 (subject to expertise after 50 years)
* Delete as appropriate

stamp of the faculty

Wrocław 2021

Abstract

In recent years, the field of Explainable Artificial Intelligence (XAI) has become increasingly important. Many new approaches to the interpretability of the deep learning models have been proposed. However, with the rise of new methods, it had become even harder to compare them and decide which one should be used for a particular solution. This thesis focuses on two important aspects of interpretability: robustness of the methods and reliability of the measures used to compare these methods. The first part checks the effect of real-world augmentations on the attribution methods by comparing similarity scores of those attributions. The results obtained during the experiments imply that even the most robust methods should not be used without a deeper understanding of potential flaws in producing the explanation. The second part concentrates on analyzing the popular measures and checks if they are reliable and ready to use for machine learning practitioners. The outcome from the experiments indicates that current measures are not reliable for comparing XAI methods. They are unable to meet the definition of a reliable measure. The measure of sensitivity can be used under some restricted conditions, but the scores returned by that measure have little value as we try to use them as a numerical measure.

Streszczenie

W ostatnich latach, dziedzina Wyjaśnialnej Sztucznej Inteligencji (XAI) stawała się coraz bardziej istotna. Zostało zaproponowane wiele nowych podejść do interpretowalności modeli głębokich. Jednak, wraz z pojawieniem się tych nowych metod, porównywanie oraz decyzja którą metodę użyć w konkretnym rozwiązaniu stało się trudniejsze. Ta praca skupia się na dwóch istotnych aspektach interpretowalności: odporności tych metod, oraz niezawodności miar służących do porównywania wspomnianych metod. Pierwsza część sprawdza efekty jakie na metodach atrybucyjnych robią spotykane w świecie rzeczywistym augmentacje danych. Rezultaty uzyskane podczas eksperymentów sugerują, że nawet najbardziej odporne metody, nie powinny być używane bez głębszego zrozumienia potencjalnych wad występujących podczas generowania atrybucji. Druga część skupia się na analizie popularnych miar i sprawdza, czy są one niezawodne oraz gotowe do użycia przez praktyków uczenia maszynowego. Wynik eksperymentów wskazuje, że obecne miary nie są niezawodne w porównywaniu metod XAI. Nie są w stanie spełnić definicji niezawodnej miary. Miara wrażliwości (ang. sensitivity) może być używana pod warunkiem spełnienia pewnych restrykcyjnych warunków, lecz wyniki zwracane przez nią, mają małą wartość, gdybyśmy chcieli traktować je jako miarę liczbową.

Contents

1 Introduction

In recent years, the field of machine learning (ML) has reached a wider audience with the introduction of digital assistants, self-driving cars, and many other innovations. This interest was caused by making machine learning models more accessible and easy to train. The increase in processing power allowed to train larger and better deep neural networks (DNN), which reached a point when they can outperform humans under certain conditions [13, 65, 54, 80]. Unfortunately, while drifting from the classic rule-based systems, we have lost the notion of explainability. The current models can reach up to trillion parameters [18] which are far beyond what a human can comprehend. These models are referred to as black-boxes, and they are not providing insides to how the decision was made. They are in opposition to more transparent techniques like decision trees or association rules, where the decision process is usually much clearer, but the accuracy is lower. With this side of the model complexity and the issue of understanding the decision, people tend not to trust the models when they do not provide the explanation [14].

1.1 Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) is one of the youngest and one of the fastest developing branches of the field. The objective of the XAI method is to provide an explanation for the deep learning model that is understandable by humans. This is especially important in safety-critical domains like healthcare or security. The methods presented in the literature over the years often promise that they will provide a clear answer to the question of how the model made its decision. With that promise and relative ease of use (most of the popular methods are available in popular XAI libraries), they are often used by non-professionals when designing the models. Authors of the most popular XAI methods are releasing the papers with a specially selected set of examples, which do not cover a wide range of situations where a specific method can be used.

XAI methods have been used for explaining models used in healthcare [55, 60, 33, 82, 77, 47, 41], genes analysis [43], hate speech detection [35], drug discovery [29], Covid-19 X-Ray analysis [50, 4] and fraud detection [62], Parkinson's Disease recognition [53], and other areas [35, 7].

The range of measures that can be used to compare XAI methods is very limited. Only two measures called *Infidelity* and *Sensitivity*[83] were implemented in the most popular XAI library [34]. Those measures have a solid theoretical background and are used to compare which XAI method works better.

1.2 Goal of the study

This thesis is going to investigate XAI methods in terms of their robustness to real-world augmentation of the input data. Additionally, the measures used to compare these methods are also part of the experiments. The goal is to verify if the measure can be used to compare the quality of XAI methods. The study is focused on the interpretability of image-recognition tasks and the attribution of the Convolutional Neural Networks (CNNs) [37, 38].

Methods tested in this study are limited to attribution methods. They were selected base on popularity and availability in popular XAI frameworks with the assumption that the most available methods have the biggest impact on the field of machine learning. To check the effect of applying augmentations on the XAI methods, the measure of similarity is going to be used. This measure is going to check how much the

attribution provided by the method changes if the augmentation does not affect the models' performance.To test the measures, a set of additional experiments is going to be performed. This should help to decide wherever the measure is reliable to compare two XAI methods.

At the end of the study, we should be able to answer which XAI methods are the most robust and what their limitations are. We also should be able to explain the effect of real-world augmentation on the performance of such methods as well as describe the reason why some methods perform better than others. Furthermore, we should have an understanding of the reliability of the measures used to compare XAI methods and if they should be used in practice to compare methods. If yes, then what are the constraints, and how to interpret the results of such comparison.

1.3 Outline

In the following chapter, the theoretical background for this study is going to be presented. The chapter focuses on exploring the idea of machine learning and the backpropagation method, as well as introducing the idea of CNNs. It also contains the definitions of interpretability and explainability, and a brief description of the idea of attribution methods and types of measures used for research. The third chapter contains a detailed explanation of every XAI method used in the study and the additional technique used to improve the quality of the attribution methods. Next, in chapter 4, the measures of XAI methods are presented: main two measures that are the subject of experiments, an additional measure used to compare similarities of the attribution, and one measure of XAI methods from the literature that is worth discussing but because of its computational complexity, cannot be reproduced easily. The fifth chapter is a description of the experiments performed in this thesis. Experiments are divided into two parts. The first part is related to the robustness of the XAI methods and is called *"Dont́ Augment Me"* (see section 5.4). The second part is focused on the measures used to compare XAI methods and their reliability, and this part is called *"Can I Rely On You?"* (see section 5.5). To execute these experiments, the preparation of the environment is required. That is why the first part of chapter 5 contains a description of the datasets, training procedure for models, and the description of the augmentations used in the experiments. Finally, chapter 6 contains all the results of the experiments. The thesis is closed with chapter 7, where this study and its effects are discussed. In addition to the thesis, appendix section A contains instructions on how to reproduce the experiments. Supplementary materials are also in the appendix (section B), including additional charts and detailed results.

2 Theoretical Background

2.1 Machine Learning

Machine learning is described as a computer system that can increase its performance in performing a specific task by learning from experiences of similar tasks it has performed in the past (Carbonell, Michalski, and Mitchell [10]). This definition was formally defined by T. Mitchell in 1997 [49], and it is currently the most common definition of machine learning:

Definition 2.1 *Machine Learning A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.*

Machine learning approaches are often divided into three categories: *Supervised learning*, *Unsupervised learning*, and *Reinforcement learning* [8]. Supervised learning uses labeled training data to approximate the function that maps the input data into correct labels. Unsupervised learning does not use the labeled training data. Instead, it uses the unlabeled data to create a representation of that data, which can then be used for tasks like clustering or as an input for other types of networks. Lastly, reinforcement learning learns decisions by interacting with a dynamic environment. Correct decisions are rewarded, which should encourage the system to make similar decisions.

This thesis focuses on supervised learning, and to be more specific, the classification task. Classification requires the model to provide a single class based on the input data, whereas regression tries to find the relation between the input and the output. Supervised learning is used in many fields, especially in the image classification task, which base on the set of input features, should infer the correct class of the object on the image.

2.2 Neural Networks

Artificial Neural Networks are systems that are based on the network of neurons of the animal brain [48]. These neurons are called *artificial neurons* and are referred to as *nodes*. They are interconnected and organized in *layers* to eventually form a neural network (see Fig. 2.1). Each node (represented by a circle) receives the information from the nodes in the previous layer and transmits the new information to the next layer. Input nodes are the exception because they already have the information. New information transmitted by the node is a value from its activation function after receiving all the inputs. The connection between two nodes modifies the output from the lower-layer node before entering the higher-layer node. Most of the networks have an additional bias value for each of the nodes, which can be seen as a regularization parameter. Neural networks have at least three layers: one input layer, one output layer, and at least one hidden layer. The output of the neural network can be defined as:

$$z_j^l = \sigma\left(\sum_i^N (x_i^{l-1} * w_{i,j}^l) + b_j^l\right) \tag{2.1}$$

Where:

- l is a current layer
- z_j^l is an output from the jth neuron in layer l
- x_i^{l-1} is an output from the ith neuron on the $l-1$ layer (previous layer)
- $w_{i,j}^l$ is a weight for the pair of a current neuron at jth position and ith neuron from the previous layer
- N is a number of neurons in the $l-1$ layer
- b_j^l is a bias of jth neuron in layer l
- σ is an activation function

The goal of a neural network is then finding the value of the weights w_{ij}^l and the biases b_j^l to generate the correct output. These values are optimized in the process called backpropagation.

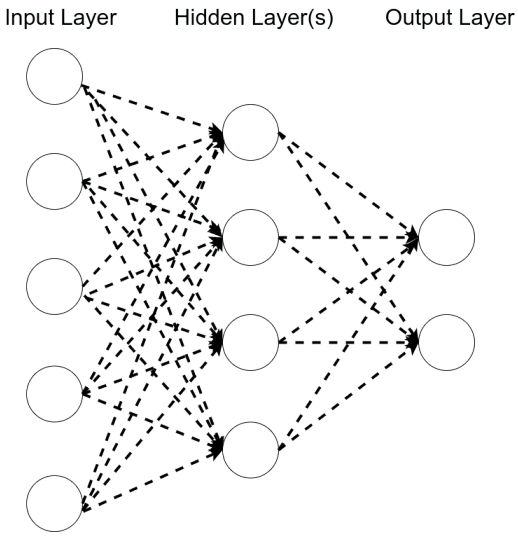


Figure 2.1: Basic artificial neural network structure.

2.2.1 Backpropagation

Backpropagation is an algorithm that calculates the changes to the weights of the network that are then used to update those weights. The process was first described by Paul Werbos in 1974 [81]. Even if the name "backpropagation" refers only to gradient computation, it is usually used to describe the whole process of updating the weights. This process consists of three parts. The first part is a feed-forward computation of the input (as defined above), then the error is calculated using the loss function, and lastly, based on the loss function with respect to the weights, the backpropagation computer the gradient used to update the weights.

After the forward propagation, the error is calculated using a loss function $L(y, \hat{y})$ (where $\hat{y}$ is a predicted output and y is a target output). Usually, the output of the network is a vector $\hat{y} \in \mathbb{R}^n$, and the loss function is a function that translates the difference between y and y into a real number, representing the "cost". One of the simplest loss functions is the Euclidean distance:

$$L(y, \hat{y}) = \frac{1}{2}||\hat{y} - y||^2 \tag{2.2}$$

In order to update weights, the gradients have to be calculated with respect to these weights:

$$\frac{\partial L}{\partial W_l} \tag{2.3}$$

Where W_l is a weight matrix at layer l. Computation of that gradient is difficult, but fortunately, the chain rule can be used. It is easier to explain the calculation when the forward function is split into parts (where CE stands for *Cross-Entropy* loss function defined in equation 2.9, often used for classification).

$$z_1 = W_1 x + b_1 \tag{2.4}$$

$$h = \text{ReLU}(z_1) \tag{2.5}$$

$$z_2 = W_2 h + b_2 \tag{2.6}$$

$$\hat{y} = \text{sorfmax}(z_2) \tag{2.7}$$

$$L = \text{CE}(y, \hat{y}) \tag{2.8}$$

$$\text{CE}(y, \hat{y}) = -(ylog(\hat{y}) + (1-y)log(1-\hat{y})) \tag{2.9}$$

Gradient for W_1:

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial z_2}\frac{\partial z_1}{\partial W_1} = \left((\hat{y}-y)^T z_2 \circ sgn(h)\right)^T x^T \tag{2.10}$$

And for b_1 :

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_2}\frac{\partial z_1}{\partial b_1} = \left((\hat{y}-y)^T z_2 \circ sgn(h)\right)^T \tag{2.11}$$

Where *sgn* is a sign function which is a derivative of the ReLU activation function (see eq. 2.13). With the value of the gradients, weights and biases can be updated.

ReLU

As described in equation 2.1 and then later in 2.5, the output is calculated with the use of the activation function. Usually that function is a ReLU function [24]. ReLU stands for *Rectified Linear Unit function* and is defined as:

$$\text{ReLU}(x) = max(x, 0) \tag{2.12}$$

and visualized in Figure 2.2. Because ReLU returns 0 every time the value of x is less than 0, it can deactivate neurons, and therefore create sparser networks. There are two issues with the ReLU function. The first one is that it can create "dead neurons" if all of the inputs of a neuron are less than zero. The second one is that it is not fully differentiable (at the $(0, 1)$ point). The first one was solved by Mass et al. [46] by replacing a zero-valued tail of the ReLU with a slightly negative value αx (where α is a small positive number). Derivative of the ReLU function is defined as follow:

$$\text{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} = sgn(\text{ReLU}(x)) \tag{2.13}$$

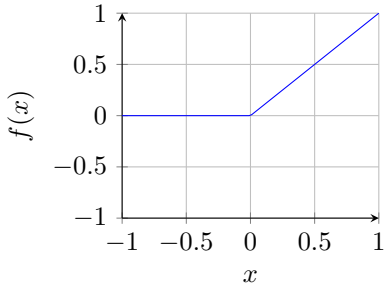


Figure 2.2: ReLU(x) where $x \in < -1, 1 >$

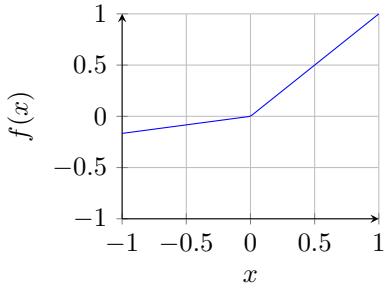


Figure 2.3: Leaky ReLU(x) where $x \in < -1, 1 >$

2.2.2 Convolutional Neural Networks

Convolutional Neural Network (CNN) [37, 38] are the type of Neural Networks with at least one convolutional layer. The first convolutional neural networks were LeNet [39] and AlexNet [36], using only a few convolutional layers. CNNs are assuming that the input data consists of hierarchical patterns which can be used instead of relying on individual features and creating a connection between every single feature and neuron. This approach helps to reduce the number of parameters required by the network.

Convolutional Layer

The convolutional layer uses the idea of filters/kernels with a given size and depth. A layer like that produces the output called a feature map, which is a combination of all the outputs from the kernels. As mentioned, each kernel is defined by the size (which corresponds to its width and height), depths (usually the depth of the input), and additional parameters like padding, stride, or dilatation. The name "convolutional"

comes from the mathematical operation of convolution, which is denoted by $f * g$, and the output of that operation is a function that described how one function modifies the other. It can be defined as:

$$g(m,n) = (h * f)(m,n) = \sum_u \sum_v h(u,v) f(m-u, n-v) \tag{2.14}$$

Where f is the input image, h is the kernel, m and n are the corresponding row and column in the feature map, and u and v range over all legal subscripts for $h(u,v)$ and $f(m-u, n-v)$ [30]. An example of the convolution operation for the $m = 0, n = 3$ is shown in Figure 2.4.

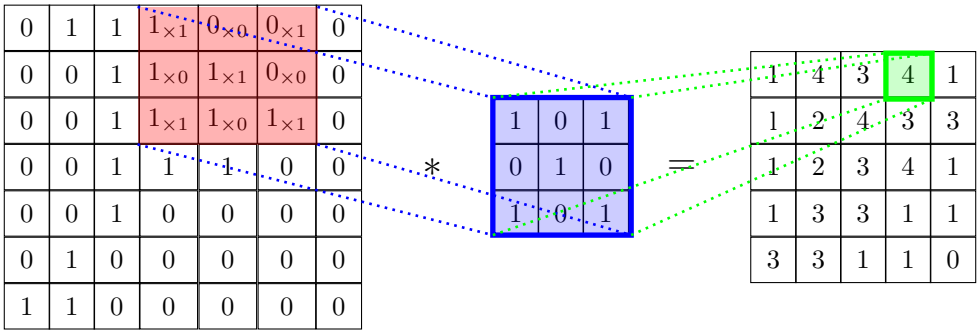
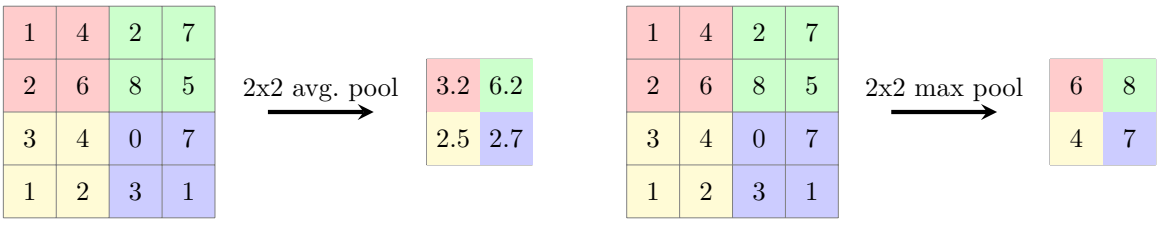


Figure 2.4: Example of the convolutional operation

The standard convolutional layer uses multiple kernels at once, and every one of them has size × size learnable parameters. Putting that in a context where a fully connected neural network requires a weight for every pixel of the input image times the number of neurons in the hidden layer, a convolutional layer is a huge decrease in the total number of parameters. A kernel with a size of 3 has 9 learnable parameters. Even with having 50 kernels in the convolutional layer, there are only 450 learnable parameters in total (for any input image size). For a 32×32 image, a fully connected layer with 10 neurons would require 10240 parameters. It is worth noticing that kernels do not have to be square. The square is computationally efficient, but there were tries to use different shapes [45, 22, 69].

Pooling Layer

CNNs have an additional type of layer called the *Pooling Layer*. There are two types of pooling layers called: *Max Pooling* and *Average Pooling*. Each pooling layer is defined by its size (called the window size), and the result is based only on pixels from that window. The value returned by the max pooling layer is the maximum value from the window. The average pooling layer returns the average from all values from that window.



(a) Average Pooling (b) Max Pooling

Figure 2.5: Example of pooling layers results applied to the same 4×4 feature map. Colors indicate the area from which the values are pulled to produce the result.

2.3 Interpretability

There is no one formal definition of interpretability and/or explainability in the context of machine learning, and often it is used interchangeably [23]. Arrieta et al. [6] distinguish between the two and define them as:

Definition 2.2 (Interpretability) *Passive characteristic on a model refers to the level of understanding of the models' internal decision process for a human observer.*

Definition 2.3 (Explainability) *Active characteristic of a model, associated with the notion of explanation of the action or procedure taken by the model with the intent of clarifying its internal decision process.*

The name of the field Explainable Artificial Intelligence (XAI) refers to the characteristic of a model, but any representation presented to the human (like input attribution) refers to the interpretability of the model. This thesis focuses on the interpretability of CNNs and uses the taxonomy of interpretability as defined in Figure 2.6 [42].

2.3.1 Taxonomy of interpretability

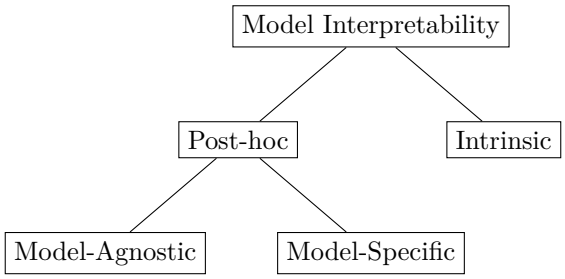


Figure 2.6: Taxonomy of the model interpretability.

There are two major types of models: *white-box models* and *black-box models*. Interpretability of the first type is defined as the *Intrinsic* [9]. This type of interpretability covers all models which have an interpretable internal structure. As an example, the structure of a decision tree is considered interpretable, as well as the internal structure of a shallow neural network. This does not apply to deep neural networks where *post-hoc* interpretability is used. The post-hoc interpretability means that we are trying to explain a prediction of the model without explaining the exact internal mechanism of that model. Because of the complexity of CNNs, post-hoc interpretability is the only way to interpret this kind of model.

Remark. This is not the only taxonomy of interpretability. The structure of the interpretability can be defined in many ways (either by the purpose, by the method, or by the application).

Model-Agnostic and Model-Specific

As shown in the taxonomy (see Fig. 2.6), post-hoc interpretability is divided into *model-agnostic* and *model-specific* [2]. The model-agnostic methods are the methods that can be applied to any black-box model without the concern about the internal structure of the model. These methods are usually less precise but because they explain the models' behavior only based on the input and the output. On the other hand, model-specific methods are associated with a particular type of model. The "type" is loosely defined and can refer to the whole domain like CNNs or the specific architecture of CNN. In this thesis, both types of post-hoc interpretability are used.

2.3.2 Right to explanation

The "right to explanation" is a term used by the European Parliament and Council in the General Data Protection Regulation (GDPR) [1]. This term is often mentioned in regard to XAI methods and requires a data controller to explain how the mechanism reached a decision. This part of the GDPR was created with an intent to prevent using the systems what decision cannot be interpreted by the human (like deep neural networks). The goal is to avoid discrimination and ethical/financial biases in such systems. As an example, we can use the automated credit scoring system. Systems like that are used in the mortgage loan process. It is legally forbidden to discriminate against a person base on a list of characteristics, but that discrimination can be hidden inside a black-box system (even without the knowledge of the creator of that system). If the application is refused by the bank, the applicant can require an explanation of why it was refused. This might help to potentially improve the score before the next application.

2.4 Attribution Methods

Attribution methods are one of the types of post-hoc methods (see section 2.3.1). Attribution methods, as the name says, attribute the input features to a given prediction. It can be defined as:

Definition 2.4 (Attribution method) *Given an input vector $x \in \mathbb{R}^n$ where n represents the number of dimensions, class C and the model $F : \mathbb{R}^n \to \mathbb{R}^C$. The attribution method is defined as $A(F, x, C) : \mathbb{R}^n \to \mathbb{R}^n$. Provided explanation corresponds to the "importance" of an element from the input vector for a given class C and model F.*

This definition can be rewritten to fit the input of the usual Convolutional Neural Network with the $m \times n$ input matrix:

Definition 2.5 (Attribution method - CNN) *Given an input matrix $x \in \mathbb{R}^{m \times n}$ where $m \times n$ represents the dimensions of the input, class C and the model $F : \mathbb{R}^{m \times n} \to \mathbb{R}^C$. The attribution method is defined as $A(F, x, C) : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$. Provided explanation corresponds to the "importance" of an element from the input matrix for a given class C and model F.*

To better visualize the attribution, we can look at Figure 2.7. For predicting a class of *ibizan_hound*, each pixel of the input image was given a value that defined its attribution to the prediction. Pixels (features) with higher attribution could be considered "more important" in predicting that class. We can see that the pixels with the highest attribution values are the pixels at the edges of the dog's head and ears.

As in the case of interpretability versus explainability (section 2.3), the is a lack of agreement on vocabulary. Attribution methods are often known as *saliency methods*, *feature relevance*, *feature importance*, *heatmaps*, *neuron activations*, and *saliency masks*.

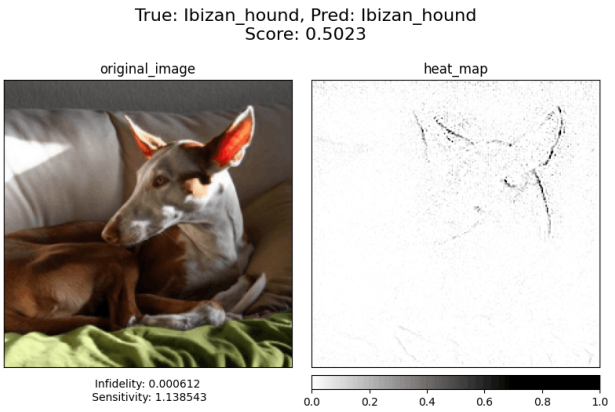


Figure 2.7: Visualization of the attribution by the Guided GradCAM generated for the class *ibizan_hound*. Image source: *Stanford Dogs* [31]

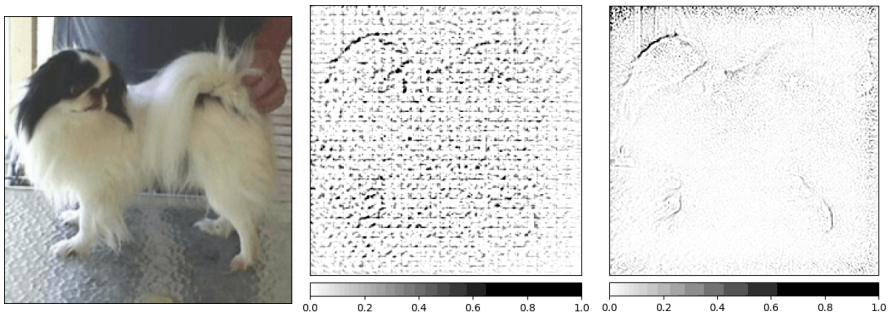
[1] Reg (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Dir 95/46/EC (General Data Protection Regulation) 2016.

2.5 Quantitative and Qualitative research

The type of research can be divided into qualitative or quantitative [11]. The qualitative type of research is often related to the observation and processing of non-numerical data. The contrast to qualitative research is called quantitative research, and it relies on the numerical analysis of collected data. Interpretability of machine learning models is a human-centric field of study. That is why most of the XAI method researchers base their solutions on qualitative measures rather than quantitative ones. This thesis argues that the current qualitative approach can prone to errors.

2.5.1 Qualitative measures

In qualitative research, the data is collected through observation, pools, or interviews. This kind of data is very subjective and related to the person providing the data. An example of such a subjective measure is shown in Figure 2.8. If presented with two attributions for the same input image, the decision of which one is better may change with the person. This is even more likely when two attributions have similar "quality".



(a) Input data (b) Attribution 1 (c) Attribution 2

Figure 2.8: Comparison of attributions from two different methods for the same input data. Both attributions are generated for the same model with different XAI methods. Image source: *Stanford Dogs* [31]

The results of the qualitative measures cannot be compared with each other without the specific context and prior knowledge about the data producers. Even with that knowledge, comparing the results might be difficult. The same issue applies to reproducibility, were to repeat the measurement, we have to be sure that all the participants are going to give the same answer. Another problem with qualitative measures is their scalability. Because most of the methods rely on human input, to measure the same method again or to compare it with another method, we have to repeat the work twice.

2.5.2 Quantitative measures

The data in the quantitative research is stored in a numeric form. This numeric form has to be comparable with other data measured in the same way. Each measurement is repeatable, and when done once, it can be reused all over again. This type of research has a huge advantage over qualitative research because of its scalability. The problem with qualitative measurement is that we have to define the measure to return meaningful results. Defining a measure of human visual perception is a difficult task, and the measure itself should be objective, which is even harder when trying to measure complex things like attributions. Having such a measure allows us to easily compare and reproduce experiments.

2.6 Related work

Ghorbani et al. [20] discuss attacks using adversarial perturbations to produce different interpretations of the same image. Their paper shows that we can generate such adversarial perturbations that produce perceptively indistinguishable inputs with the same prediction, but the attribution differs by a considerable margin from the original attribution. This thesis also focuses on the effect of input perturbation, but instead of searching for adversarial perturbation, it applies augmentations that aim to simulate real-world image processing that happens in modern cameras and check how they influence attribution.

Another approach to the influence of perturbation on interpretability was taken by Kindermans et al. [32]. They have studied the effect of applying noise to the input image and what effect it has on the explanation provided by the methods available at that time.

One of the most influential papers on the subject was done by Adebayo et al. [3] in 2018. Their work examined the influence of noise on the gradient values on different layers in the CNN. This was done by checking the correlation between the value of the gradient before and after applying the noise. The check was performed using multiple similarity metrics. Their work proved that gradients could be influenced by simple noise.

Hooker et al. [26] were one of the first who tried to measure the XAI methods quantitatively. The exact framework used is described in section 4.3. The study assumed that if after removing pixels attributed as important to the prediction and retraining the model, the model still performs well, then the attribution is wrong. This approach might be logical, but it is extremely computationally expensive and therefore useless in real-world scenarios when there are limited resources.

3 Attribution Methods

3.1 Deconvolution

The idea of Deconvolution [84] comes from the work of Zeiler et al. [85] about *Deconvolutional Networks (deconvnets*. Deconvnets are designed to work similar to convolutional networks but reverse (reversing pooling component, reversing filter component etc.), and they can be trained using an unsupervised approach. In a deconvolutional approach to explaining the model, we are not training a deconvnet but rather probe our CNN with it.

To reconstruct the activation on a specific layer, we are attaching *deconv layers* to corresponding *CNN layers* (see Fig. 3.1). Then an image is passed through the CNN, and the network computes the output. To examine a reconstruction for a given class c, we have to set all activations except the one responsible for predicting class c to zero. Then we can propagate through deconvnet layers and pass all the feature maps as inputs to corresponding layers.

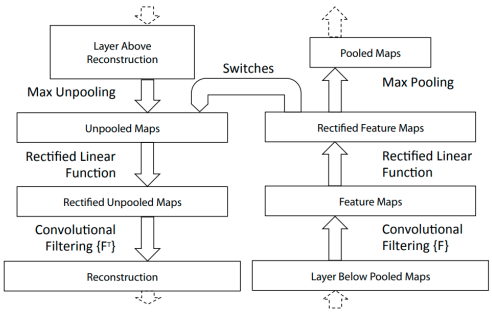


Figure 3.1: A deconvnet layer (left) attached to a CNN layer (right), source [84]

To calculate the reconstruction, deconvnet layer has to be able to reverse operations performed by the CNN layers. Authors designed specific components to compote the reverse operations done by CNN layers:

Filtering in the original CNN computes *feature maps* using learned filters. Reversing that operation requires the use of a transposed version of the same filters. Those transposed filters are then applied to the *Rectified Unpooled Maps*.

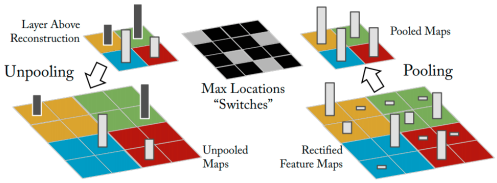


Figure 3.2: The unpooling layer, source [84]

Rectification uses the same *ReLU* non-linearity [24] to compute *Rectified Unpooled Maps* as it is used in CNN. It is simply just rectifying the values and propagate only non-negative ones to the *filtering* layer.

Unpooling corresponds to the *Pooling Layer* of CNN (see Fig. 3.2). The original max-pooling operation is non-invertible, but this approach uses additional variables called *switch variables*, which are responsible for remembering the locations of the maxima for each pooling region. The unpooling layer uses these variables to make a reconstruction into the same locations as when the pooling was calculated.

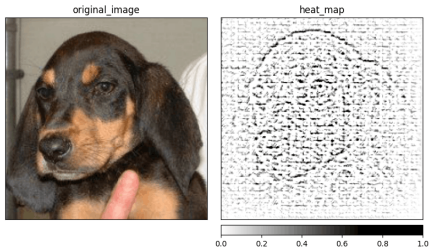


Figure 3.3: Visualization of the saliency map generated by deconvolution for the class *black-and-tan-coonhound*. Image source: *Stanford Dogs* [31]

Propagation through the whole deconvnet gives us a representation of the features from the first layer of the original CNN (the last deconvnet layer corresponds to the first CNN layer). This approach causes the saliency map to feature some biases from the first convolutional layer and the representation looks like a localized edge detector (see Fig. 3.3). It usually works better when there is a clear distinction in the feature importance rather than similar values for the whole image.

3.2 Saliency

Saliency [61] is one of the first attribution methods designed to visualize the input attribution of the Convolutional Network. Because the word *saliency* is often related to the whole approach to display input attribution called *Saliency Map*, this method is also known as *Vanilla Gradient*.

The idea of the Saliency method starts from the class visualization by finding L_2-regularized image I that maximizes score S_c for a given class c. It can be written formally as:

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2 \tag{3.1}$$

Where λ is a regularisation parameter. To find the value of I, we can use the back-propagation method. Unlike in the standard learning process, we are going to back-propagate with respect to the input image, not the first convolution layer. This optimization allows us to produce images that visualize a particular class in our model (see Fig. 3.4).

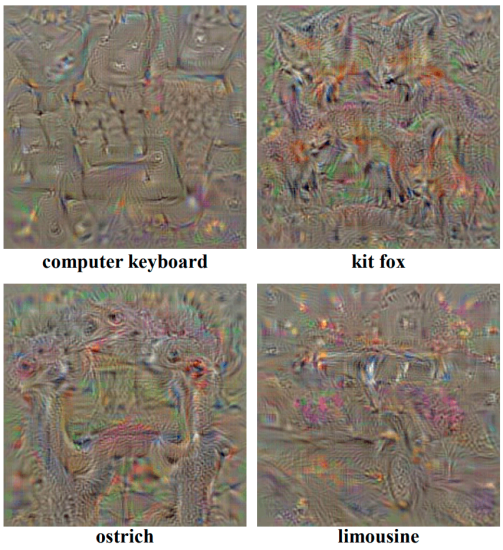


Figure 3.4: The class model visualizations for several classes, source [61]

This idea can be extrapolated, and with minor modifications, we should be able to query for spatial support of class c in a given image I_0. To do this, we have to rank pixels of I_0 in relation to their importance in predicting score $S_c(I_0)$. Authors assume that we can approximate $S_c(I)$ with a linear function in the neighborhood of I_0 with:

$$S_c(I) \approx w^\intercal I + b \tag{3.2}$$

where w is a:

$$w = \left.\frac{\partial S_c}{\partial I}\right|_{I_0} \tag{3.3}$$

For a pair of input image $I_0 \in \mathbb{R}^{m \times n}$ and the class c, we are able to compute saliency map $A \in \mathbb{R}^{m \times n}$ (where m and n are the height and width of the input in pixels). All we have to do is to compute derivative w (3.3) and rearrange elements in the returned vector.

This method uses different approaches base on the number of channels in the input image I_0. For grey-scale pixels (one color channel), we can rearrange the pixels to match the shape of the image. If the number of channels is greater than one, we are going to use the maximum value from each set of values related to the specified pixel.

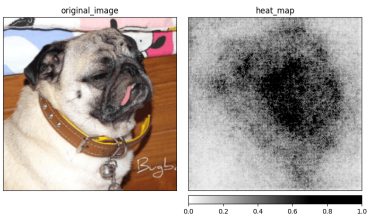


Figure 3.5: Visualization of the saliency map by the Saliency generated for the class *pug*. Image source: *Stanford Dogs* [31]

$$A_{i,j} = \max_{ch} |w_{h(i,j,ch)}| \tag{3.4}$$

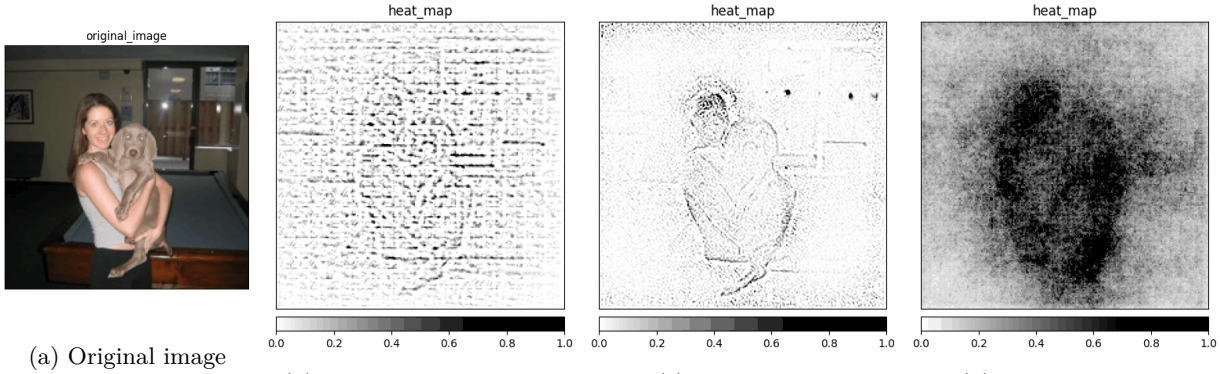
where ch is a color channel of the pixel (i, j) and $h(i, j, ch)$ is an index of the w corresponding to the same pixel (i, j). With the obtained map, we can visualize pixel importance for the input image I_0 as shown in the Figure 3.5.

The original *Saliency* method produces a lot of additional noise but still gives us an idea of which part of the input image is relevant when predicting a specific class. This often causes a problem when the object on the image has a lot of details and the model is using most of them to make a prediction.

3.3 Guided Backpropagation

Guided Backpropagation (GBP) [64] is an approach designed by Springenberg et al., relying on the ideas of *Deconvolution* (Section 3.1) and *Saliency* (Section 3.2). Authors argue that the approach taken by Simonyan et al. [61] has an issue with the flow of negative gradients, which decreases the accuracy of the higher layers we are trying to visualize. Their idea is to combine two approaches and add a "guide" to the *Saliency* with the help of deconvolution.

To achieve that, we have to focus on the *ReLU* activation function in the CNN. When computing values at the *Rectification* component of the *deconvnet*, we are masking all non-positive values with the *ReLU*. In that layer, the computed values are calculated only base on the top signal (reconstruction from the upper layer), and the input is ignored. On the other hand, in the *Saliency* method, we are focusing on the gradient values computed base on the input image. If we take deconvnet masking of the *Rectification* layer and apply it on the gradient values of the *Saliency* method, we could remove noise caused by the negative gradient values. This noise removal is the reason why the method has the prefix "guided". Deconvolution guides backpropagation values of the *Saliency* method to produce sharper images (Fig. 3.6c).



(a) Original image (b) Deconvolution results (c) GBP results (d) Saliency results

Figure 3.6: Visualisation of saliency maps produced by *Saliency* (3.6d), *Deconvolution* (3.6), and *GBP* (3.6c) of the same input image 3.6a for a class *weimaraner*. All the maps are generated using the same model (ResNet18). Image source: *Stanford Dogs* [31]

As we can see in Figure 3.6, the use of the "guide" significantly reduces the amount of noise generated by the *Saliency* method (Fig. 3.6d). The idea of GBP is often misunderstood and interpreted as "applying deconvolution results on the saliency results". This is not true because *ReLU* masking extracted from the deconvnet is applied on every level and therefore affecting the gradient values all the way down to the input of the CNN, not only at the first level of the CNN.

3.4 Integrated Gradients

Integrated Gradients (IG) [67] is a method proposed by Sundararajan et al. that bases on two axioms: *Sensitivity* and *Implementation Invariance*. Authors argue that those two axioms should be satisfied by all attribution methods. The definition of those two axioms is as follows:

Definition 3.1 (Axiom: Sensitivity) *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. If the function implemented by the deep network does not depend (mathematically) on some variable, then the attribution to that variable is always zero.*

Definition 3.2 (Axiom: Implementation Invariance) *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.*

The sensitivity axiom introduces the *baseline* which is an important part of the IG method. Baseline is defined as an *absence of feature* in an input. This definition is confusing, especially when dealing with complex models, but the baseline could be interpreted as *"input from the input space that produces a neutral prediction"*. A baseline can be treated as an input to produce a counterfactual explanation by checking how the model behaves when moving from baseline to the original image.

Remark. The authors give the example of the baseline for an object recognition network, which is a black image. I personally think that a black image doesn't represent an "absence of feature", because this absence should be defined based on the manifold that represents data. This means that a black image could work as an absence of a feature in one network but may not work for a network trained on a different dataset, allowing a network to use black pixels in prediction.

Authors argue that gradient-based methods are violating Sensitivity (Def. 3.1). As an example, we are presented with the case of simple function, $f(x) = 1 - \text{ReLU}(1 - x)$ (see Fig. 3.7) and the baseline being $x = 0$. When trying to generate attribution for $x = 2$, the functions' output changes from 0 to 1 but after $x = 1$, it becomes flat and causes the gradient to equal zero. Obviously, x attributes to the result, but because the function is flat at the input we are testing results in invalid attribution and breaks the Sensitivity. Sundararajan et al. think that breaking Sensitivity causes gradients to focus on irrelevant features.

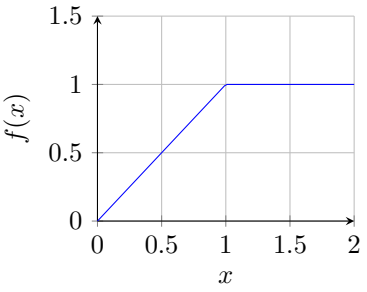


Figure 3.7: Values of $f(x) = 1 - \text{ReLU}(1 - x)$ where $x \in < 0, 2 >$

In IG definition we have a function F representing our model, input $x \in \mathbb{R}^n$ (x is in $\mathbb{R}^n$ because this is a general definition of IG and not CNN specific), and the baseline $x' \in \mathbb{R}^n$. We assume straightline path between x and x' and compute gradients long that path. The integrated gradient along i^th dimension is defined as:

$$IntegratedGrads_i(x) ::= (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \tag{3.5}$$

The original definition of Integrated Gradients is incalculable (because of integral). Therefore, the implementation of the method uses approximated value by replacing the integral with summation:

$$IntegratedGrads_i^{approx}(x) ::= (x_i - x'_i) \times \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m} \times (x - x'))}{\partial x_i} \times \frac{1}{m} \tag{3.6}$$

In the approximated calculation (Eq. 3.6), m defines a number of interpolation steps. As an example, we can visualize the interpolations with m equals five (see Fig. 3.8). In practice, the number of interpolation steps is usually between 20 and 300, but the mode value is equal to 50. The results of applying IG can be seen in Figure 3.9.

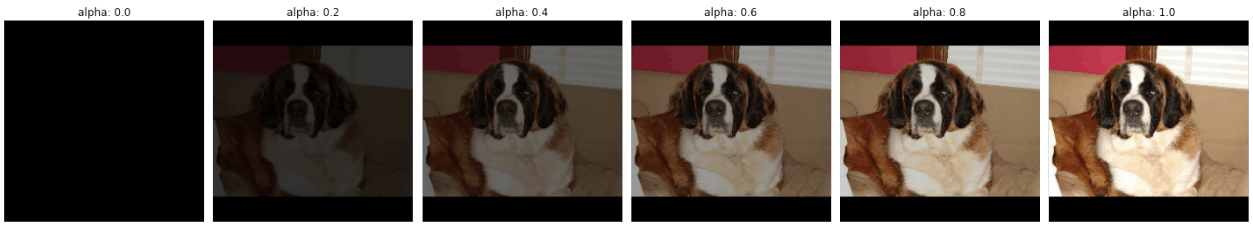


Figure 3.8: Five-step interpolation between the baseline x' and the input image x. The first image on the left (alpha:0.0) is not a part of the interpolation process. Image source: *Stanford Dogs* [31]

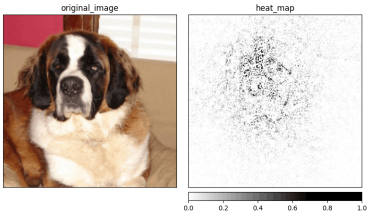
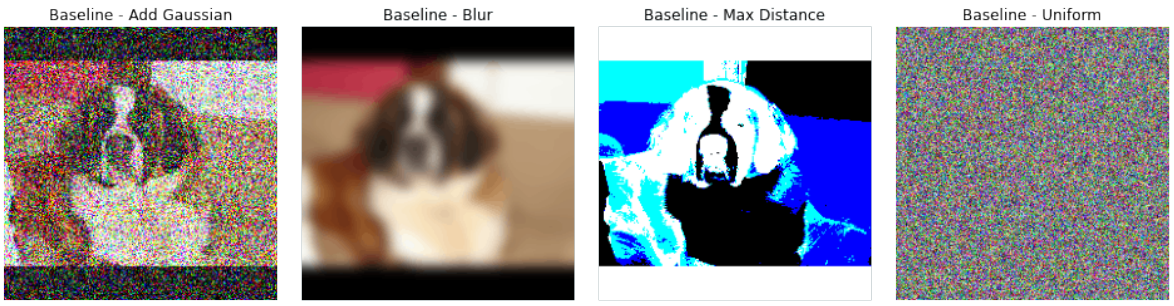


Figure 3.9: Visualization of the saliency map by the IG generated for the class *saint_bernard*. The result is averaged over 50 interpolation steps. Image source: *Stanford Dogs* [31]

In recent years there was discussion about replacing constant color baseline with an alternative. One of the first propositions was to add Gaussian noise to the original image (see Fig. 3.10a. *Gaussian baseline* was introduced by Smilkov et al. [63] and used a Gaussian distribution centered on the current image with a variance σ. This variance is the only parameter when tuning the method. Another baseline is called *Blur baseline* and uses a multi-dimensional gaussian filter (see Fig. 3.10b). The idea presented by Fong and Vedaldi [19] blurred version of the image is a domain-specific way to represent missing information and therefore be a valid baseline according to the original definition. Inspired by the work of Fong and Vedaldi, Sturmfels et al. [66] introduced another version of the baseline, which is based on the original image. This baseline is called the *Maximum Distance baseline* and creates a baseline by constructing an image with the largest value of the $L1$ distance from the original image. The result of that can be seen in Figure 3.10c. The problem with the maximum distance is that it doesn't represent the "absence of feature". It contains the information about the original image, just in a different form. In the same work, Sturmfels et al. created another baseline called *Uniform baseline*. This time, the baseline doesn't require an input image and uses only uniform distribution to generate a baseline (see Fig. 3.10d). The problem with selecting a baseline is not solved, and for any further experiments, the "black image" baseline is going to be used.



(a) Gaussian Baseline (b) Blur Baseline (c) Max Distance Baseline (d) Uniform Baseline

Figure 3.10: Alternative baselines for IG. Gaussian baseline is using $\sigma = 0.5$ to generate noise. Blur baseline is using $\sigma = 5$ in a gaussian filter. All the values are clipped at $< 0, 1 >$ to be within the range of the scaled colors. Image source: *Stanford Dogs* [31]

3.5 Guided GradCAM

Guided GradCAM is a method created by Selvaraju et al. [59] which combines both GBP and GradCAM (also created by the same authors). To explain the idea of Guided GradCAM, we have to split it into separate components.

Class Activation Mapping (CAM) [86] is an approach to localize regions responsible for a class prediction. This idea replaces the fully connected layers on the CNN with more convolutional layers and global average pooling (GAP) [40].

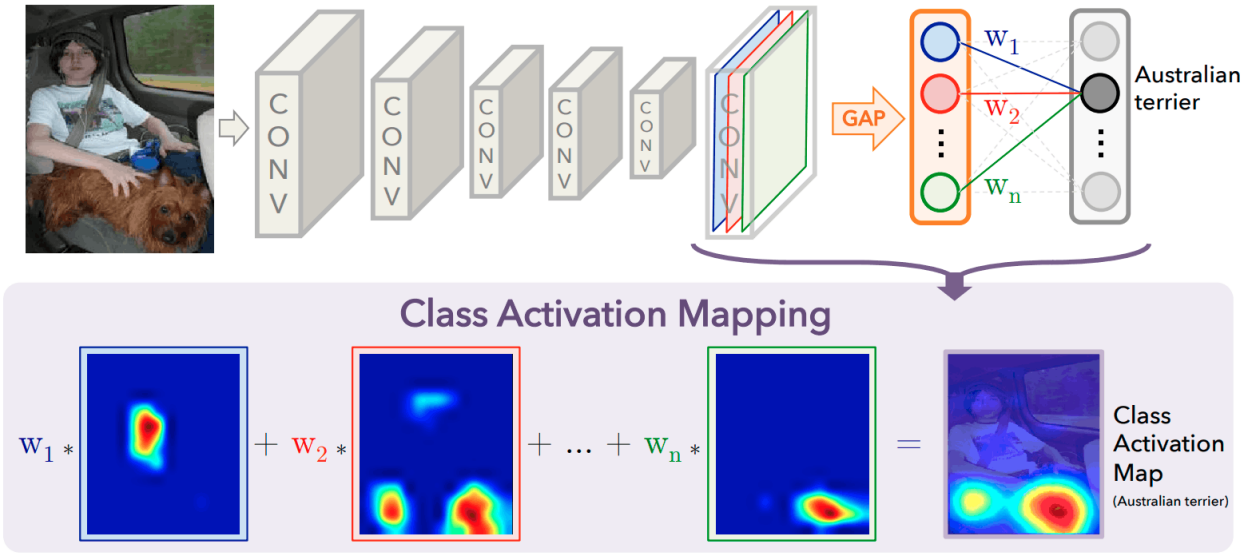


Figure 3.11: The modified architecture of the CNN that produces CAMs. Source: *Learning Deep Features for Discriminative Localization* [40]

As shown in Figure 3.11, just before the Softmax layer, the GAP operation is performed. This computes the spatial average of the feature map from the last CNN layer for every unit. The weighted sum of these spatial averages is calculated to produce the final output, and the Softmax layer gives us a normalized output used for classification. Class activation maps are computed similarly to the output, but the process is reversed. As an input, we have a class c, and the activation map for that class can be computed as:

$$M_c(x, y) = \sum_{k=1}^{n} w_k^c f_k(x, y) \tag{3.7}$$

Where w_k^c is a weight corresponding to the unit k and class c, and $f_k(x, y)$ is an activation of the last convolutional layer at the location (x, y) for unit k (visualized in Fig. 3.11, the bottom part of the image).

Selvaraju et al. argue that the CAM approach has a major drawback which is a requirement to modify the network and replace all fully-connected layers with convolutional layers. This makes CAM only applicable to certain tasks and reduces the performance of the network itself. They introduce the method that fixes all mentioned issues and call it **GradCAM**. This method is implementation agnostic (within the CNNs) and can be applied without any modifications to the network. As the name says, GradCAM uses a gradient to generate CAMs. Unlike in the original CAM paper, GradCAM allows selecting the convolutional layer we use as a feature map against which we compute the gradient. The gradient is computed for a given score y^c (c indicated a class) before the softmax layer. Then we use a feature map from the selected layer and calculate

weight a_k^c for every neuron (eq. 3.8) (similar to calculating weight for every unit in CAM).

$$a_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{3.8}$$

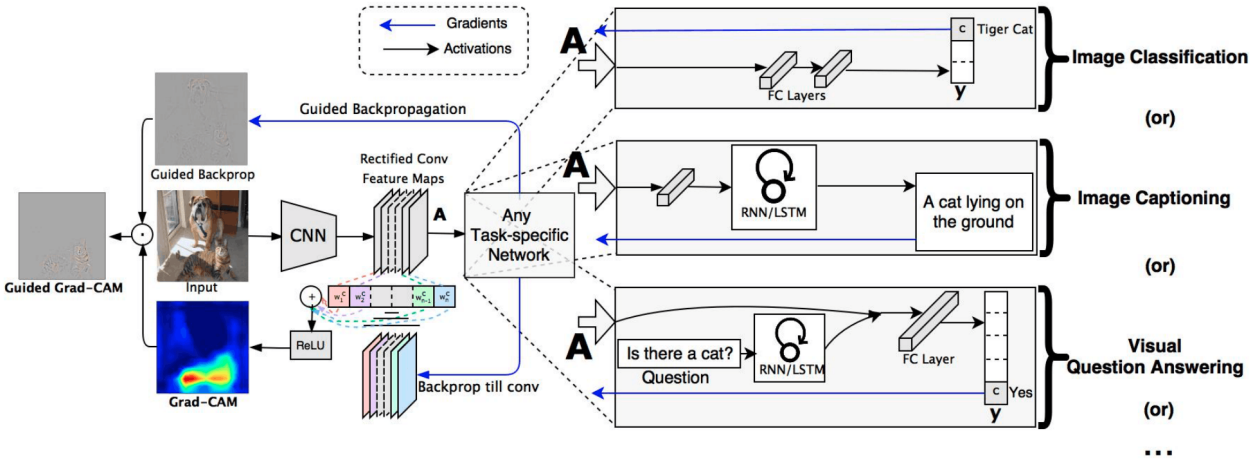


Figure 3.12: Guided GradCAM computation process. Source: *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization* [59]

$\frac{1}{Z}\sum_i\sum_j$ defines a global average pooling over the width (i) and height (j) and $\frac{\partial y^c}{\partial A_{ij}^k}$ is the gradient of score with respect to selected layers' feature map. With all a_k^c calculated, we can perform a linear combination of feature maps A^k and the neuron importance weights a_k^c (eq. 3.9).

$$M_{\text{GradCAM}}^c = \text{ReLU}\left(\sum_k a_k^c A^k\right) \tag{3.9}$$

The ReLU is used there to cutoff any non-positive values. The intuition behind using ReLU is that negative values more likely belong to other classes, which are also present in the image. The size of the map M_{GradCAM}^c is the same as the size of the convolutional feature map used to compute the gradient. Values from the map then have to be mapped on the original image to visualize which region was relevant in predicting class c.

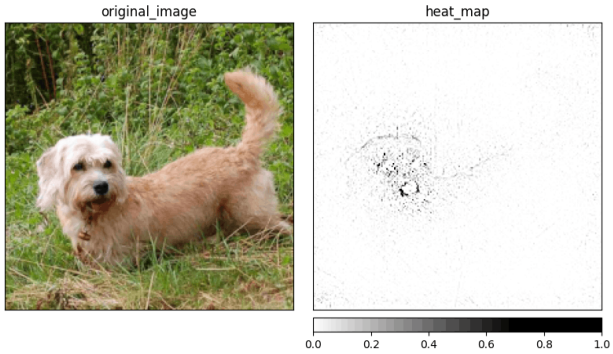


Figure 3.13: Visualization of the saliency map by the Guided GradCAM generated for the class *dandie_dinmont*. Image source: *Stanford Dogs* [31]

Guided GradCAM is a combination of GradCAM's map and GBP attribution. To compute the Guided GradCAM value, we are performing Hadamard product (also known as element-wise multiplication) of the attribution from GBP with a map from GradCAM (see Fig. 3.12). Combining GBP and GradCAM allow us to generate sharp attributions, as presented in Figure 3.13.

3.6 Gradient SHAP

Gradient SHAP (GradSHAP) was introduced by Erion et al. [15][1] and combined ideas from Integrated Gradients [67], SHAP [44], and SmoothGrad [63]. As mentioned in chapter 3.4, IG has a problem with selecting the right baseline for specific a model. There were attempts to fix a problem with the baseline [63][19][66] but selecting the right baseline with the "absence of feature" depends highly on the model architecture and datasets used in the training process.

$$ExpectedGradients_i(x) ::= \int_{x'} \left((x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \right) p_D(x') \partial x' \tag{3.10}$$

Erion et al. present an idea of *expected gradients*, which tries to avoid specifying single x' and use the data distribution instead p_D (eq. 3.10). Where x is an input vector, F is a model, α is an interpolation factor.

$$ExpectedGradient_i^{expected}(x) ::= \underset{x' \sim D, \alpha \sim U(0,1)}{E} \left[(x_i - x'_i) \times \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} \right] \tag{3.11}$$

As in the IG definition (eq. 3.5), calculating the integral is not possible, two integrals in the Expected Gradients have the approximate value solution (eq. 3.11). Where D is a our data and α comes from the uniform distribution $U(0, 1)$.

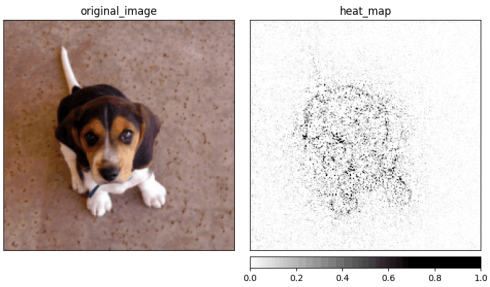


Figure 3.14: Visualization of the saliency map by the GradSHAP generated for the class *beagle*. The result is calculated with five random samples. Image source: *Stanford Dogs* [31]

In practice, this approach allows to approximate SHAP values. It computes expectations of gradients by sampling from the baselines and add noise to each input sample, n times. GradSHAP assumes that input features are independent of each other and the explanation is simply a composition of individual features' contribution.

In recent implementations [34], the method has multiple variants. One of the most important changes is to allow to replace $U(0, 1)$ with the Gaussian distribution $\mathcal{N}(0, \sigma^2)$. Using a Gaussian distribution allows modifying the standard deviation σ base on the original data. An example of the attribution generated by the GradSHAP method is shown in Figure 3.14.

3.7 Noise Tunnel

Noise Tunnel [34] is not an attribution method but a technique that improves the accuracy of attribution methods. It combines SmoothGrad [63], SmoothGrad-Square (unpublished version of the SmoothGrad), and VarGrad [3] and works with most of the attribution methods.

Noise Tunnel addresses a problem described in sections 3.3 and 3.4, where we discussed the problem with $ReLU$ activation function and gradients producing noisy, often irrelevant attributions. Because the partial derivative $\frac{\partial F_c}{\partial x_i}$ of the models' score F_c for a class c with respect to the value of the pixel x_i fluctuates, Smilkov et al. [63] thought that adding a Gaussian noise $\mathcal{N}(0, 0.01^2)$ and calculating an average of sampled attributions is going to solve the problem.

$$\hat{M}_c(x) = \frac{1}{n} \sum_{1}^{n} M_c \left(x + \mathcal{N}\left(0, \sigma^2\right) \right) \tag{3.12}$$

[1] This paper is a preprint, it is still under the review at ICLR conference

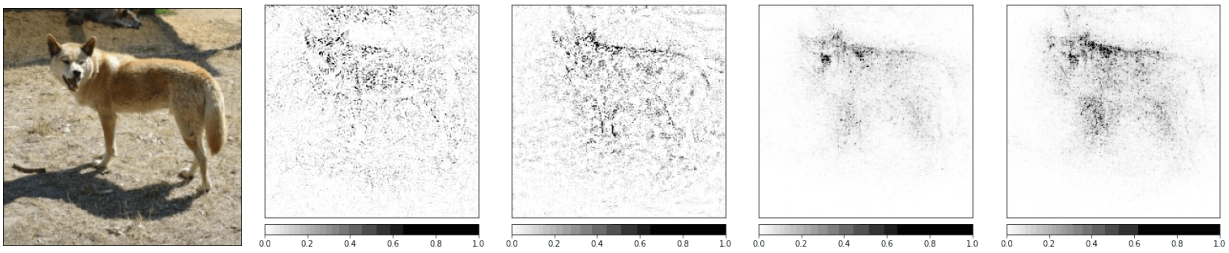
SmoothGrad (eq. 3.12) calculates the attribution (M_c) using any available method by providing that method an input with Gaussian noise. It then calculates a mean value from all the samples to reduce the importance of less frequent attributions. The idea is that when adding noise to the input image, important attributions are going to be visible most of the time, and noise might change between attributions.

Another version of the SmoothGrad Noise Tunnel is SmoothGrad-Square. It changes only the way that the mean value is calculated by using the mean of squared attributions instead of just attributions (eq. 3.13). This method usually provides less noisy results (compare Fig. 3.15c and Fig. 3.15d) but often removes less important features, which are still valid features.

$$\hat{M}_c(x) = \frac{1}{n}\sum_{1}^{n}\sqrt{M_c\left(x + \mathcal{N}\left(0, \sigma^2\right)\right)} \tag{3.13}$$

The third version of Noise Tunnel is a version using VarGrad (see Fig. 3.15e) which is a variance version of the SmoothGrad and can be defined as Eq. 3.14, where $\hat{M}_c$ is a value of SmoothGrad.

$$\tilde{M}_c(x) = \frac{1}{n}\sum_{k=1}^{n}\left\{M_c\left(x + \mathcal{N}\left(0, \sigma^2\right)\right)\right\}^2 - \left\{\hat{M}_c(x)\right\}^2 \tag{3.14}$$



(a) Original image (b) IG (c) SmoothGrad (d) SmoothGrad-Sq (e) VarGrad

Figure 3.15: Visualisation of saliency maps produced by *IG* (3.15b), *IG with SmoothGrad* (3.15c), *IG with SmoothGrad-Square* (3.15d), and *IG with VarGrad* (3.15e) of the same input image 3.15a for a class *dingo*. All the maps are generated using the same model (ResNet18). All noise tunnels are generating 10 random samples from $\mathcal{N}(0, 1)$ distribution. Image source: *Stanford Dogs* [31]

When comparing all the methods used in Noise Tunnel, we can see major differences in comparison with the original attribution (see Fig. 3.15). Using SmoothGrad (Fig. 3.15c) seams to detect more edges of the input image (in comparison with pure IG attribution in Fig. 3.15b), and that can be interpreted as detecting decision boundary. SmoothGrad-Square (Fig. 3.15d) and VarGrad (Fig. 3.15e) are removing a large amount of noise but usually also some of the important features visible on the attribution from SmoothGrad (look on the tail of the dingo). In my work, if referring to Noise Tunnel, the default option will be the SmoothGrad version (unless specified otherwise).

4 Measures

4.1 Sensitivity

Sensitivity [83] (*max-sensitivity* to be precise) is defined as a measure of change in the attribution with a small perturbation to the input.

Definition 4.1 (Sensitivity) *For a given model F, attribution method Φ, input x, insignificant perturbation δ, and input neighborhood radious r, max-sensitifity for a attribution is defined as:*

$$\mathrm{SENS}_{\mathrm{MAX}}(\Phi, \mathbf{F}, \mathbf{x}, r) := \max_{\|\delta\| \leqslant r} \|\Phi(\mathbf{F}, \mathbf{x} + \delta) - \Phi(\mathbf{F}, \mathbf{x})\| \tag{4.1}$$

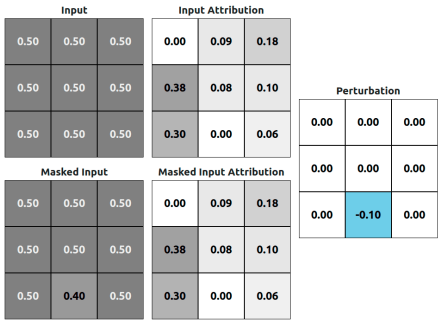


Figure 4.1: **Sensitivity = 0**. Perturbation of the input did not change the attribution between the input and masked input (one with added perturbation).

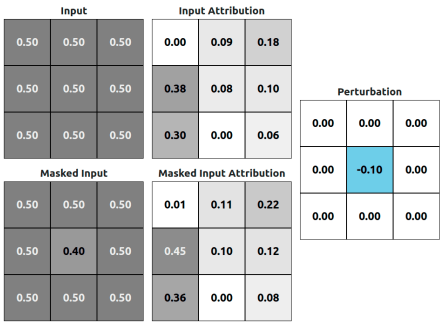


Figure 4.2: **Sensitivity = 0.11**. Perturbation of the feature with higher than zero initial attribution causes Sensitivity value to rise.

The radius r serves as a radius of a $l_p norm$ used to generate insignificant perturbation. It restricts the range of the perturbation and usually is set to around 0.02. Sensitivity returns a positive scalar which is the length of the sensitivity vector. This vector (as shown in eq. 4.1) is the difference between two attribution vectors, so the absolute length of this vector is proportional to the difference in every dimension of the attribution vectors.

To explain the intuition behind the sensitivity, we can use a toy example of the model that takes $3x3$ input and predicts the class $[0, 1]$. This model predicts class 0 with a 0.95 score when receiving the input filled with 0.50. Attribution for this prediction is shown in Figure 4.1 (Input Attribution). If a perturbation is added to the input, then new attribution is calculated for the same class (0) but for the *Masked Input*. If the perturbation doesn't affect the attribution (as in Fig. 4.1), then returned Sensitivity measure is equal to zero.

It is common for attribution to remain constant when perturbing features of the input with initial attribution close to zero. Attribution defines how much specific feature is involved in the model decision. Changing the value, which is irrelevant for the model, usually results in close to no change in the attribution.

An example that causes the SENS value to change can be seen in Figure 4.2. This time perturbation changes the feature with non-zero attribution of the original input (*Input Attribution*). That perturbation causes the new attribution (*Masked Input Attribution*) to differ slightly from the original one. The length of the vector (total difference between two attributions) is equal to 0.11 which is a square root of squared differences between corresponding attribution positions.

Remark. Perturbation shown in the example is created for the purpose of the example. In practice, values in the perturbation will be significantly smaller and will affect the whole input, not only one feature.

4.2 Infidelity

Infidelity [83] comes from the idea of fidelity, which could be defined as the amount of how many features with high attributions belong to a priori subset of features that are relevant to the prediction. This definition assumes that we have such a subset of relevant features, and that is not a valid assumption in most situations. In another paper [5], Ancona et al. are trying to come up with a quantitative measure for the XAI methods, which has no human bias. The idea is to relate the new measure to the correlation between the importance of features and the change in models' prediction. This can be achieved by perturbing the input and calculating the correlation between the perturbation of features with the predicted score.

Definition 4.2 (Infidelity) *For a given model F, attribution method Φ, input x, and significant perturbation around input I, infidelity for an attribution method is defined as:*

$$\text{INFD}(\Phi, \mathbf{F}, \mathbf{x}) = \mathbb{E}_{\mathbf{I} \sim \mu_{\text{I}}} \left[\left(\mathbf{I}^T \Phi(\mathbf{F}, \mathbf{x}) - (\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x} - \mathbf{I})) \right)^2 \right] \tag{4.2}$$

Infidelity measure (eq. 4.2) consists of two major part which expected difference is a measure of infidelity. The first part ($\mathbf{I}^T \Phi(\mathbf{F}, \mathbf{x})$) is a dot product between the perturbation and the attribution generated by the XAI method. Attribution used for the calculation is the attribution for the image without perturbation. This method does not rely on the attribution of the perturbed image anywhere. The second part $(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x} - \mathbf{I}))^2$ is a squared difference between the score of the original image and the perturbed image. The perturbation is usually drawn from the Gaussian distribution with the $\mu = 0$, and the σ is the parameter.

Figure 4.3: Small perturbation of the feature value with the high initial attribution results in a small infidelity measure.

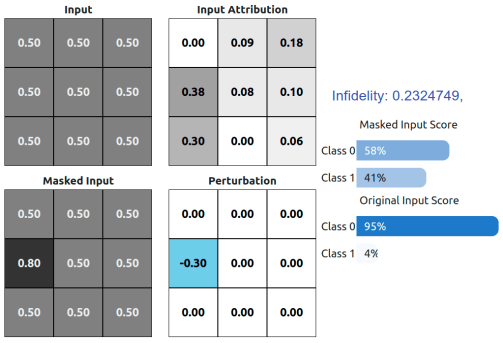


Figure 4.4: High perturbation of the feature value with the high initial attribution results in a large infidelity measure.

To explain the intuition behind the infidelity measure as mentioned in section 4.1. This time the perturbation is done on the feature with high initial attribution of 0.38. The first part of the equation is a dot product between transposed *Perturbation* matrix and the *Input Attribution* matrix. The second part is a difference between the *Original Input Score* of class 0 and the *Masked Input Score* of class 0.

As shown in Figure 4.3, changing the feature with high attribution by a small amount (-0.1) resulted in a relatively small infidelity score of 0.0091. This value is primarily due to a small change in the class prediction between input and the masked input. The next example (Fig. 4.4) shows a significantly larger value of infidelity. This time the change in score is greater in relation to change in the input features (score for class 0 dropped by 0.37 for 0.3 change in the value of the feature).

These two examples show how infidelity behaves and what it tries to measure. If the initial attribution of the feature is correct, then the infidelity values should be close to zero. If the correlation between attribution value and the change in models' prediction varies, then the infidelity values should be higher. By definition, the measure checks if the fidelity of the attributions relates to what the model predicts.

4.3 RemOve And Retrain

The ROAR (RemOve And Retrain) framework [26] is another approach to compare XAI methods. The assumption is that if the feature attribution is correct, then by removing those features, the classifies should degrade. This assumption is based on the work done by Samek et al.[58] and shows promising results. Hooker argues that the method violates the key assumption of machine learning, where the training data and the test data have to be from the same distribution. By removing attributed features, we are creating data samples that come from different distribution, and therefore loss in the classification performance might be due to the distribution shift. To solve that problem, the authors propose to retrain the model on the dataset with removed features and then compare the models' performance.

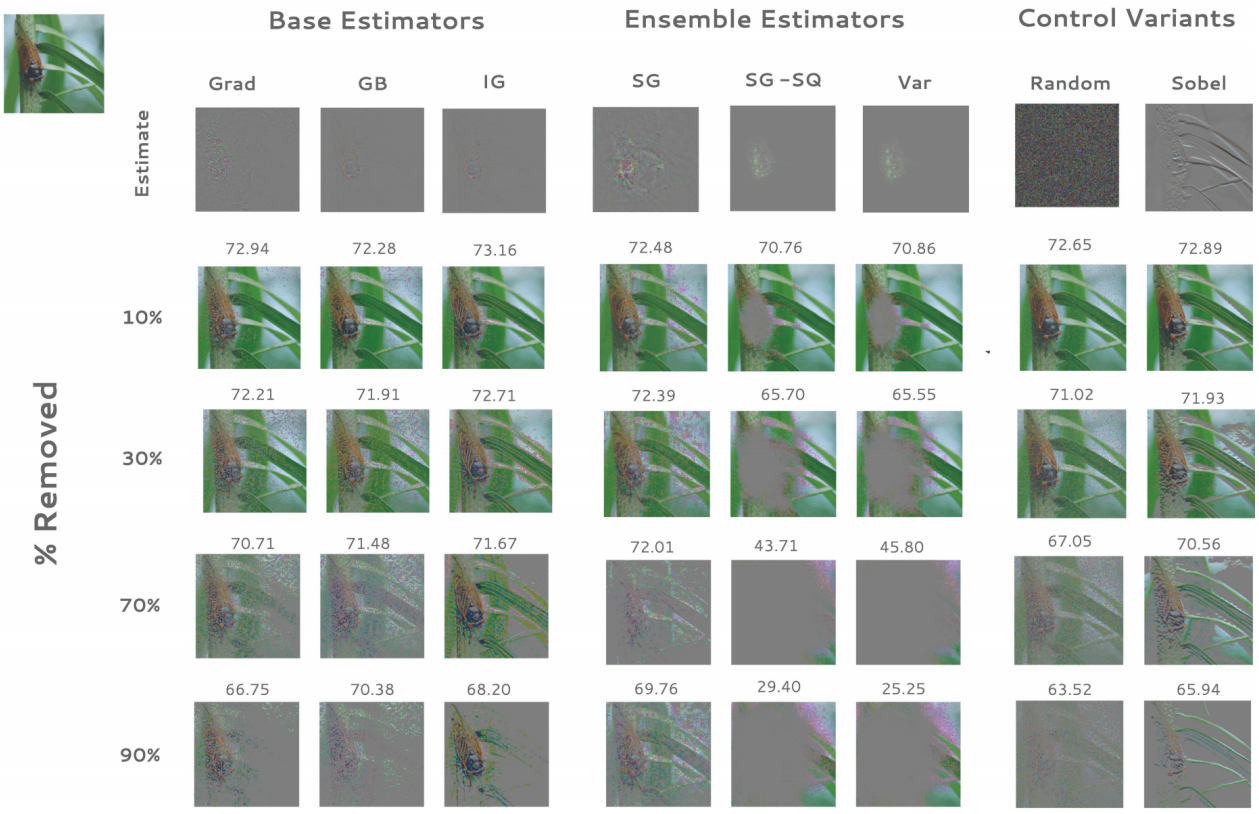


Figure 4.5: Modification of the image according to the ROAR framework. The percentage of the pixels with the highest attributions is replaced by the mean value (*% Removed*). Scores above each image refer to the average test accuracy of the models trained on modified dataset. Methods used in the experiments: Saliency (GRAD) [61], Guided Backpropagation (GPB) [64], Integrated Gradients (IG) [67], SmoothGrad (SG) and SmoothGrad-Square (SG-SQ) [61], VarGrad (Var) [59], Random Bernoulli, Sobel Edge Filter. Source: *A Benchmark for Interpretability Methods in Deep Neural Networks* [26]

For the framework to work, it has to generate input attribution for the image and order each feature attribution from the most important to the less important. With the ordered list, it can remove a given top t% of features and replace them with a mean value of all pixels (see Fig. 4.5). This process is repeated for $t = [0, 10, ..., 100]$ (t is a percentage of all features). After removing those features, a new model is trained on the modified dataset, and the degradation in accuracy is calculated.

As shown in Figure 4.6, methods using Noise Tunnel achieve significantly better results from their original versions. The ROAR framework with its approach to compare XAI methods is an interesting proposition but has one important drawback. Every time the method is tested, we have to retrain hundreds of models, and without a computation cluster, this is not possible in an acceptable time frame.

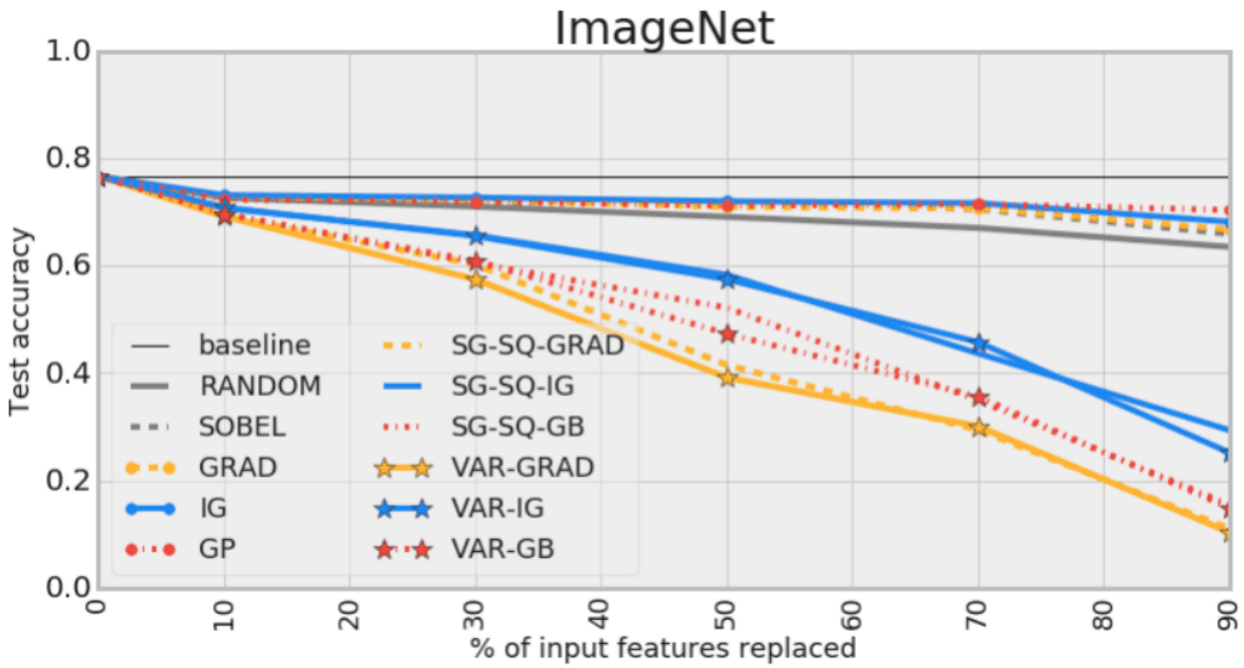


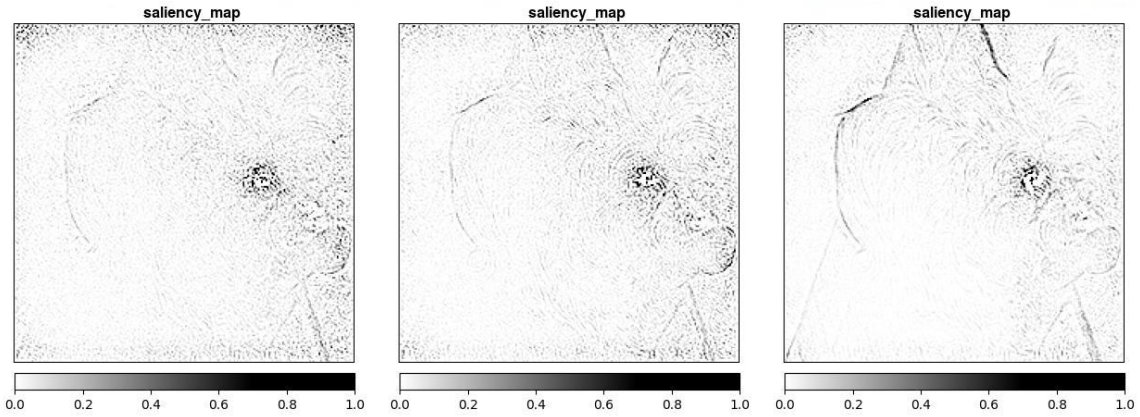
Figure 4.6: Results from the ROAR framework on the ImageNet dataset [57]. A method with the highest drop (Saliency-based Noise Tunnel with VarGrad) is considered to be the one with the most relevant attribution. The biggest drop in the accuracy, the better the XAI method is according to the assumptions. Source: *A Benchmark for Interpretability Methods in Deep Neural Networks* [26]

4.4 Structural Similarity Index Measure

Structural Similarity Index Measure (SSIM) [79] is a metric used to calculate the similarity between two images. It measures a perceived change in the structural information between images and is used for picture comparison in television. The index does not use the whole image but rather NxN windows to calculate similarity scores. The window size used in the original paper is $11x11$, and if not specified otherwise, this is a window used in my work. The SSIM is defined as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + (k_1L)^2)(2\sigma_{xy} + (k_2L)^2)}{(\mu_x^2 + \mu_y^2 + (k_1L)^2)(\sigma_x^2 + \sigma_y^2 + (k_2L)^2)} \tag{4.3}$$

Where x and y are the images to compare, mu_x and mu_y are the mean values of pixels from the window. σ_x and σ_y are the variance of those pixels from the window, and σ_{xy} is a covariance of x and y. There is also the L parameter, which is important when trying to compare SSIM values calculated for the images from different distributions. This parameter defined a dynamic range of values present in the image. In the case of comparing attributions, it usually is a maximum value of attribution from a given dataset (assuming the same attribution method and model). Values of k_1 and k_2 are constant and equal to 0.01 and 0.03 accordingly.



(a) Baseline (b) $SSIM = 0.955$ (c) $SSIM = 0.885$

Figure 4.7: SSIM scores for comparing the baseline attribution (Fig. 4.7a) with attributions of augmented images. Image source: *Stanford Dogs* [31]

The amount SSIM value change with the change in the perception of the image is presented in Figure 4.7. The higher score achieved when comparing image 4.7a with 4.7b can be related to the visual similarity between two images. We can clearly see that attribution from image 4.7c is sharper, and therefore less similar to the baseline.

Remark. Quantifying human visual perception is still an active field of research, and SSIM metric is not the perfect solution. There are other methods, but they all had some issues when compared with real humans. This metric is going to be used to qualitatively compare attributions because comparing thousands of images qualitatively is not possible in a reasonable amount of time.

5 Experiments

In this chapter, I will describe the experimental setup with all requires libraries. Section 5.1 includes the description, analysis, and examples from the datasets used in the experiments. It also provides an explanation and the assumption on how to simulate a drop in the model performance 5.1 used in training. The model training process is defined in section 5.2. Because the model performance and training are not a part of experiments, training results are also added to this section. Real-world augmentations used in this study are defined in section 5.3. These augmentations are applied to the images in the first part of the experiments (described in Section 5.4). Experiments are divided into two parts. The first part (defined in Section 5.4) is going to test the effect that augmentations have on attribution methods. The second part (defined in Section 5.5) is going to check if there is a reliable measurement that can be used to compare XAI methods.

Environment and framework

The training environment was designed around the PyTorch library [51][1]. Augmentations described in section 5.3 are performed with the help of the G'MIC library [73]. Most of the experiments use Captum [34] and scikit-learn [52] libraries to process, analyze and produce results of attribution methods. In the experiments, not all methods are used for all experiments. This is due to the limited amount of memory available on the graphic card used to perform analysis. Methods like Integrated Gradients [67] cannot be used to calculate the value of Sensitivity [83] because it requires storing over 500 interpolations to generate the result. These details will be explained in sections describing a particular experiment. All the experiments are run on the *GeForce GTX 1080 Ti* with 11GB of memory, and the total processing time of all experiments is around 508 hours (this assumes running a version with example generation).

5.1 Datasets

All the datasets used in this study are publicly available. The selection process was aimed at finding a variety of different domains and complexity. Five datasets were selected, and each dataset is split into a training dataset and test dataset. If not defined in the dataset description, the split was done using 80 : 20 proportions (*training : test*). Besides the original split, each training dataset was additionally split into five training datasets. The second split creates the following structure for every dataset:

Table 5.1: Default dataset split structure

	Training	Test
A	100% * 80%	20%
B	80% * 80%	20%
C	60% * 80%	20%
D	40% * 80%	20%
E	20% * 80%	20%

Table 5.1 can be read as *"split B of the dataset X uses 80% training dataset which consists of 80% of the entire dataset, and full test dataset which consists of 20% of the total dataset"*. If the class distribution is not constant, then splits of the train datasets are stratified.

[1] Detailed implementation and guide is available at https://github.com/burnpiro/xai-correlation, Wiki page with additional instructions and results is available at https://github.com/burnpiro/xai-correlation/wiki

Remark. This split assumes that using a fraction of the training dataset for training will cause models to perform worse when tested on the same test dataset. It should simulate real-world issues with model accuracy and allow to experiment on none perfect models. The drop in the accuracy is not going to be linear because of the different complexities of each data domain.

5.1.1 Stanford Dogs

The dataset created by Stanford University consists of dog pictures. It has 120 classes, and the train/test data split is defined by the authors. Class distribution is proportional between train and test splits. Dataset is slightly unbalanced, with the ratio between the least frequently occurring class and the most common class being 1.7 (detailed class distribution available in appendix A.1.1). The total number of images is 20580 (train: 16418, test: 4162). Examples from the datasets can be seen in Figure 5.1.

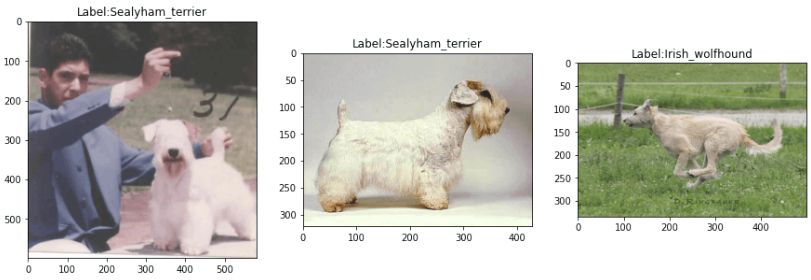


Figure 5.1: Example images from in *Stanford Dogs* [31]

5.1.2 Food 101

The dataset consists of food pictures. It has 101 classes, and the train/test data split is defined by the authors. Classes are distributed proportionally, and each class has 750 examples in the train dataset and 250 examples in the test dataset, with the total number of images at 101000 (train: 75750, test: 25250). Examples from the datasets can be seen in Figure 5.2.

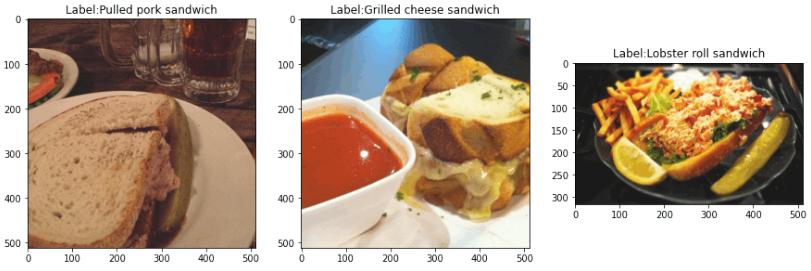


Figure 5.2: Example images from in *Food 101* [16]

5.1.3 Edible wild plants

The dataset consists of edible wild plants pictures. It has 62 classes, and the train/test data split is defined by the authors. Class distribution differs between train and test splits. Train dataset is highly unbalanced, with the ratio between the least frequently occurring class and the most common class being 21 (detailed class distribution available in appendix A.1.3). The unbalance is caused by 3 major classes. The rest of the dataset has the same ratio equal to 3. The test dataset is balanced with 5 examples for each class. The total number of images is 6868 (train: 6558, test: 310). Examples from the datasets can be seen in Figure 5.3.

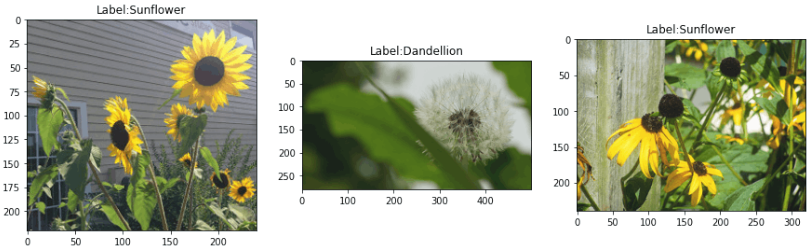


Figure 5.3: Example images from in *Edible wild plants* [78]

5.1.4 Marvel Heroes

The dataset consists of pictures from the Marvel Universe franchise. It has 8 classes, and the train/test data split is defined by the authors. Class distribution is proportional between train and test splits. Dataset is fairly balanced with ratio between the least frequently occurring class and the most common class being 1.12 (detailed class distribution available in appendix A.1.4). The total number of images is 3035 (train: 2584, test: 451). Examples from the datasets can be seen in Figure 5.4.

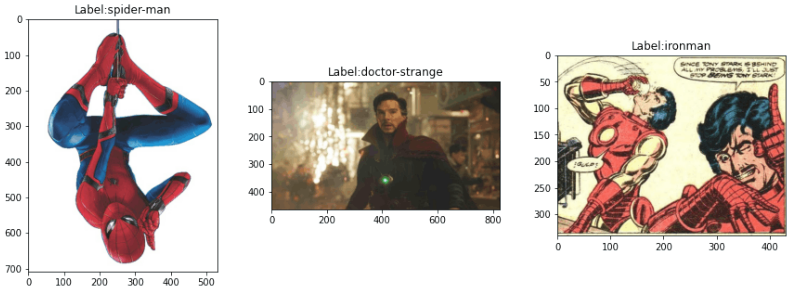


Figure 5.4: Example images from in *Marvel* [17]

5.1.5 Plants

The dataset consists of plant pictures. It has 99 classes, and the train/test data split is defined by the authors. Class distribution is proportional between train and test splits (with the test dataset being slightly more balanced). Dataset is balanced with ratio between the least frequently occurring class and the most common class, being 7 (detailed class distribution available in appendix A.1.5). The total number of images is 14670 (train: 13149, test: 1521). Examples from the datasets can be seen in Figure 5.5.

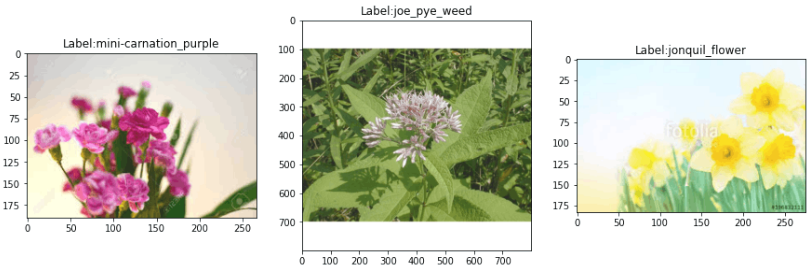
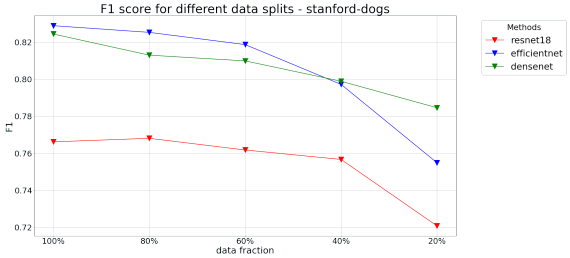


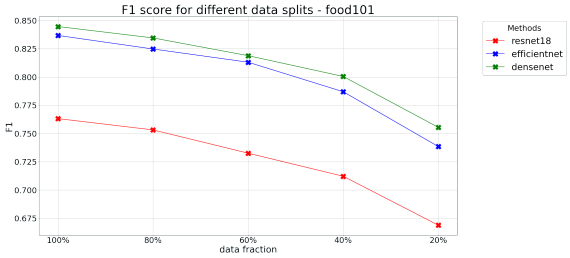
Figure 5.5: Example images from in *Plants* [28]

5.2 Models and training

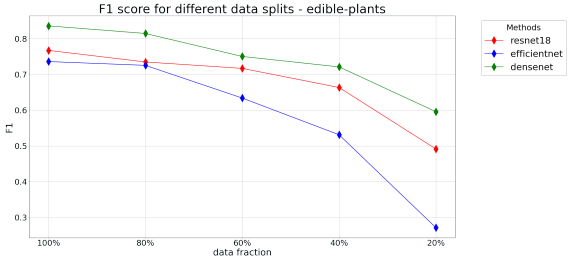
This study uses three different model architectures: ResNet 18 [25], DenseNet 121 [27] and EfficientNet B0 [68]. These architectures were selected base on popularity in the data science community and their availability in popular frameworks. Each model was first pretrained on the ImageNet dataset [12], and fully connected layers from the top of the architecture were removed. There were no further changes to the original structures. On top of the model, a single fully-connected layer was added with the output size matching the number of classes in the particular dataset used for fine-tuning. Fine-tuning is done using Stochastic Gradient Descent (SGD) with momentum [56] and Cross-Entropy loss function. Each model was trained for 15 epochs with a decrease of the learning rate (from *0.001* to *0.0001*) after the 7th epoch. Each architecture was trained on every available dataset and every training fraction split of that dataset (train dataset splitting described in section 5.1). In total, 75 different models were trained for the purpose of the experiments.



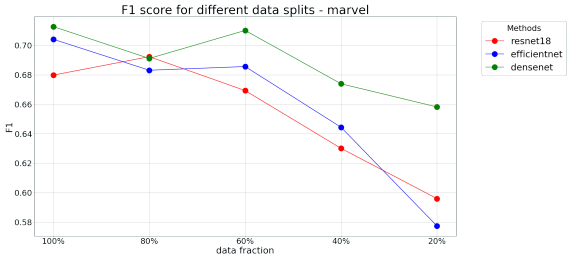
(a) Stanford Dogs dataset, F1 scores



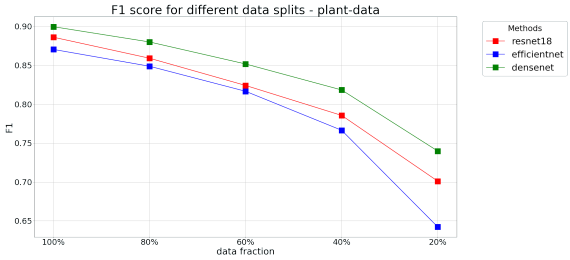
(b) Food 101 dataset, F1 scores



(c) Edible wild plants dataset, F1 scores



(d) Marvel dataset, F1 scores



(e) Plants dataset, F1 scores

Figure 5.6: F1 scores achieved by models. Each model was trained using a different fraction of the train dataset and then tested on the full test dataset. We can see a significant drop in the models' performance in most of the datasets. The only outliers are visible in models trained on the *Marvel Heroes* dataset (Fig. 5.6d), where scores achieved by the models trained on less data are better in two cases.

The results of the training process are aligned with the assumption made (remark 5.1). Models' performance drops with removing more and more data from the training dataset. There are three exceptions, and all of them are related to the Marvel dataset (see Fig. 5.6d). All the architectures observed a slight increase of models' score when were trained on less than 100% of the training dataset. That increase did not last when more data was removed, and the rest of the datasets (Fig. 5.6a, Fig. 5.6b, Fig. 5.6c, Fig. 5.6e) did not experience the same phenomenon. Detail results are available in appendix A.2.

5.3 Augmentations

To simulate real-world scenarios when evaluating attribution methods on datasets, this work uses the multiple augmentations technique. All described augmentations happen only when the calculation of the SSIM metric is done and never influences the results of the *Infidelity* and the *Sensitivity* measures.

Figure 5.7: Examples of real-world augmentations applied on the input image. Image source: *Stanford Dogs* [31]

Rotation

This type of augmentation is one of the simplest augmentations we can apply to any image. It uses the rotation matrix[2] to rotate the position of every pixel and fills the empty space with black pixels. This work is using four rotations $\{-30°, -15°, 15°, 30°\}$ degrees. Because rotation influences the position of the object on the image, when calculating the SSIM value between two attributions, the same rotation is applied to the attribution of the original image. The result of the applied rotation of 15° and −15° can be seen in Figure 5.7 (bottom row).

Freaky Details

Freaky details filter [72] is a method used to enhance details of the input image. It works by creating a copy of the image with inverted color and applying a Bilateral filter [70] to it. After the filter, two images are blended together with a *Vivid Light* mode which blends images base on how "light" is a given pixel. It lightens the pixels that are lighter than 50% gray color and darkens the pixels that are darker than 50% gray color. The result of the applied filter can be seen in Figure 5.7 in column 2, row 1. It can be applied iteratively, but for the purpose of experiments, it is only applied once.

Normalize Local

Local Normalization filter [74] is a filter that uses local mean and variance to correct local illumination or shading artifacts. For every pixel at the (x, y) position, we normalize its value:

$$j_{x,y} = \frac{i_{x,y} - \mu_{loc}(i_{x,y})}{\sigma_{loc}(i_{x,y})} \tag{5.1}$$

Where $i_{x,y}$ is a value of the pixel at (x, y) position, μ_{loc} is a mean value of pixels within a given radius, σ_{loc} is a variance of the pixels within a given radius. The radius is a parameter, and in the rest of the experiments, it is always 10 pixels.

Boost Chromaticity

Boost Chromaticity filter [71] is a filter responsible for changing the "colorfulness" of the pixels. It uses the chromaticity diagram to adjust the chromaticity of the pixels. It usually moves the value of the pixel

[2]Rotation matrix is defined as $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ for a two-dimensional vector, where θ is an angle we want to rotate

along with the selected color space to avoid spikes on the chromaticity histogram. In the experiments, the $YCbCr$ color space is used.

Mighty Details

Mighty details filter [75] is a filter similar to Freaky details. The main difference is that Mighty details apply smoothing (Gaussian blur) before blending two images.

Sharpen

Sharpening filter [76] is one of the basic filters available in most image processing software. It uses Gaussian blur to create a blurred version of the original image. A blurred image is then compared with the original image, and if the difference between them is greater than a specific threshold, images are subtracted.

5.4 Don't Augment Me

The first part of the experiments relies on the hypothesis that real-world augmentations, which do not cause any significant changes in the augmented image and also do not cause a model to change its prediction significantly, may cause an attribution method to produce significantly different attribution of the augmented image. To measure the difference in attributions quantitatively, I am going to use SSIM [79] metric. Because quantifying a human visual perception cannot be measured with any known metric and is an active field of research, in addition to the SSIM metric, some of the attributions are going to be presented for qualitative measurement.

Procedure

To be able to calculate the SSIM metric for the attributions, each dataset and method needs to have a maximum range calculated first. This requires an additional run through all images from the dataset through every model and used method to get a maximum value of attribution assigned by a method for a model. This value is used as a data range in SSIM for all attributions using the same method and model. Lists of calculated maximum attribution values are in appendix B.1.1. This range is set for a given experiment and not going to be explicitly mentioned in every one of them unless it is relevant to the change in the setup.

For an image from the test dataset, a set of augmentations will be applied. Augmentations are defined in Section 5.3, and all of them are used. The original image, as well as augmented images, are then forwarded through a selected model to get a prediction score. If the prediction score of the augmented image is within the $threshold = 0.05$ that of the original image (non-augmented), then the augmented image is accepted for an attribution comparison. To get the attribution, both the augmented image and the original image are forwarded through a selected attribution method, and resulted attributions are the inputs to the SSIM function. The whole process could be defined as:

$$SSIM_{x,x_i} = \begin{cases} SSIM(A(F,x), A(F,x_i)) & \text{if } |F(x) - F(x_i)| < 0.05 \\ \text{NaN} & \text{otherwise} \end{cases} \tag{5.2}$$

Where x is an original image, x_i is an augmented image, F is a model, A is an attribution method. There is one exception from this equation, and it occurs when the augmentation is a *rotation* (by any angle). The special case includes applying the same rotation to the attribution from the original image and can be summarized as:

$$SSIM(R(A(F,x),\theta), A(F,R(x,\theta)))) \tag{5.3}$$

The main condition ($|F(x) - F(x_i)| < 0.05$) is still in place, the x_i is replaced by the $R(x,\theta)$, which uses a rotation function R and the angle θ. The same rotation R is applied on the result from the attribution method $A(F,x)$. This fixes the issue of calculating the similarity of rotated attribution to the none rotated

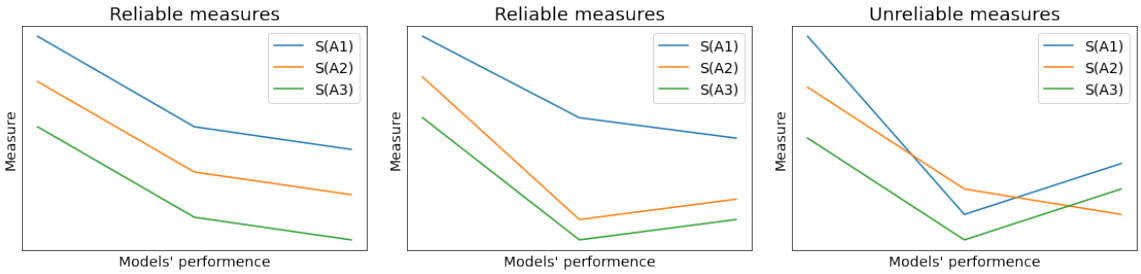
attribution. This special case is done under the assumption that rotation of the input image should cause the a similar rotation of the attribution for this image. Comparing attribution from the rotated image to the attribution from the original image would not make sense because the region which is the most important for a prediction is rotated by the θ angle.

With all SSIM values calculated, each method will have a mean and the standard deviation value calculated. Additionally, mean and standard deviation values will also be calculated per augmentation. The first calculation should allow comparing the methods, and the second is going to be a supplementary check if there are any exceptions for particular augmentation. The qualitative comparison of the results will include visual rendering of selected examples (original images) along with all augmentations applied to that example. Because the amount of augmentations is far greater than it can be displayed in a comprehensive way, augmentations are going to be split into *rotations* and *filters*. *Rotations* will include all available rotations 5.3 (4 in total), and *filters* will include everything except the *rotations* (5 in total). For an example to be considered a valid example, all augmented versions of that example have to achieve class score within the $threshold < 0.05$ from the a score of the original image (non-augmented). This is a more restrictive criterion than the one used for each augmentation separately but removes any potential misinformation on which augmentation is used for valid or not.

5.5 Can I Rely On You

The second part of the experiments is focused on testing the hypothesis that two of the most popular measures of XAI methods are not reliable when trying to use them to compare methods even within the same data domain. To be able to check if that hypothesis is true, we need to define how the measure should behave for it to be considered reliable.

Definition 5.1 *Given an XAI method A_1 and A_2 and the measure $S(A)$, measure S is considered reliable, when within the same data domain and model architecture, measures calculated for method A_1 always have the same relation to the measures calculated for the method A_2.*



(a) (b) (c)

Figure 5.8: An example of reliable measures (Fig. 5.8a, 5.8b) and unreliable measures (Fig. 5.8c)

To illustrate definition 5.1, let us consider an example in Figure 5.8. Each chart represents a relation between the measure score and the models' performance (it could be $F1$ or any valid metric). The ideal measure is shown in Figure 5.8a, where the relation between values for different attribution methods ($A1$, $A2$, and $A3$) remain the same with the change of models' performance. Changes in the absolute value are not relevant here. The second example (Fig. 5.8b) is also a valid measure but not as ideal as the first one. Relation between values is changed, but the order of the values is always the same. The third example (Fig. 5.8c) shows the behavior of the unreliable method. The relation between values is changing, and the order in which values for each method are presented is also changing. If there were only two methods in the third example ($A1$ and $A3$), we could assume the measure is reliable. That is the reason why when testing a given measure, we should use more data points.

Procedure

To properly validate measure Infidelity (Section: 4.2) and Sensitivity (Section: 4.1), each of them has to be tested on every trained model. That gives a total of 375 different data points for the measure (5 XAI methods $\times$ 5 datasets $\times$ 5 train data fractions $\times$ 3 models). Each data point consists of measures for every sample from the train dataset used for a particular model.

Remark. The Integrated Gradients (Section 3.4) method is not used in this experiment because computing the value of Sensitivity for this method requires more than 11GB of memory.

Infidelity uses a Gaussian noise with a $\mu = 0$ and $\sigma = 0.003$ to perturb images. Sensitivity creates 10 perturbed images with the help of the function that samples uniformly random from the $L_{Infinity}$ ball with 0.02 radius. The results from each data point are processed, and the mean and standard deviation values are calculated.

6 Results

6.1 Don't Augment Me[1]

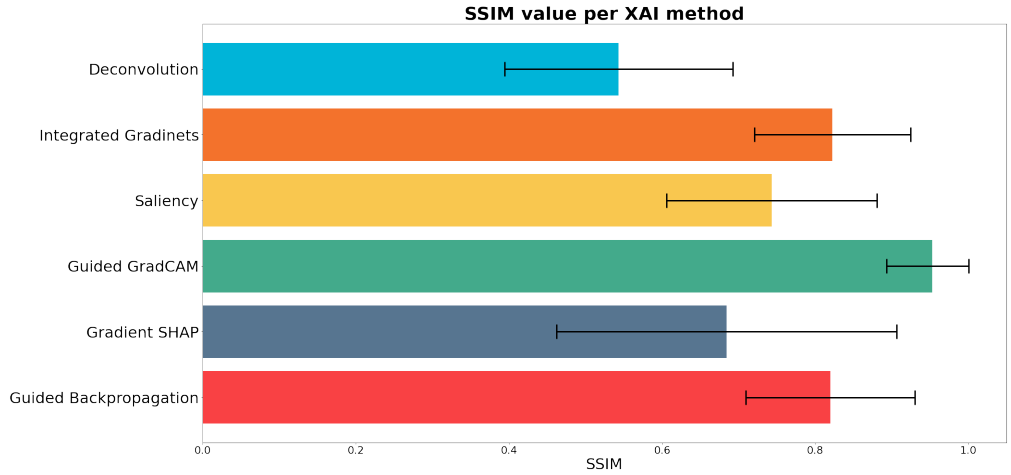
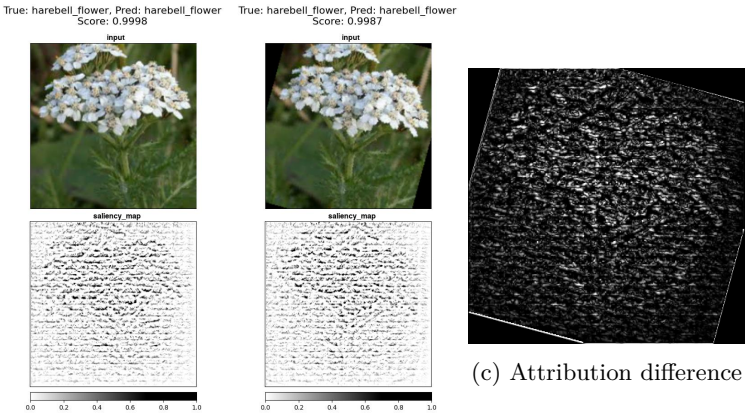


Figure 6.1: Average SSIM values per attribution method. Each bar represents a methods' mean value of SSIM. These mean values exclude examples when the classification score of the augmented image is outside the *threshold* of a non-augmented image. Because the score is independent of the attribution method, all mean values are calculated from exactly the same images and augmentations.

After calculating mean SSIM scores per tested method, we can clearly see large discrepancies between XAI methods. The method with the highest average score (Fig. 6.1) is *Guided GradCAM*, which achieved an average of 0.95 ($\sigma = 0.05$). On the other hand, we have *Deconvolution* with an average score of only 0.54 ($\sigma = 0.15$). In terms of SSIM value, this difference is larger than expected, and additional explanation is needed.

When creating an attribution, the Deconvolution method is using a reverse convolutional operation (building a reverse network, described in Section 3.1). Because the output from that network ends with a filter, it creates attributions from basic geometric shapes (see Fig. 6.2). This approach has a huge disadvantage when trying to compare attributions of rotated images (unless



(a) Original image (b) Rotation of -15° (c) Attribution difference

Figure 6.2: Visualisation of attributions for *Harebell Flower* image from Plants [28] tested on ResNet18 [25] trained on 100% of the training dataset. SSIM value for this pair: **0.2518**. The difference between attribution of rotated image (Fig. 6.2b) and the rotated attribution of the original image (Fig. 6.2a). Black color indicates the same values, and the white color indicates differences.

[1]Results from this section are also part of the paper submitted to the *"Thirty-fifth Conference on Neural Information Processing Systems", NeurIPS 2021* called *"Don't Augment Me: On the Robustness of Saliency Methods"*. At the moment of writing this thesis, the paper is under the review process. The paper was written in collaboration with Piotr Mazurek.

rotation is defined within a convolutional filter). From a wider perspective, when we look at Figures 6.2a and 6.2b, they look very similar. When we look closer and try to actually compare attribution from Figure 6.2b with rotated attribution from Figure 6.2a, we can see that lines are not aligned with each other (see Figure 6.2c).

This issue is better visualized when compare the mean value of Deconvolition from Figure 6.3 with the value from Figure 6.4. There is a clear difference between means (and also standard deviations), with the second Figure having the Deconvolution mean closer to the rest of the methods.

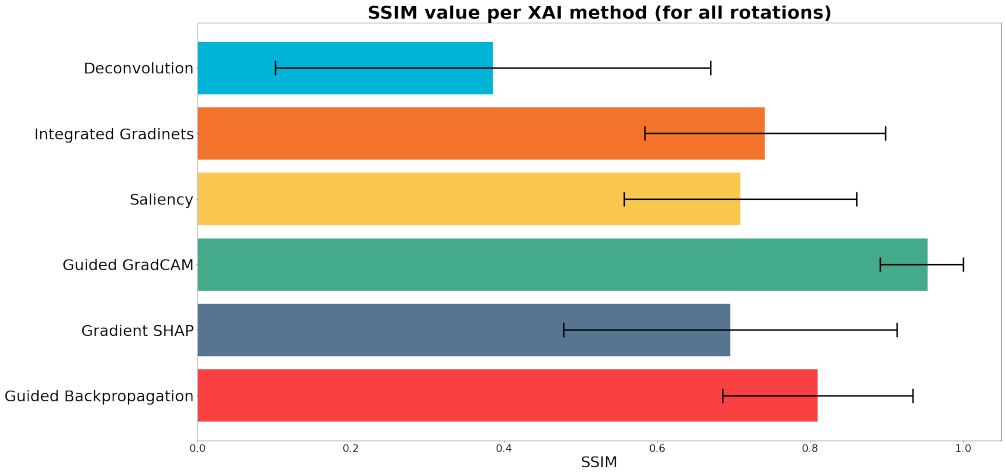


Figure 6.3: Average SSIM values per attribution method. Each bar represents a methods' mean value of SSIM. Values used to calculate the mean value are restricted to come only from images augmented by applying rotations. The detailed version of this chart is available in Figure B.1.

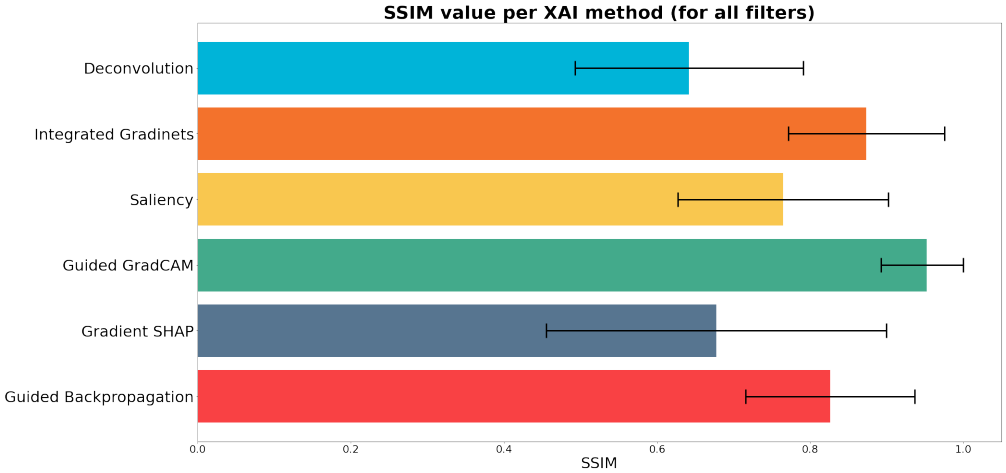
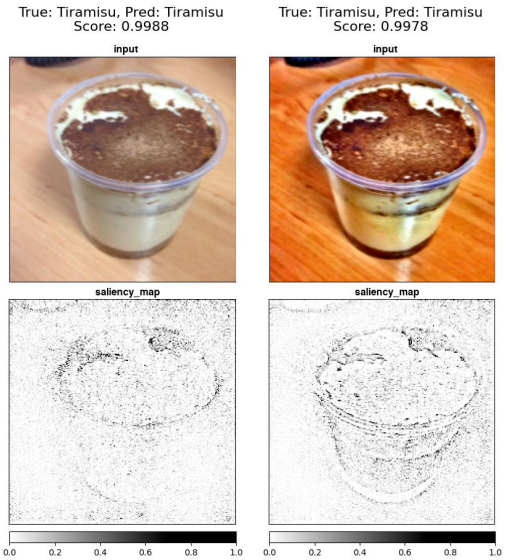


Figure 6.4: Average SSIM values per attribution method. Each bar represents a methods' mean value of SSIM. Values used to calculate the mean value are restricted to come only from images augmented by applying filters. The detailed version of this chart is available in Figure B.2.

The values of Guided GradCAM are closest to 1.0 (see Fig. 6.1), which would indicate the ideal method (no changes in attributions). A mean score of 0.95 might be confusing and provide a false sense of trust in the method. A good example can be seen in Figure 6.5. Two attributions have the SSIM score of 0.9357, but the visual comparison of those two images allows us to see major differences. The main difference is the number of details shown in Figure 6.5b. Both images achieve almost the same score, so the augmentation

does not cause the model to change its decision. An average score for the Guided GradCAM method is just 0.0143 away from the score achieved by this example.

To provide an overview of the used method, we can look at Figure 6.6. This overview does not contain all types of examples, which are stored separately and linked in Appendix B.1.3. Each pair of rows represents a set of augmentations (without rotations) per image per the XAI method. Augmentations names are visible above the examples, and all the predictions of the augmented images are within the *threshold* of the *"none"* image (first column). This is a slightly more restrictive list than the one used in calculating SSIM values. It is easier to pick examples by searching for the pair of images within the *threshold* than to find a set of 5 examples where all of them are within the *threshold*. That is why in this list, there are fewer visually different attributions.



(a) Original image (b) Local Norm

Figure 6.5: Visualisation of attributions done by GradCAM [59] for *Tiramisu* image from Food101 [16] tested on DenseNet121 [27] trained on 100% of the training dataset. SSIM value for this pair: **0.9357**.

Every set of augmentations contain at least one example where the attribution is significantly different than the original one. Lets start with the Deconvolution (*Rottweiler*). Two augmentations (*Freaky Details* and *Local Norm*) are causing the attribution method to display more details than the original one. Some of those details are related to the dog, but the rest is focusing on the window shutter. A similar problem with focusing on the different parts of the image can be seen in the *Dandellion* example generated with Integrated Gradients [67]. Only *Mighty Details* attribution stays the same (visually) when attributions for the rest of augmentations are changing their focus of the plant.

The main problem with these changes in the attribution is that they are nondeterministic. One augmentation might cause a method to focus on the details more, and the same augmentation might create the opposite behavior when applied on a different image (or even the same image but a different model). Some of the changes are minor and can be ignored (like the two just described), but there are more visually outstanding examples.

Let us look on the examples from Guided GradCAM [59], GradSHAP [15], and Saliency [61]. An example attributed by Guided GradCAM changes its attribution significantly when the *Mighty Details* filter is applied. This change is significant enough to change our understanding of what is important for a model in predicting the *sussex_spaniel* class. The attribution of the original image focuses mostly on the dog's head when the attribution of the augmented image focuses on the whole body. The same applies to the example attributed by GradSHAP. Here, the attribution of the original image points to the dog's nose when four other attributions of the augmented images are pointing to other parts of the animal (like ears and eyes). The example from the Saliency method shows different behavior. This time, when applying the *Local Norm* filter on the image, the attribution ignores some of the details in comparison with the original attribution.

The most relevant example is that generated with the help of GBP [64]. In that example, the model predicts the *black_widow* class for all images. The attribution of most is focused on the scene details and the character of Black Widow. This changes when *Local Norm* is applied, and the attribution starts to display a lot of details of the Hulk character. A shift like that might be dangerous when the understanding of the models' decisions is critical.

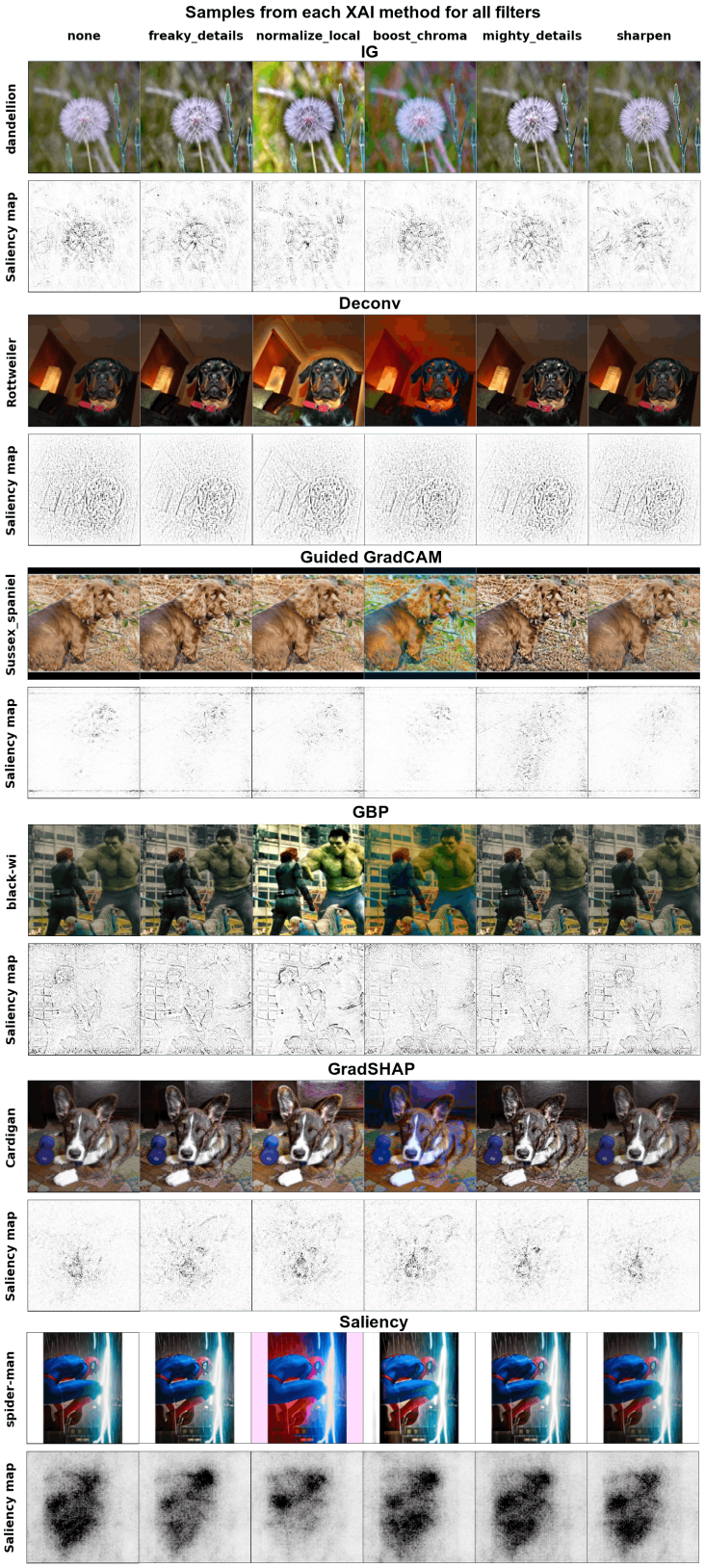


Figure 6.6: Augmentation attribution samples for every method. Samples are selected to all be within the *threshold* of the original image. Each row represents samples for all augmentations except rotations. More samples available in Appendix B.1.3

6.2 Can I Rely On You

6.2.1 Infidelity

Infidelity (see Section 4.2) was the first measure used in the experiments. This measure is producing results that are highly sensitive to the data domain and have similar values within the same dataset and model (see Figure 6.7). This forces us to compare values only within the same dataset and model architecture. Additionally, the comparison of values for the Deconvolution method is impossible, as shown in the same figure. The reason for that is the way how Deconvolution is calculating the value of attribution. Deconvolution is not using a softmax layer in the calculation, and therefore the absolute output attribution might be a couple of orders of magnitude higher than other methods. In the rest of the section, I am going to remove Deconvolution from the comparison.

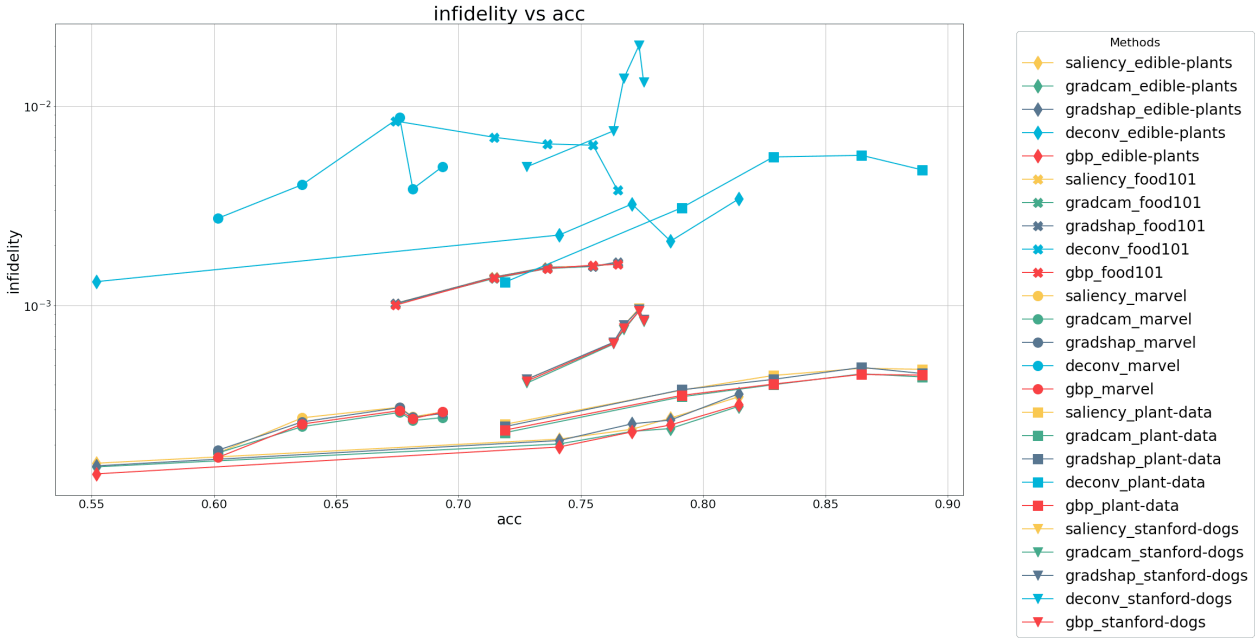
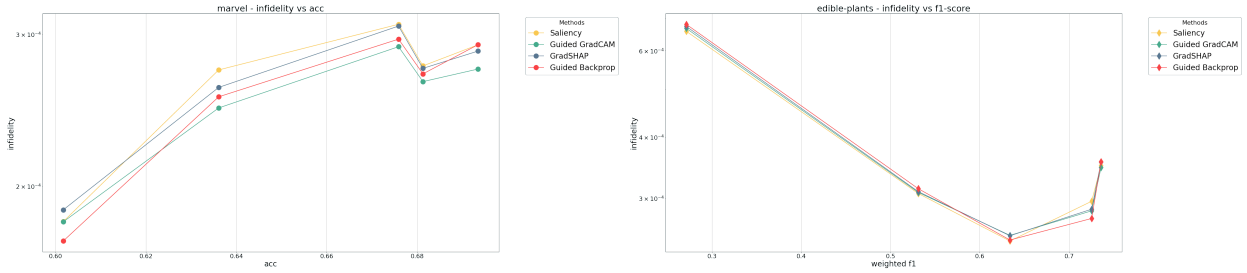


Figure 6.7: Infidelity scores on *ResNet18* architecture. All scores are the mean value for the particular model and are related to that models' accuracy (x-axis). Most of the scores for a dataset are within the same range, except for the Deconvolution method (see Section 3.1).

Because comparing the mean infidelity values from the combined chart is visually difficult (see Figure 6.7), a set of sub-charts was created. All sub-charts can be found in Appendix B.5, because the results on all of them are similar in terms of importance. We can discuss the measure using only a selected few (see Figure 6.8).

To check whether the infidelity meets the definition of reliable measure (see Def. 5.1), we need to compare relative changes of the mean scores for a set of methods within the same dataset. As shown in Figure 6.8a, the measure is not reliable according to the definition of reliable measure. The values of mean infidelity are not consistent between different model versions, and if we look at the *Guided Backprop* method, they even changing from being the best to being the worst (with an increase in the accuracy). The same result can be seen in Figure 6.8b, where methods are changing order base on the models' accuracy (e.g. Saliency is starting as the best method, then at around 0.78 accuracy is considered to be the worst, and at 0.81 accuracy is the second-best). This proves the hypothesis that Infidelity is not a reliable measure to compare the XAI methods.

Another case against using the infidelity can be seen if we draw the standard deviation of the infidelity measure for all models and methods. As shown in Figure 6.11, a standard deviation is usually similar to the



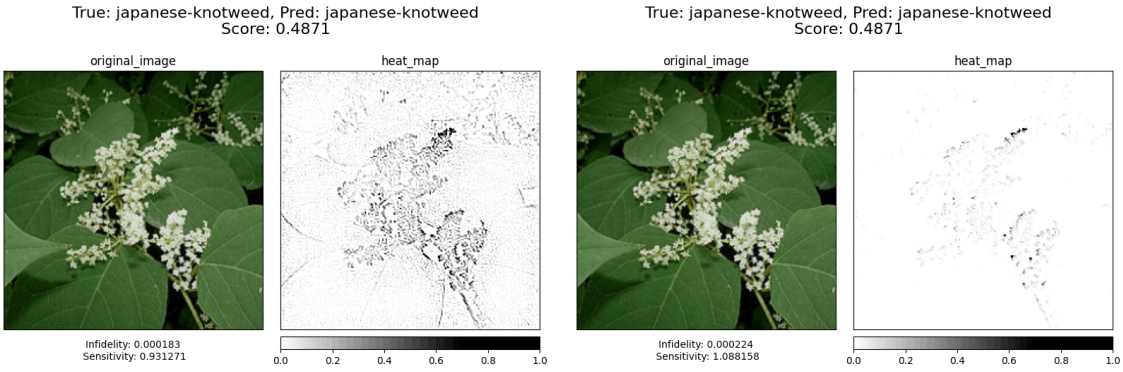
(a) Infidelity scores for *Marvel Heroes* dataset on *ResNet18* architecture

(b) Infidelity scores for *Edible wild plants* dataset on *EfficientNet B0* architecture

Figure 6.8: Sample results for Infidelity measure selected for visualizing the behavior of the measure with the change of model performance. Both charts are showing the relation between the mean infidelity on the test dataset and the models' accuracy on the same dataset.

mean value. This makes a single value unreliable when trying to compare it with another value. To visualize this unreliability of values, we can pick particular examples from the dataset. If we look at Figure 6.9, we can see that the value of infidelity for two examples indicated that the explanation provided by the GBP method is better (lower value of infidelity). Both examples are calculated using the same image and the same model version, so there is no discrepancy in models' behavior. Analyzing the attributions, we could agree that the GBP method provides a better explanation (Fig. 6.9a) because it shows more details from the input image than Guided GradCAM (Fig. 6.9b). This kind of example would be perfect when trying to convince that the measure is working correctly, but we can pick another image from the test dataset and use the same model and methods to compare infidelity values. This time the comparison is between values for the *mallow* class (Fig. 6.10). The infidelity value for GradCAM (Fig. 6.10b) is much lower than the value for GBP (6.10a), and once again, GBP shows more details from the input image. In this example, infidelity indicates that an explanation from GBP is much worse than an explanation from GradCAM, and this can be argued when looking at the attributions.

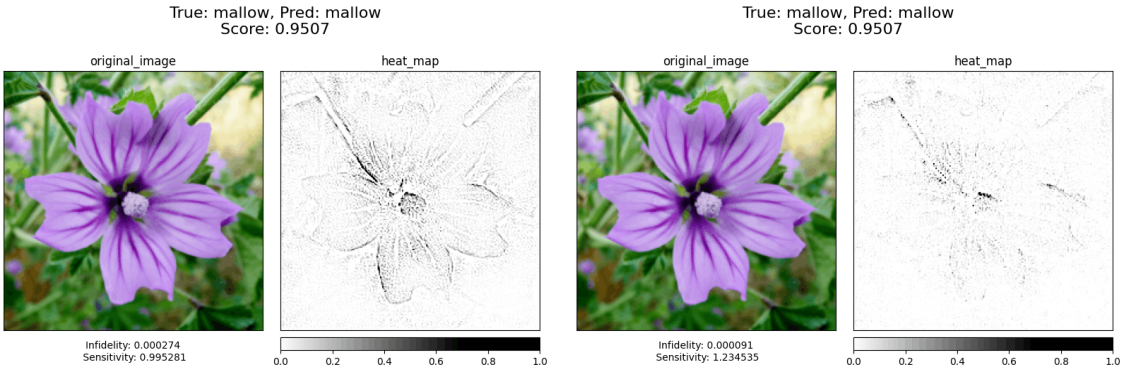
Remark. Indication of which explanation is better is just my subjective opinion. Comparing two explanations is a qualitative problem and can vary base on the person comparing the explanations. Provided examples are here to indicate the issue with values and not give an answer to which one is better.



(a) Guided Backprop

(b) Guided GradCAM

Figure 6.9: Attributions with assigned infidelity and sensitivity values for a given XAI method. All attributions are generated using *ResNet18* trained with 100% of training data. Results for the class *japanese-knotweed* from *Edible wild plants* [78].



(a) Guided Backprop (b) Guided GradCAM

Figure 6.10: Attributions with assigned infidelity and sensitivity values for a given XAI method. All attributions are generated using *ResNet18* trained with 100% of training data. Results for the class *mallow* from *Edible wild plants* [78].

Figure 6.11: Infidelity scores (with standard deviation) on *DenseNet121* architecture. All scores are the mean value for the particular model and are related to that models' predicted scores (x-axis). Each data point is a mean value of the same amount of samples per dataset. Amounts differ between datasets to always split the results into 30 equally-sized bins.

6.2.2 Sensitivity

The sensitivity (see Section 4.1) measure is more complicated in terms of interpreting the result. Unlike infidelity, which suppose to tell us which XAI method is better, sensitivity measure tells us which XAI method is more prone to change its decision with slight perturbation in the input data. We should keep that in mind when discussing this measure. This time, the Deconvolution method is included in the results because the sensitivity does not have the same issue with an absolute value of attribution as infidelity.

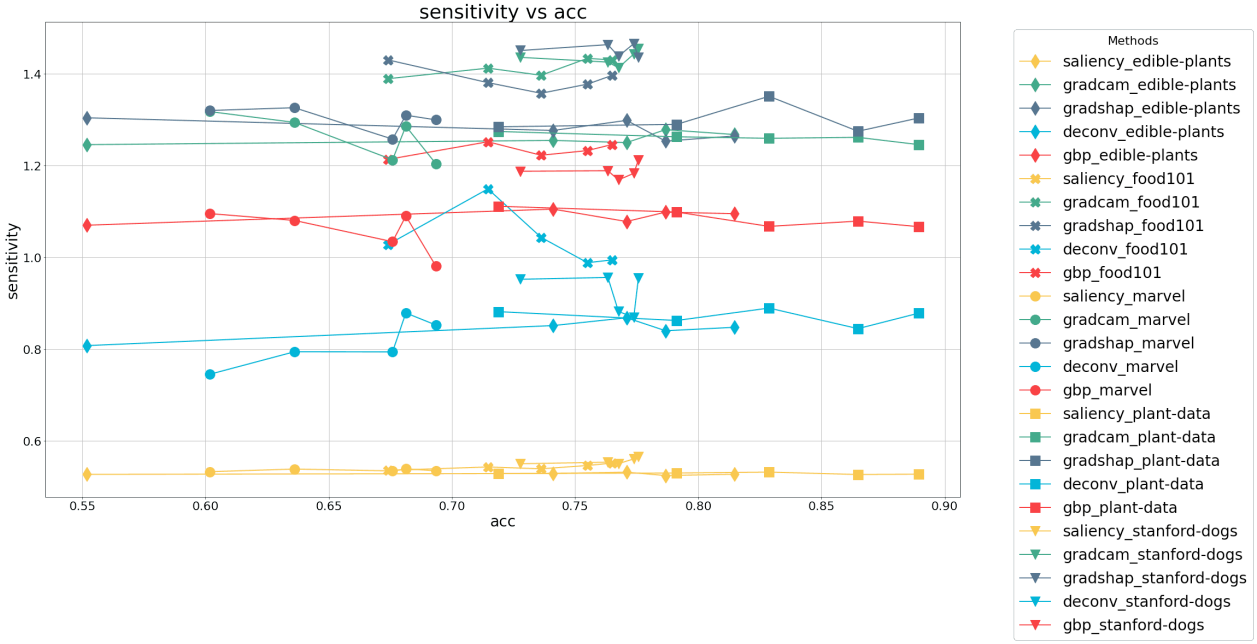
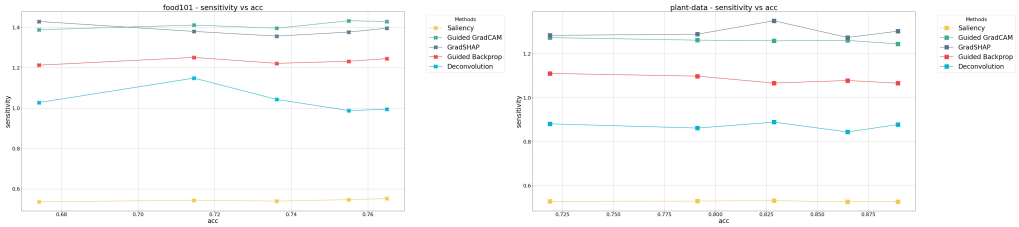


Figure 6.12: Sensitivity scores on *ResNet18* architecture. All scores are the mean value for the particular model and are related to that models' accuracy (x-axis).

The combined results for the sensitivity shown in Figure 6.12 differ from those for infidelity in Figure 6.7. This time the data range is not related to the dataset that much, and we can see a distinction between different XAI methods in the form of colored layers. Based on this example, we can decide which XAI method is the most and the less sensitive to input perturbation. It is even easier when we use values that come from the same dataset (see Figure 6.13). The order of the mean values for XAI methods is preserved between datasets. Even with a small irregularity in Figure 6.13a between GradCAM and GradSHAP values at the low end of the models' accuracy, we can agree that sensitivity is a far better measure in terms of reliability (Def. 5.1) than infidelity.



(a) Sensitivity scores for *Food101* dataset on *ResNet18* architecture

(b) Sensitivity scores for *Plants* dataset on *ResNet18* architecture

Figure 6.13: Sample results for the Sensitivity measure selected for visualizing the behavior of the measure with the change of model performance. Both charts show the relation between the mean sensitivity on the test dataset and the models' accuracy on the same dataset. More individual charts available in Appendix B.2.2.

Unfortunately, this reliability does not apply when trying to compare values between architectures. Figure 6.14 shows a different order of XAI methods according to the mean sensitivity value. Saliency is still the least sensitive method, but the rest of the methods did not keep their position. Comparing the orders between two architectures allows us to confirm that hypothesis saying that this measure is not capable of reliable compare XAI methods is true.

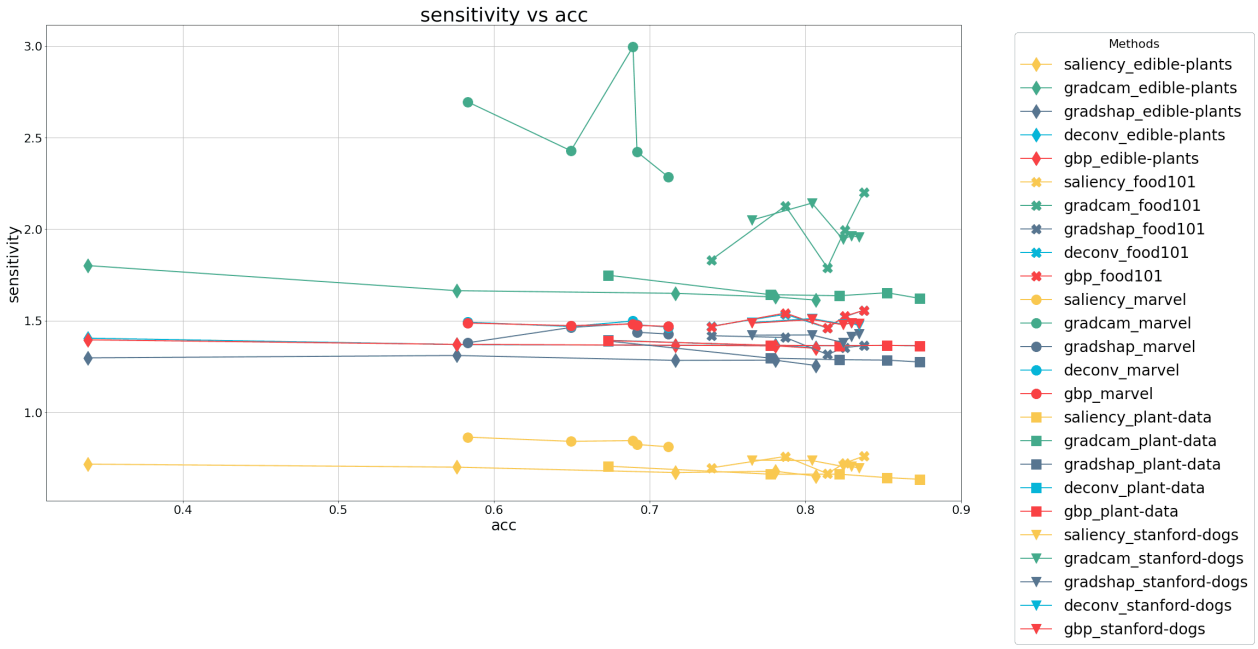
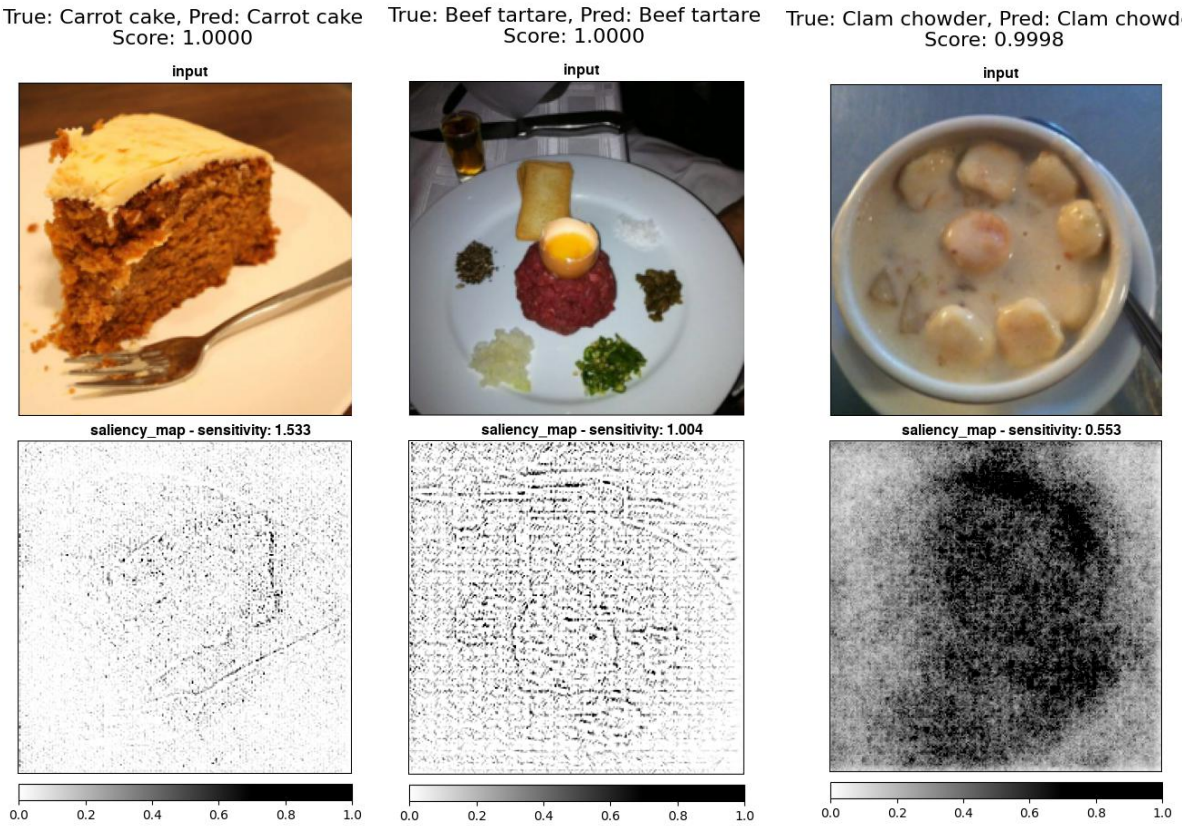


Figure 6.14: Sensitivity scores on *EfficientNet B0* architecture. All scores are the mean value for the particular model and are related to that models' accuracy (x-axis).

As mentioned at the beginning of this section, sensitivity shows how much a method is sensitive to perturbations in the input data. There is a pattern that can be spotted when looking at the generated examples. Usually, the lower the sensitivity score is, the less accurate the attribution appears to be. As an example, let us compare sensitivity scores for three methods shown in Figure 6.15. An example with the highest sensitivity is a GradCAM (Fig. 6.15a, which has less attribution in total. The method with the lowest sensitivity is always the Saliency (Fig. 6.15c) which has more noisy attribution than other methods. This might be caused by the way the sensitivity is calculated. The end value is a total change in attribution after applying perturbation. Methods with fewer attributions (working more like edge detectors) are going to have more attribution change because if the edge is changed, then there is a large difference in value. Methods similar to Saliency, even if the attribution changes, it usually does not change that much because of the initial noise. That might explain differences in mean sensitivity between different models. The attributions produced by the same XAI method usually differ between architectures, and therefore the value of sensitivity is different.

Additionally, we have to look at the standard deviation of the sensitivity scores (Fig. 6.16). This chart significantly differs from the one showing standard deviation for infidelity. The standard deviation for mean sensitivity measure per score shows that some methods (like Saliency) have low variance in sensitivity when methods with more detailed attributions (like GradCAM) have higher variance. That is related to some methods showing less noise in the attributions. The standard deviation also is dependent on the architecture, and sometimes the values of the deviation might vary a lot. A good example is a chart created for EfficientNet B0 architecture in Figure 6.17. Some of the values are outside the value range (GradCAM).



(a) GradCAM (b) Deconvolution (c) Saliency

Figure 6.15: Sensitivity scores for different methods produced for *ResNet18* architecture and examples from *Food101* dataset: *Beef tartare* 6.15b, *Carrot cake* 6.15a, *Clam chowder* 6.15c.

Figure 6.16: Sensitivity scores (with standard deviation) on *ResNet18* architecture. All scores are the mean value for the particular model and are related to that models' predicted scores (x-axis). Each data point is a mean value of the same amount of samples per dataset. Amounts differ between datasets to always split the results into 30 equally-sized bins.

Figure 6.17: Sensitivity scores (with standard deviation) on *EfficientNet B0* architecture. All scores are the mean value for the particular model and are related to that models' predicted scores (x-axis). Each data point is a mean value of the same amount of samples per dataset. Amounts differ between datasets to always split the results into 30 equally-sized bins.

7 Discussion and Conclusion

The discussion in this chapter focuses separately on each part of the experiments. This division is due to other aspects of the problem being touched. The first part, named *"Don't Augment Me"*, focuses on the problem with XAI methods, when the second part, titled *"Can I Rely On You"*, discusses a broader problem of being able to compare these methods. The chapter ends with the conclusion and presents the idea of possible future exploration in the field.

7.1 Don't Augment Me

The common practice in designing machine learning models is to use predefined building blocks, and with the evolution of frameworks, it is going to become even more accessible for non-researchers to develop and deploy models. Because explainability is becoming a requirement, some of those developers are going to use available XAI methods base on their popularity. The usual process of understanding the models' decision is to check the input attribution for a set of test examples, and base on that attribution, decide if the model is working correctly. As shown in this thesis, that kind of approach can provide us with insufficient information to understand our model. Real-world examples might provide a different explanation than the ones from the test set (see Figure 5.7 and the *cardigan* attributions for original and locally normalized images).

The first experiment results gave us an understanding of how the current XAI methods behave when the input of the model is augmented with easily accessible augmentations. Base on similarity comparison, we could deduct that the method with the highest robustness is Guided GradCAM. This method achieved the highest SSIM scores with the lowest standard deviation, but like mentioned in section 6.1, the GradCAM method is far from being ideal (see Fig. 6.5). Every method tested in this study appears to be influenced by the perturbations that have little to no effect on the models' prediction. This is concerning because the methods' behavior is nondeterministic. When applied to the actual model, we cannot be sure if the result given by the method is the correct one unless we test every single one.

The positive aspect of the experiments is that the robustness of the method is changing in a minimal way between different types of augmentations. The substantial change is seen only in the case of Deconvolution (see Fig, 6.3), where the method has a significant drop in the performance. The drop was caused by the way Deconvolution presents the attributions and the difference between the attribution of the rotated image and the rotated attribution of the original image. This issue could be fixed by using a different methodology than initially assumed. However, the SSIM metric is not an ideal metric and can be improved to match the human visual perception better. Even with those flaws, the results were consistent across different methods and augmentations.

7.2 Can I Rely On You

Base on results from the first part of the experiments, we can agree that having a reliable and comparable measure is beneficial. With that measure, we could compare existing XAI methods and improve new ones. Two of the most popular XAI methods were tested in the second part of the experiments. The results differed between the methods, and it is better to discuss them separately.

Infidelity was the first measure tested against multiple models and datasets. The results showed that this measure is extremely dependent on the dataset, model architecture, and even model performance. It failed the reliability test on any level, and the results showed that values returned by the measure could not be compared even within the same trained model. Even if it has a strong, sensible, theoretical background, the absolute value of infidelity cannot be used to compare the two values. The relative value of infidelity

for method A when compared with method B changes within the same model architecture and dataset just after the model performance changes (see Fig. 6.8). A method that was considered the best (lowest infidelity score) after the model is better trained on the same dataset achieves the worst result. That kind of behavior is unacceptable in the measure. Even values of the infidelity from the same trained model and methods are inconsistent (see a comparison of infidelity values between Fig. 6.9 and Fig. 6.10.

Sensitivity measure achieves far better results than infidelity, but unlike infidelity, it is not a measure of quality but the measure of robustness. As shown in Figures 6.12 and 6.13, the measure is able to compare different methods within the same model architecture, and there is less value change between different accuracy. However, the absolute value of sensitivity is still a meaningless value because the value changes between architectures. The mean value for the ResNet18 cannot be compared with the value for EfficientNet B0 (compare Fig. 6.12 and Fig. 6.14). The order of methods also differs between different architectures. Base on that information, we can assume that sensitivity could be used as a measure within the same architecture, and because ML practitioners are usually reusing a limited set of predefined architectures, there could be a set of XAI methods that work best for a specific architecture. During the experiments, the sensitivity seems to give the lower scores to the methods with less detailed attributions (see Fig. 6.15). Attribution methods that work more like edge detectors (GBP or GradCAM) tend to receive larger values of sensitivity.

7.3 Conclusion

XAI is still a new branch of the field but already gotten implemented in some of the popular libraries [34, 1], and even by a top-tier cloud provider [21]. Package developers and cloud service providers are doing their best to help the engineers understand their models, but as shown in the number of papers and this thesis, current methods are far from being a ready solution. Without understanding the particular method's drawbacks, the method might provide a false sense of understanding for the developer and cause a severe problem for the end-user.

As shown in this study, the reliability and robustness of XAI methods are a major issue. Not having a sensible measure to compare those methods is one of the concerns because we as a community are not able to validate if a newly released method is better than the previous one. This also does not help the authors of the methods because they have to rely on qualitative measures, like manual attribution comparison, to determine the quality of their work. This could cause a problem with cherry-picked examples presented in scientific papers. To provide the confirmation that our new method works, we only have to find a handful of examples to include in a paper instead of calculating a numerical value that determines the quality of the method.

7.4 Future Work

This thesis focused mostly on exposing the problem with XAI methods and available measures. With the results, new ideas for improving the current state of the field emerged. One of those ideas is to create a reliable measure for XAI methods which is based on structural similarities of the attribution for visual tasks. SSIM used in this thesis is just a basic structural similarity metric and already have multiple improved version. This should also fix the issue with methods similar to Deconvolution, which had terrible scores on rotated images. Another idea is to improve the measure of infidelity to be able to compare not only the reliability of the method but the quality as well. Eventually, the goal is to create a set of tests integrated with one of the popular XAI frameworks, which can be performed on any method to determine its quality and robustness.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] A. Adadi, M. Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[3] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, B. Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

[4] K. B. Ahmed, G. M. Goldgof, R. Paul, D. B. Goldgof, L. O. Hall. Discovery of a generalization gap of convolutional neural networks on covid-19 x-rays classification. *IEEE Access*, 2021.

[5] M. Ancona, E. Ceolini, C. Öztireli, M. Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[6] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, i in. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[7] M. Asif, Y. Orenstein. Deepselex: inferring dna-binding preferences from ht-selex data using multi-class cnns. *Bioinformatics*, 36(Supplement_2):i634–i642, 2020.

[8] T. O. Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[9] O. Biran, C. Cotton. Explanation and justification in machine learning: A survey. *IJCAI-17 workshop on explainable AI (XAI)*, wolumen 8, strony 8–13, 2017.

[10] J. G. Carbonell, R. S. Michalski, T. M. Mitchell. An overview of machine learning. *Machine learning*, strony 3–23, 1983.

[11] J. W. Creswell. *Educational research: Planning, conducting, and evaluating quantitative*. Prentice Hall Upper Saddle River, NJ, 2002.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, F.-F. Li. Imagenet: a large-scale hierarchical image database. strony 248–255, 06 2009.

[13] S. Dodge, L. Karam. A study and comparison of human and deep learning recognition performance under visual distortions. *2017 26th international conference on computer communication and networks (ICCCN)*, strony 1–7. IEEE, 2017.

[14] L. Edwards, M. Veale. Slave to the algorithm: Why a right to an explanation is probably not the remedy you are looking for. *Duke L. & Tech. Rev.*, 16:18, 2017.

[15] G. Erion, J. D. Janizek, P. Sturmfels, S. Lundberg, S.-I. Lee. Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*, 2019.

[16] ETH. Food images (food-101). https://www.kaggle.com/kmader/food41, 2018. Accessed: 2021-05-01.

[17] Ethan. Marvel heroes. https://www.kaggle.com/hchen13/marvel-heroes, 2019. Accessed: 2021-05-01.

[18] W. Fedus, B. Zoph, N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

[19] R. C. Fong, A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *Proceedings of the IEEE International Conference on Computer Vision*, strony 3429–3437, 2017.

[20] A. Ghorbani, A. Abid, J. Zou. Interpretation of neural networks is fragile. *Proceedings of the AAAI Conference on Artificial Intelligence*, wolumen 33, strony 3681–3688, 2019.

[21] Google Cloud Blog. Introduction to vertex explainable ai for vertex ai. https://cloud.google.com/vertex-ai/docs/explainable-ai/overview#ig. Accessed: 2021-05-01.

[22] B. Graham. Sparse 3d convolutional neural networks. *arXiv preprint arXiv:1505.02890*, 2015.

[23] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[24] R. Hahnloser, R. Sarpeshkar, M. Mahowald, R. J. Douglas, S. Seung. Digital selection and analog amplification co-exist in an electronic circuit inspired by neocortex. *Nature*, 405(6789):947–951, 2000.

[25] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. corr abs/1512.03385 (2015), 2015.

[26] S. Hooker, D. Erhan, P.-J. Kindermans, B. Kim. A benchmark for interpretability methods in deep neural networks. *arXiv preprint arXiv:1806.10758*, 2018.

[27] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 4700–4708, 2017.

[28] M. Jawad, M. Safwan, H. Usman, R. Khan. Plants dataset[99 classes]. https://www.kaggle.com/muhammadjawad1998/plants-dataset99-classes?select=Plant_Data, 2020. Accessed: 2021-05-01.

[29] J. Jiménez-Luna, F. Grisoni, N. Weskamp, G. Schneider. Artificial intelligence in drug discovery: Recent advances and future perspectives. *Expert Opinion on Drug Discovery*, strony 1–11, 2021.

[30] F. Keller. Computational foundations of cognitive science, convolutions and kernels. http://www.inf.ed.ac.uk/teaching/courses/cfcs1/lectures/cfcs_l15.pdf, 2010.

[31] A. Khosla, N. Jayadevaprakash, B. Yao, L. Fei-Fei. Stanford dogs dataset. https://www.kaggle.com/jessicali9530/stanford-dogs-dataset, 2019. Accessed: 2021-05-01.

[32] P.-J. Kindermans, K. Schütt, K.-R. Müller, S. Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv preprint arXiv:1611.07270*, 2016.

[33] A. Kleppe, O.-J. Skrede, S. De Raedt, K. Liestøl, D. J. Kerr, H. E. Danielsen. Designing deep learning studies in cancer diagnostics. *Nature Reviews Cancer*, strony 1–13, 2021.

[34] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, i in. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

[35] G. Kovács, P. Alonso, R. Saini. Challenges of hate speech detection in social media. *SN Computer Science*, 2(2):1–15, 2021.

[36] A. Krizhevsky, I. Sutskever, G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[37] Y. LeCun, Y. Bengio, i in. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[38] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[39] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[40] M. Lin, Q. Chen, S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[41] A. Lind, E. Akbarian, S. Olsson, H. Nåsell, O. Sköldenberg, A. S. Razavian, M. Gordon. Artificial intelligence for the classification of fractures around the knee in adults according to the 2018 ao/ota classification system. *PloS one*, 16(4):e0248809, 2021.

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

[43] B. Liu, M. Zhou, X. Li, X. Zhang, Q. Wang, L. Liu, M. Yang, D. Yang, Y. Guo, Q. Zhang, i in. Interrogation of gender disparity uncovers androgen receptor as the transcriptional activator for oncogenic mir-125b in gastric cancer. *Cell death & disease*, 12(5):1–22, 2021.

[44] S. Lundberg, S.-I. Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[45] J. Luo, W. Zhang, J. Su, F. Xiang. Hexagonal convolutional neural networks for hexagonal grids. *IEEE Access*, 7:142738–142749, 2019.

[46] A. L. Maas, A. Y. Hannun, A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. *Proc. icml*, wolumen 30, strona 3. Citeseer, 2013.

[47] B. Mamandipoor, F. Frutos-Vivar, O. Peñuelas, R. Rezar, K. Raymondos, A. Muriel, B. Du, A. W. Thille, F. Ríos, M. González, i in. Machine learning predicts mortality based on analysis of ventilation parameters of critically ill patients: multi-centre validation. *BMC Medical Informatics and Decision Making*, 21(1):1–12, 2021.

[48] W. S. McCulloch, W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[49] T. M. Mitchell, i in. Machine learning. 1997.

[50] H. Panwar, P. Gupta, M. K. Siddiqui, R. Morales-Menendez, P. Bhardwaj, V. Singh. A deep learning and grad-cam based color visualization approach for fast detection of covid-19 cases using chest x-ray and ct-scan images. *Chaos, Solitons & Fractals*, 140:110190, 2020.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala. Pytorch: An imperative style, high-performance deep learning library. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett, redaktorzy, *Advances in Neural Information Processing Systems 32*, strony 8024–8035. Curran Associates, Inc., 2019.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[53] T. Pianpanit, S. Lolak, P. Sawangjai, T. Sudhawiyangkul, T. Wilaiprasitporn. Parkinson's disease recognition using spect image and interpretable ai: A tutorial. *IEEE Sensors Journal*, 2021.

[54] N. Rank, B. Pfahringer, J. Kempfert, C. Stamm, T. Kühne, F. Schoenrath, V. Falk, C. Eickhoff, A. Meyer. Deep-learning-based real-time prediction of acute kidney injury outperforms human predictive performance. *NPJ digital medicine*, 3(1):1–12, 2020.

[55] N. Rethmeier, N. O. Şerbetci, S. Möller, R. Roller. Efficare: Better prognostic models via resource-efficient health embeddings. *medRxiv*, 2020.

[56] D. E. Rumelhart, G. E. Hinton, R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[57] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, i in. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[58] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[59] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE international conference on computer vision*, strony 618–626, 2017.

[60] J.-W. Sidhom, I. J. Siddarthan, B.-S. Lai, A. Luo, B. C. Hambley, J. Bynum, A. S. Duffield, M. B. Streiff, A. R. Moliterno, P. Imus, i in. Deep learning for diagnosis of acute promyelocytic leukemia via recognition of genomically imprinted morphologic features. *NPJ precision oncology*, 5(1):1–8, 2021.

[61] K. Simonyan, A. Vedaldi, A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

[62] D. Sinanc, U. Demirezen, Ş. Sağıroğlu. Explainable credit card fraud detection with image conversion. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 10(1):63–76, 2021.

[63] D. Smilkov, N. Thorat, B. Kim, F. Viégas, M. Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[64] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[65] O. Sturman, L. von Ziegler, C. Schläppi, F. Akyol, M. Privitera, D. Slominski, C. Grimm, L. Thieren, V. Zerbi, B. Grewe, i in. Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. *Neuropsychopharmacology*, 45(11):1942–1952, 2020.

[66] P. Sturmfels, S. Lundberg, S.-I. Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

[67] M. Sundararajan, A. Taly, Q. Yan. Axiomatic attribution for deep networks. *International Conference on Machine Learning*, strony 3319–3328. PMLR, 2017.

[68] M. Tan, Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*, strony 6105–6114. PMLR, 2019.

[69] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, strony 6411–6420, 2019.

[70] C. Tomasi, R. Manduchi. Bilateral filtering for gray and color images. *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, strony 839–846. IEEE, 1998.

[71] D. Tschumperlé, S. Fourey. Boost chromacity filter. https://natron.readthedocs.io/en/rb-2.3/plugins/eu.gmic.BoostChromaticity.html#gmic-boost-chromaticity-node. Accessed: 2021-05-01.

[72] D. Tschumperlé, S. Fourey. Freaky details filter. https://natron.readthedocs.io/en/rb-2.3/plugins/eu.gmic.FreakyDetails.html. Accessed: 2021-05-01.

[73] D. Tschumperlé, S. Fourey. G'mic (greyc's magic for image computing): A full-featured open-source framework for image processing. https://gmic.eu. Accessed: 2021-05-01.

[74] D. Tschumperlé, S. Fourey. Local normalization filter. https://natron.readthedocs.io/en/rb-2.3/plugins/eu.gmic.LocalNormalization.html. Accessed: 2021-05-01.

[75] D. Tschumperlé, S. Fourey. Mighty details filter. https://natron.readthedocs.io/en/rb-2.3/plugins/eu.gmic.MightyDetails.html. Accessed: 2021-05-01.

[76] D. Tschumperlé, S. Fourey. Sharpening filter. https://natron.readthedocs.io/en/rb-2.3/plugins/eu.gmic.SharpenOctaveSharpening.html. Accessed: 2021-05-01.

[77] T. Van Craenendonck, B. Elen, N. Gerrits, P. De Boever. Systematic comparison of heatmapping techniques in deep learning in the context of diabetic retinopathy lesion detection. *Translational Vision Science & Technology*, 9(2):64–64, 2020.

[78] G. Verzea. Edible wild plants. https://www.kaggle.com/gverzea/edible-wild-plants, 2018. Accessed: 2021-05-01.

[79] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[80] M. A. Wani, F. A. Bhat, S. Afzal, A. I. Khan. *Advances in deep learning*. Springer, 2020.

[81] P. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974.

[82] W. Xiao, X. Huang, J. H. Wang, D. R. Lin, Y. Zhu, C. Chen, Y. H. Yang, J. Xiao, L. Q. Zhao, J.-P. O. Li, i in. Screening and identifying hepatobiliary diseases through deep learning using ocular images: a prospective, multicentre study. *The Lancet Digital Health*, 3(2):e88–e97, 2021.

[83] C.-K. Yeh, C.-Y. Hsieh, A. S. Suggala, D. I. Inouye, P. Ravikumar. On the (in) fidelity and sensitivity for explanations. *arXiv preprint arXiv:1901.09392*, 2019.

[84] M. D. Zeiler, R. Fergus. Visualizing and understanding convolutional networks, 2013.

[85] M. D. Zeiler, G. W. Taylor, R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. *2011 International Conference on Computer Vision*, strony 2018–2025. IEEE, 2011.

[86] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba. Learning deep features for discriminative localization. *Proceedings of the IEEE conference on computer vision and pattern recognition*, strony 2921–2929, 2016.

List of Figures

List of Tables

A Open Science Framework

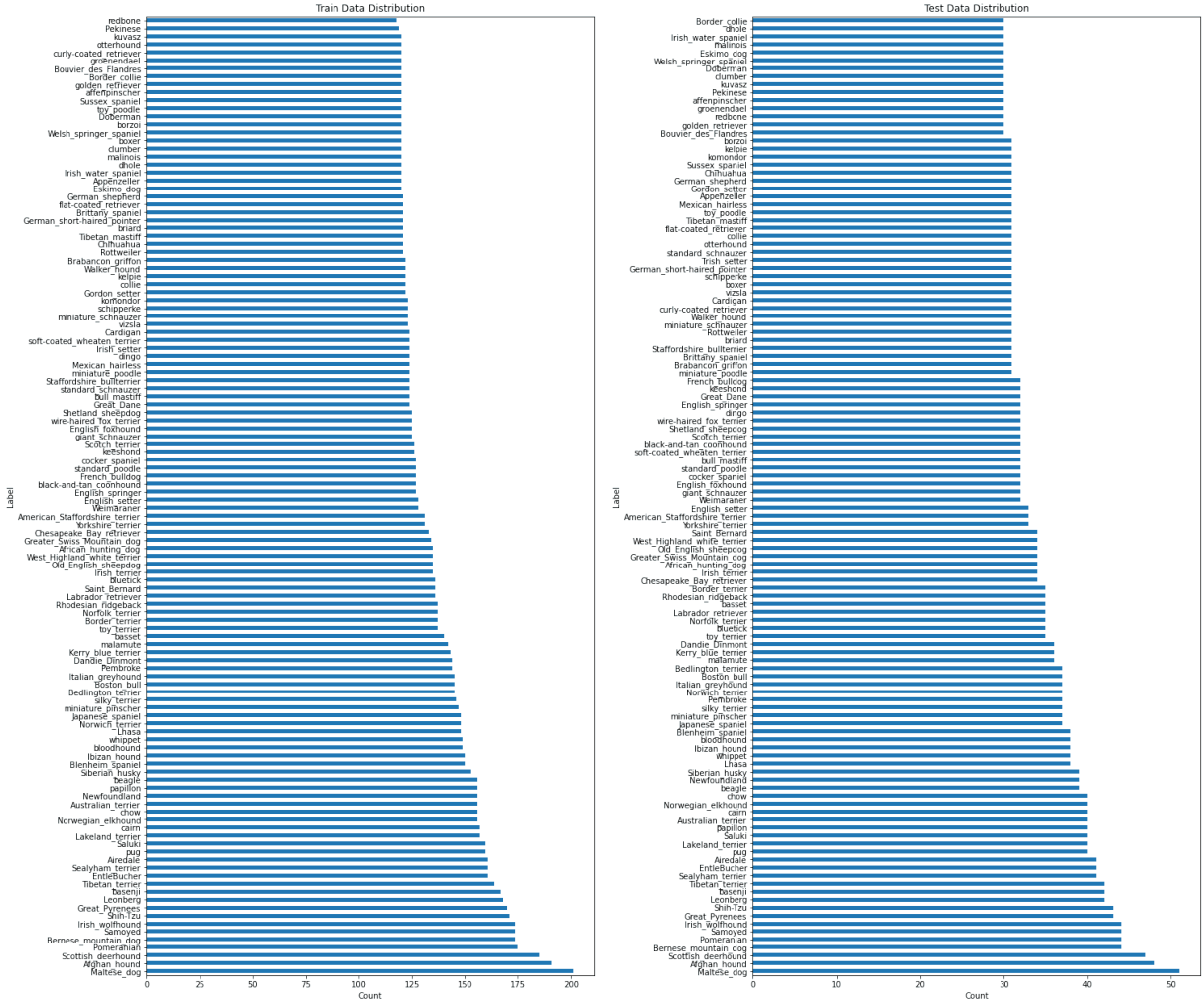
Source code is available at https://github.com/burnpiro/xai-correlation. Repository includes detailed instruction on how to run and reproduce experiments. The source code has an additional Wiki page available at https://github.com/burnpiro/xai-correlation/wiki with some of the results, links to trained models and additional instructions.

Requirements

- Python 3.8
- git
- Graphic card with at least 11GB of free memory (tested on GForce GTX 1080Ti)

A.1 Datasets

A.1.1 Stanford Dogs



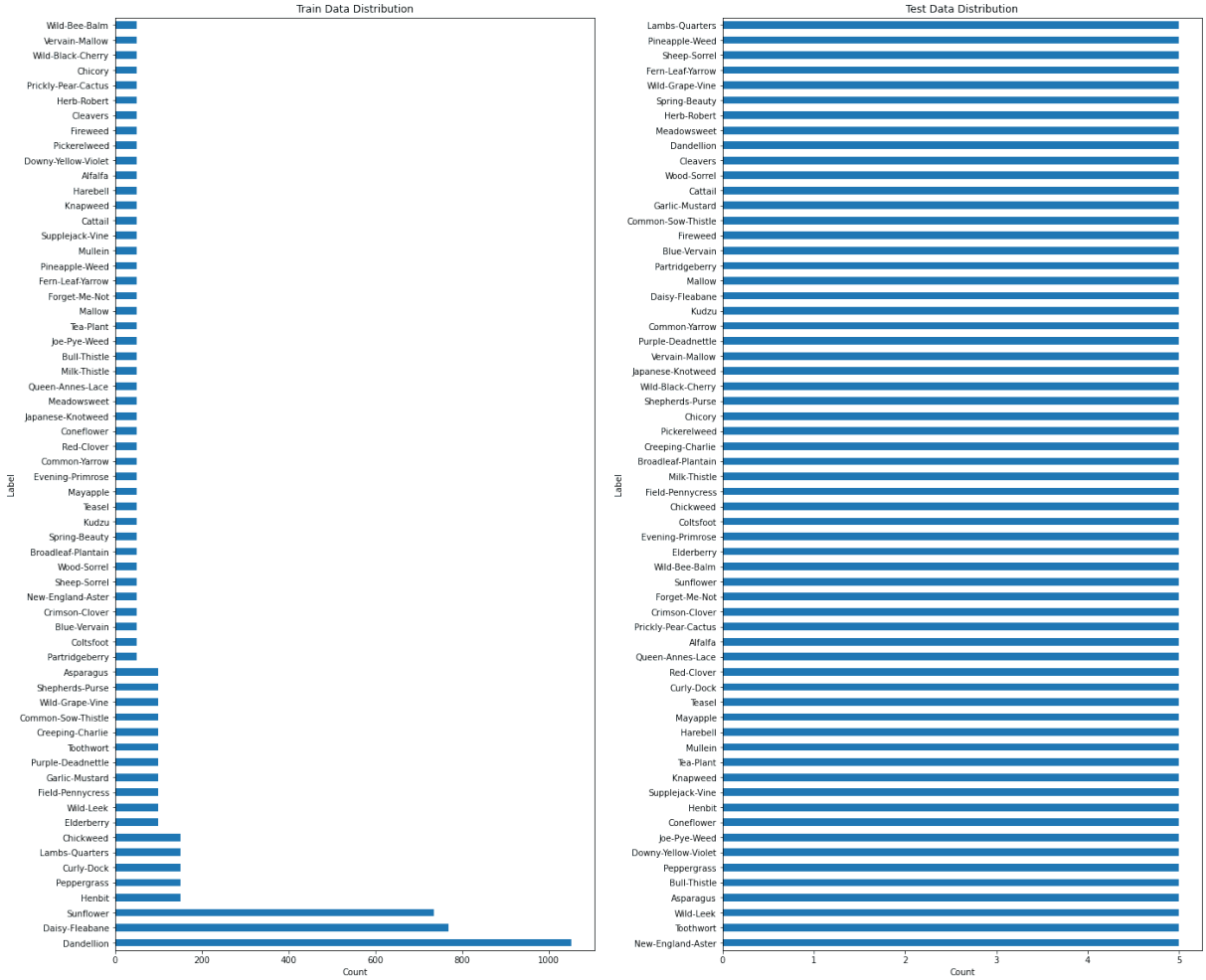
(a) Train dataset (b) Test dataset

Figure A.1: Class distribution in *Stanford Dogs*[31] dataset. Data available at http://vision.stanford.edu/aditya86/ImageNetDogs/.

A.1.2 Food 101

Data available at https://www.kaggle.com/dansbecker/food-101. Each class has 750 examples in training dataset and 250 examples in test dataset. 101 classes in total.

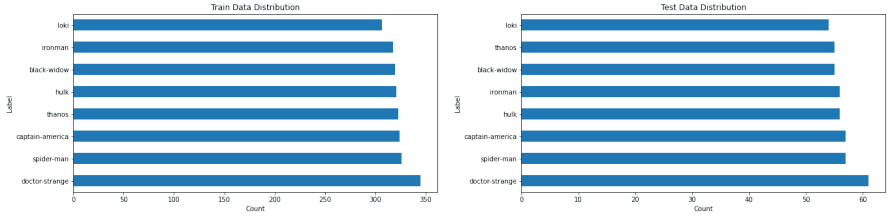
A.1.3 Edible wild plants



(a) Train dataset (b) Test dataset

Figure A.2: Class distribution in *Edible wild plants*[78] dataset. Data available at https://www.kaggle.com/gverzea/edible-wild-plants.

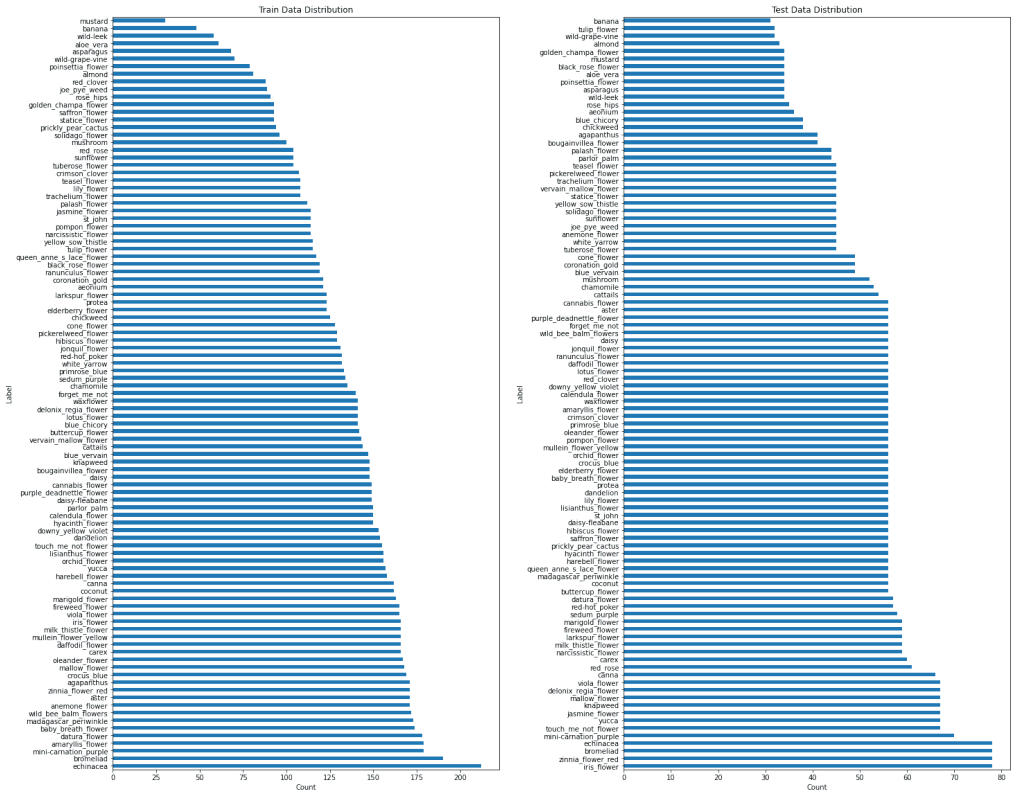
A.1.4 Marvel Heroes



(a) Train dataset (b) Test dataset

Figure A.3: Class distribution in *Marvel Heroes*[17] dataset. Data available at https://www.kaggle.com/hchen13/marvel-heroes.

A.1.5 Plants



(a) Train dataset (b) Test dataset

Figure A.4: Class distribution in *Plants*[28] dataset. Data available at https://www.kaggle.com/muhammadjawad1998/plants-dataset99-classes?select=Plant_Data.

A.2 Models

Trained models are available at https://drive.google.com/drive/folders/1WVmokp5vaPwOGhpEC6elGVE0aM7fPqNU?usp=sharing. Step by step instruction on how to train models is available at https://github.com/burnpiro/xai-correlation/wiki/Train-and-Eval-Models.

Table A.1: Models' F1 and Accuracy scores

ID	Dataset	Model	Data Fraction	Acc	F1
0	Edible wild plants	ResNet18	100%	0.8146	0.7671
1	Edible wild plants	ResNet18	80%	0.7867	0.7348
2	Edible wild plants	ResNet18	60%	0.7709	0.7168
3	Edible wild plants	ResNet18	40%	0.7411	0.6634
4	Edible wild plants	ResNet18	20%	0.5519	0.4914
5	Food101	ResNet18	100%	0.7650	0.7632
6	Food101	ResNet18	80%	0.7550	0.7532
7	Food101	ResNet18	60%	0.7362	0.7326
8	Food101	ResNet18	40%	0.7146	0.7122
9	Food101	ResNet18	20%	0.6741	0.6689
10	Marvel Heroes	ResNet18	100%	0.6813	0.6798
11	Marvel Heroes	ResNet18	80%	0.6935	0.6923
12	Marvel Heroes	ResNet18	60%	0.6759	0.6693
13	Marvel Heroes	ResNet18	40%	0.6361	0.6301

14	Marvel Heroes	ResNet18	20%	0.6017	0.5960
15	Plants	ResNet18	100%	0.8895	0.8860
16	Plants	ResNet18	80%	0.8648	0.8590
17	Plants	ResNet18	60%	0.8287	0.8240
18	Plants	ResNet18	40%	0.7912	0.7855
19	Plants	ResNet18	20%	0.7189	0.7010
20	Stanford Dogs	ResNet18	100%	0.7738	0.7662
21	Stanford Dogs	ResNet18	80%	0.7757	0.7681
22	Stanford Dogs	ResNet18	60%	0.7676	0.7618
23	Stanford Dogs	ResNet18	40%	0.7633	0.7567
24	Stanford Dogs	ResNet18	20%	0.7278	0.7208
25	Edible wild plants	EfficientNet B0	100%	0.8067	0.7359
26	Edible wild plants	EfficientNet B0	80%	0.7805	0.7253
27	Edible wild plants	EfficientNet B0	60%	0.7165	0.6338
28	Edible wild plants	EfficientNet B0	40%	0.5760	0.5312
29	Edible wild plants	EfficientNet B0	20%	0.3392	0.2713
30	Food101	EfficientNet B0	100%	0.8374	0.8368
31	Food101	EfficientNet B0	80%	0.8253	0.8248
32	Food101	EfficientNet B0	60%	0.8140	0.8131
33	Food101	EfficientNet B0	40%	0.7871	0.7870
34	Food101	EfficientNet B0	20%	0.7395	0.7385
35	Marvel Heroes	EfficientNet B0	100%	0.7119	0.7041
36	Marvel Heroes	EfficientNet B0	80%	0.6918	0.6831
37	Marvel Heroes	EfficientNet B0	60%	0.6892	0.6856
38	Marvel Heroes	EfficientNet B0	40%	0.6493	0.6444
39	Marvel Heroes	EfficientNet B0	20%	0.5831	0.5774
40	Plants	EfficientNet B0	100%	0.8734	0.8703
41	Plants	EfficientNet B0	80%	0.8523	0.8487
42	Plants	EfficientNet B0	60%	0.8218	0.8165
43	Plants	EfficientNet B0	40%	0.7774	0.7664
44	Plants	EfficientNet B0	20%	0.6734	0.6422
45	Stanford Dogs	EfficientNet B0	100%	0.8343	0.8289
46	Stanford Dogs	EfficientNet B0	80%	0.8293	0.8253
47	Stanford Dogs	EfficientNet B0	60%	0.8240	0.8187
48	Stanford Dogs	EfficientNet B0	40%	0.8041	0.7972
49	Stanford Dogs	EfficientNet B0	20%	0.7654	0.7549
50	Edible wild plants	DenseNet 121	100%	0.8777	0.8357
51	Edible wild plants	DenseNet 121	80%	0.8520	0.8145
52	Edible wild plants	DenseNet 121	60%	0.8062	0.7502
53	Edible wild plants	DenseNet 121	40%	0.7718	0.7210
54	Edible wild plants	DenseNet 121	20%	0.6576	0.5957
55	Food101	DenseNet 121	100%	0.8465	0.8447
56	Food101	DenseNet 121	80%	0.8356	0.8346
57	Food101	DenseNet 121	60%	0.8213	0.8189
58	Food101	DenseNet 121	40%	0.8023	0.8005
59	Food101	DenseNet 121	20%	0.7585	0.7555
60	Marvel Heroes	DenseNet 121	100%	0.7196	0.7127
61	Marvel Heroes	DenseNet 121	80%	0.6919	0.6911
62	Marvel Heroes	DenseNet 121	60%	0.7164	0.7101
63	Marvel Heroes	DenseNet 121	40%	0.6853	0.6739
64	Marvel Heroes	DenseNet 121	20%	0.6777	0.6582
65	Plants	DenseNet 121	100%	0.9016	0.8995
66	Plants	DenseNet 121	80%	0.8829	0.8799
67	Plants	DenseNet 121	60%	0.8567	0.8516

68	Plants	DenseNet 121	40%	0.8233	0.8182
69	Plants	DenseNet 121	20%	0.7494	0.7397
70	Stanford Dogs	DenseNet 121	100%	0.8290	0.8244
71	Stanford Dogs	DenseNet 121	80%	0.8179	0.8130
72	Stanford Dogs	DenseNet 121	60%	0.8153	0.8099
73	Stanford Dogs	DenseNet 121	40%	0.8042	0.7989
74	Stanford Dogs	DenseNet 121	20%	0.7932	0.7846

B Supplementary Results

B.1 Don't Augment Me - appendix

B.1.1 SSIM attribution ranges

Table B.1: SSIM ranges for attribution methods for ResNet18

Attribution \ Dataset	Edible plants	Food 101	Marvel	Plants	Stanford Dogs
Deconvolution	0.784622	0.423285	0.473029	0.682504	1.283490
Integrated Gradients	0.421971	1.306827	0.611498	0.887589	1.565911
Saliency	0.047666	0.045500	0.067406	0.094456	0.060711
Grad-Cam	0.000047	0.000051	0.000022	0.000085	0.000064
Grad-Shap	0.145404	0.093035	0.102947	0.285072	0.147860
Guided Backprop	0.079774	0.069185	0.050991	0.126553	0.152545

Table B.2: SSIM ranges for attribution methods for DenseNet121

Attribution \ Dataset	Edible plants	Food 101	Marvel	Plants	Stanford Dogs
Deconvolution	4677.117	80.11916	3037.732	1715.939	2044.188
Integrated Gradients	1.099644	2.577734	0.997656	0.816421	2.090753
Saliency	0.068720	0.048571	0.052243	0.068942	0.149736
Grad-Cam	0.000278	0.000187	0.000510	0.000395	0.002106
Grad-Shap	0.142074	0.119778	0.162216	0.143393	0.345822
Guided Backprop	0.400560	0.218944	0.512460	0.720738	1.565153

Table B.3: SSIM ranges for attribution methods for EfficientNet B0

Attribution \ Dataset	Edible plants	Food 101	Marvel	Plants	Stanford Dogs
Deconvolution	0.078107	0.025042	0.030603	0.055335	0.071512
Integrated Gradients	0.732469	5.059399	0.810054	1.551912	2.329165
Saliency	0.095080	0.032253	0.043658	0.140426	0.136111
Grad-Cam	0.000028	0.000005	0.000007	0.000042	0.000037
Grad-Shap	0.464270	0.054290	0.154112	0.280005	0.352823
Guided Backprop	0.068730	0.037462	0.030622	0.066075	0.102107

B.1.2 Detailed SSIM results

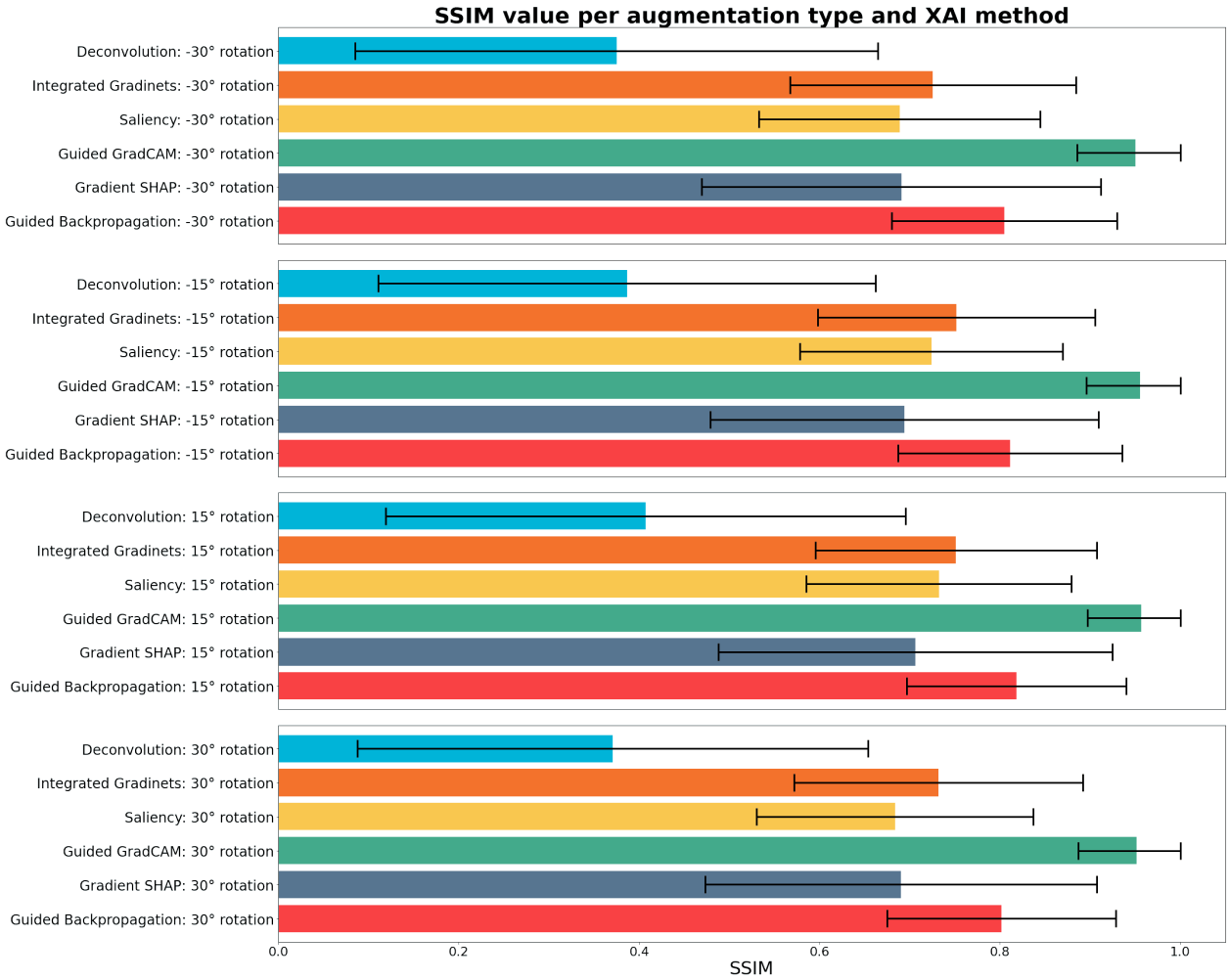


Figure B.1: Average SSIM values per attribution method and augmentation type (rotations). Each bar represents a methods' mean value of SSIM. Values used to calculate the mean value are restricted to come only from images augmented by applying specific rotations.

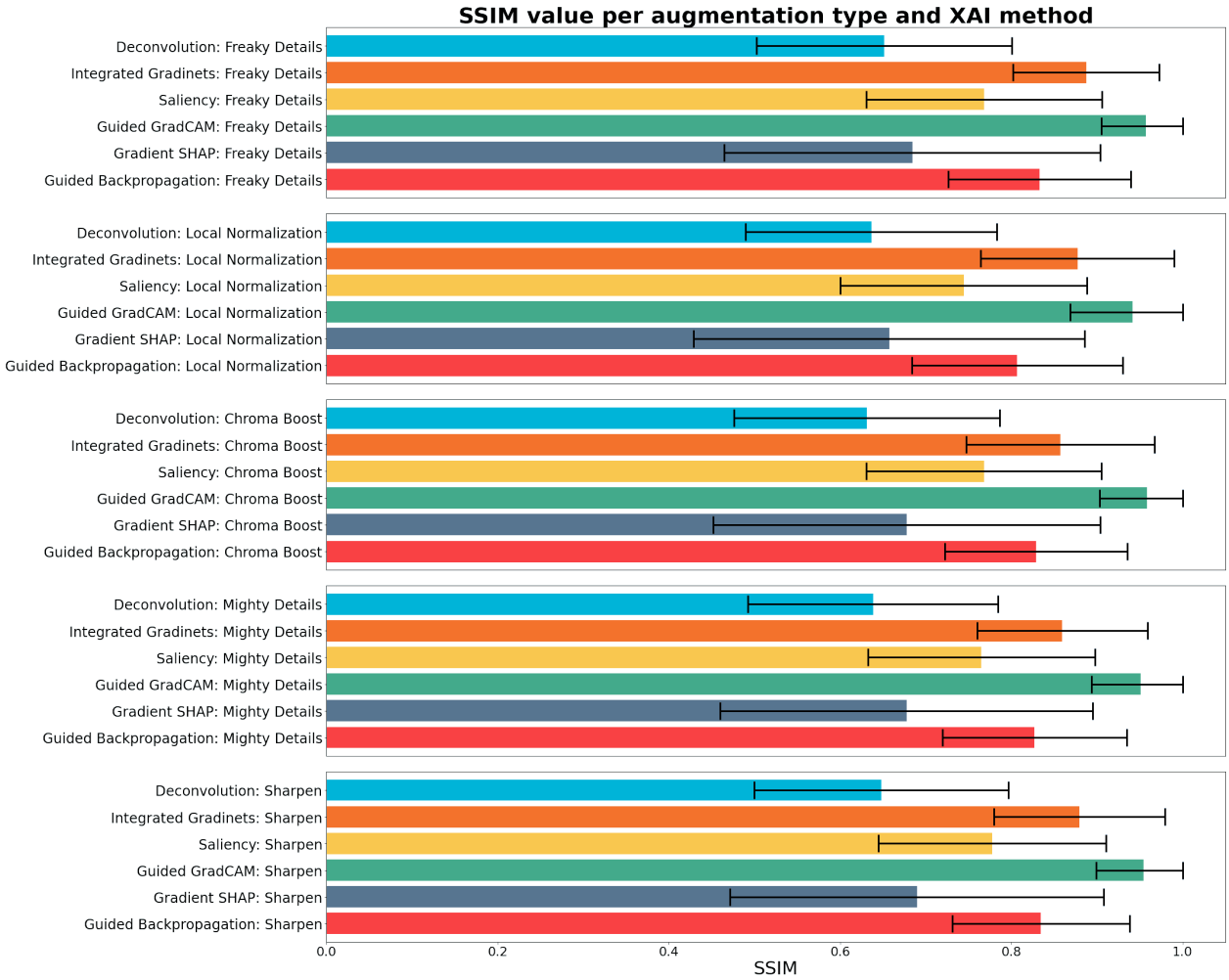


Figure B.2: Average SSIM values per attribution method and augmentation type (filters). Each bar represents a methods' mean value of SSIM. Values used to calculate the mean value are restricted to come only from images augmented by applying specific filter.

B.1.3 Attributions samples

Because the amount of attribution examples is to large to fit them in the appendix, they are available at https://drive.google.com/drive/folders/1Vp3MYRg8j-6ZAfDePeT5kHYPu2Rv6r4t?usp=sharing (total number of files equals 97990). File structure is following the pattern `{Augmentation Type}/{Dataset}/{Model version}/{XAI m`
Each image has a filename containing:

`{index}-{sample id}-{class id}-{augmentation method}-{true class}-{predicted class}.png`

Combined attribution samples are available at
https://drive.google.com/drive/folders/1lnNQwH3-H1cHiNOEcmnd4OhSScDSuzt0?usp=sharing.

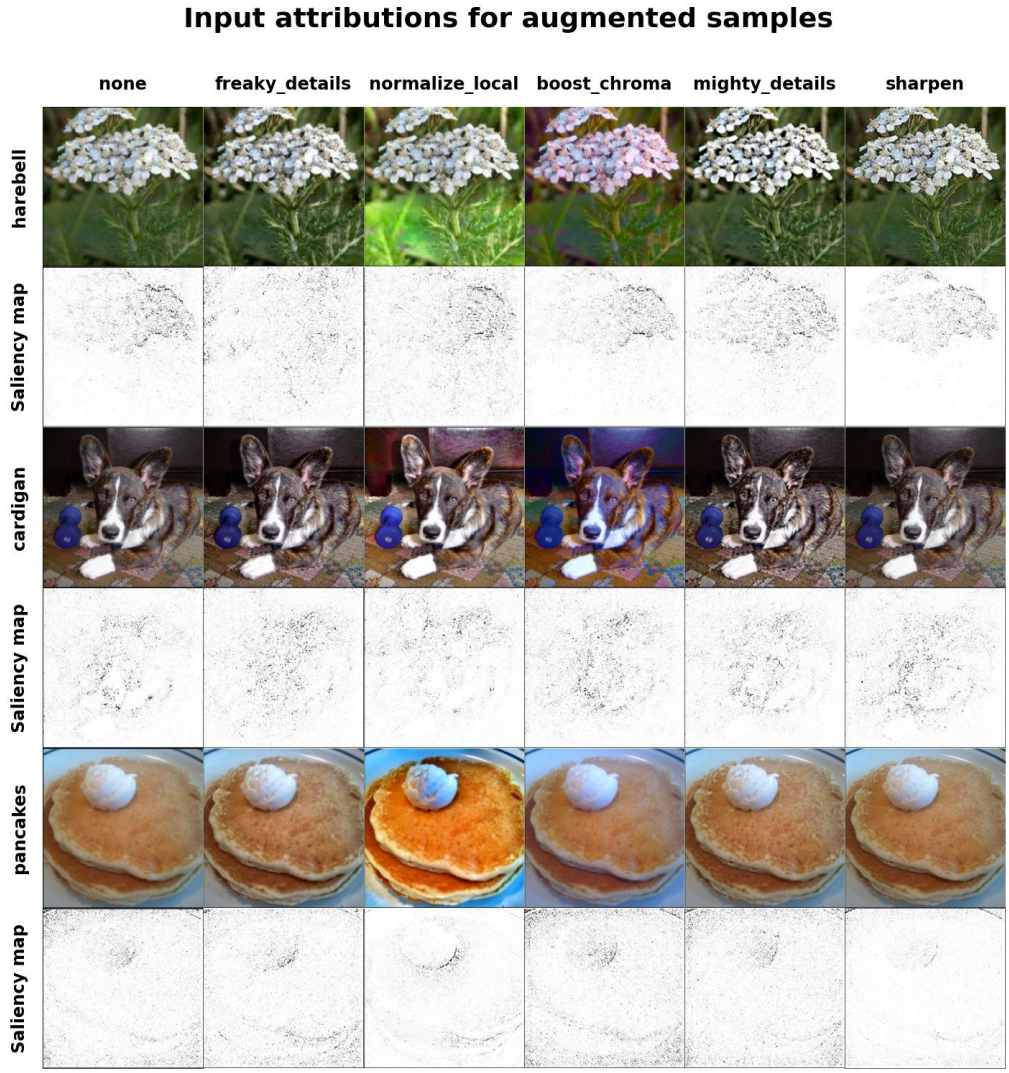


Figure B.3: **Attribution comparison of inputs with applied filters.** Figure shows three distinct images (*harebell_flower*, *cardigan*, *pancakes*) with different filters. All attributions done by Guided GradCAM.

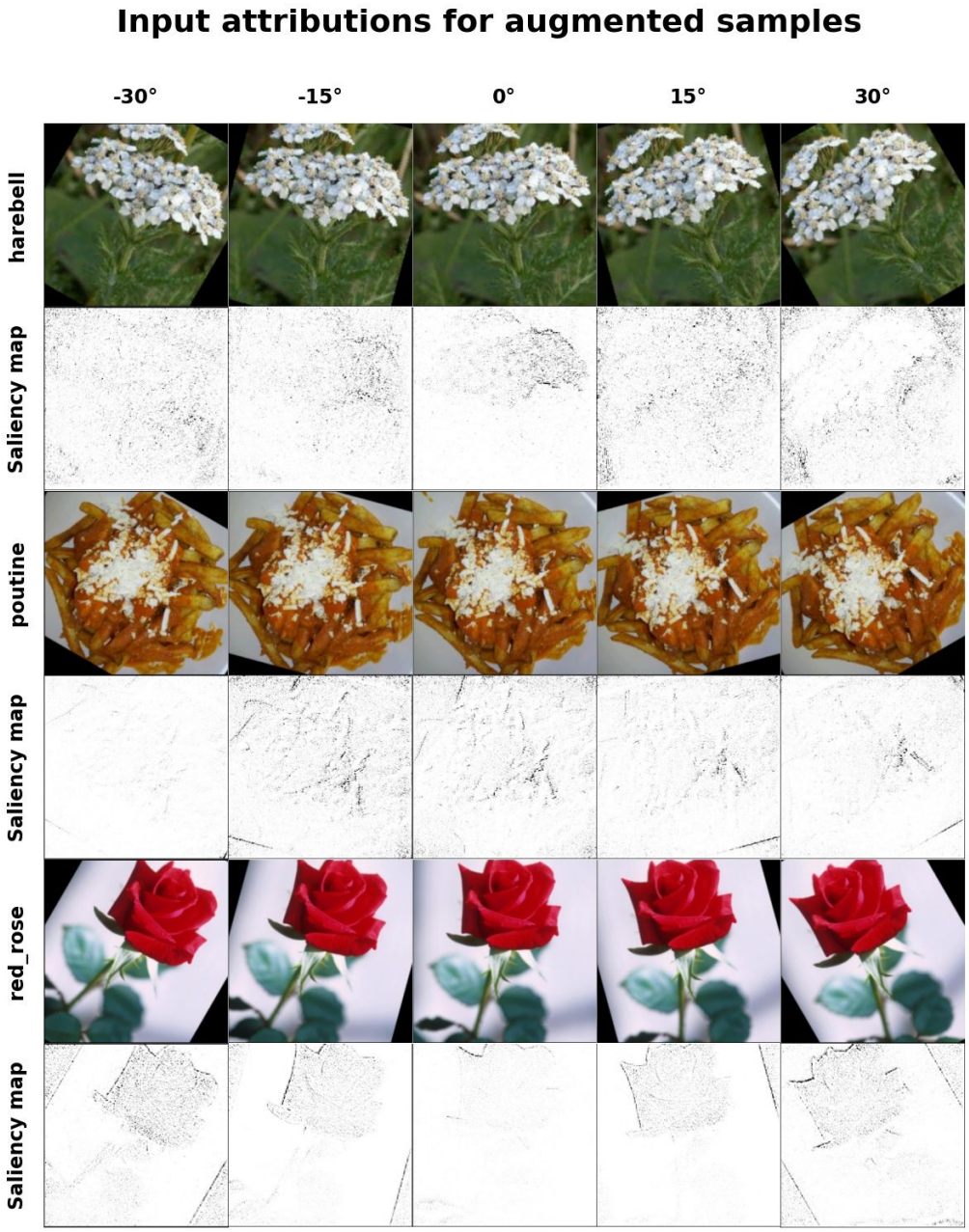


Figure B.4: **Attribution comparison of rotated inputs.** Figure shows three distinct images (*harebell_flower*, *poutine*, *red_rose*) with different rotations. All attributions done by Guided GradCAM.

B.2 Can I Rely On You - appendix

Attribution Examples

Because the amount of attribution examples is to large to fit them in the appendix, they are available at https://drive.google.com/drive/folders/177nMz2Y21Z505Eb5r8NZwAMQoMfX6S5f?usp=sharing (total number of files equals 23150). File structure is following the pattern `{{Dataset}/{Model version}/{XAI method}`. Each image has a filename containing:

`{index}-{true class}-{predicted class}.png`

B.2.1 Infidelity - combined scores

Individual scores can be find at https://drive.google.com/drive/folders/1_0PaXj2QbuW5DyAFjlrKAUjJ_iaoEn9a?usp=sharing. Folder "**individual**" contains separated scores per dataset/model. Folder "**combined**" contains combined scores. Folder "**confidence**" contains infidelity values with standard deviation values.

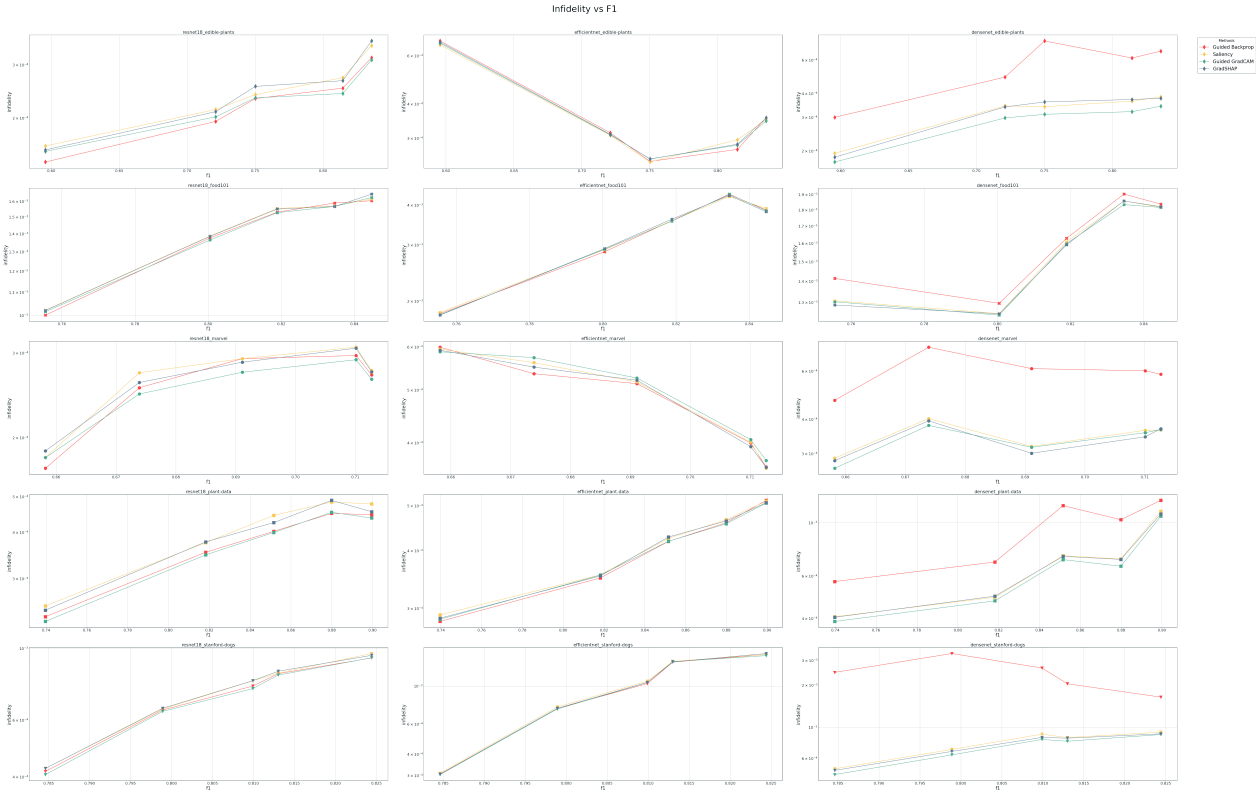


Figure B.5: Combined Infidelity scores against F1 scores. Each row represents a dataset, each column represents an architecture (see individual subplot titles)

B.2.2 Sensitivity - combined scores

Individual scores can be find at https://drive.google.com/drive/folders/12eYJSZFMfI2FZhXQSuwJQ6I8w65EU25e?usp=sharing. Folder "**individual**" contains separated scores per dataset/model. Folder "**combined**" contains combined scores. Folder "**confidence**" contains sensitivity values with standard deviation values.

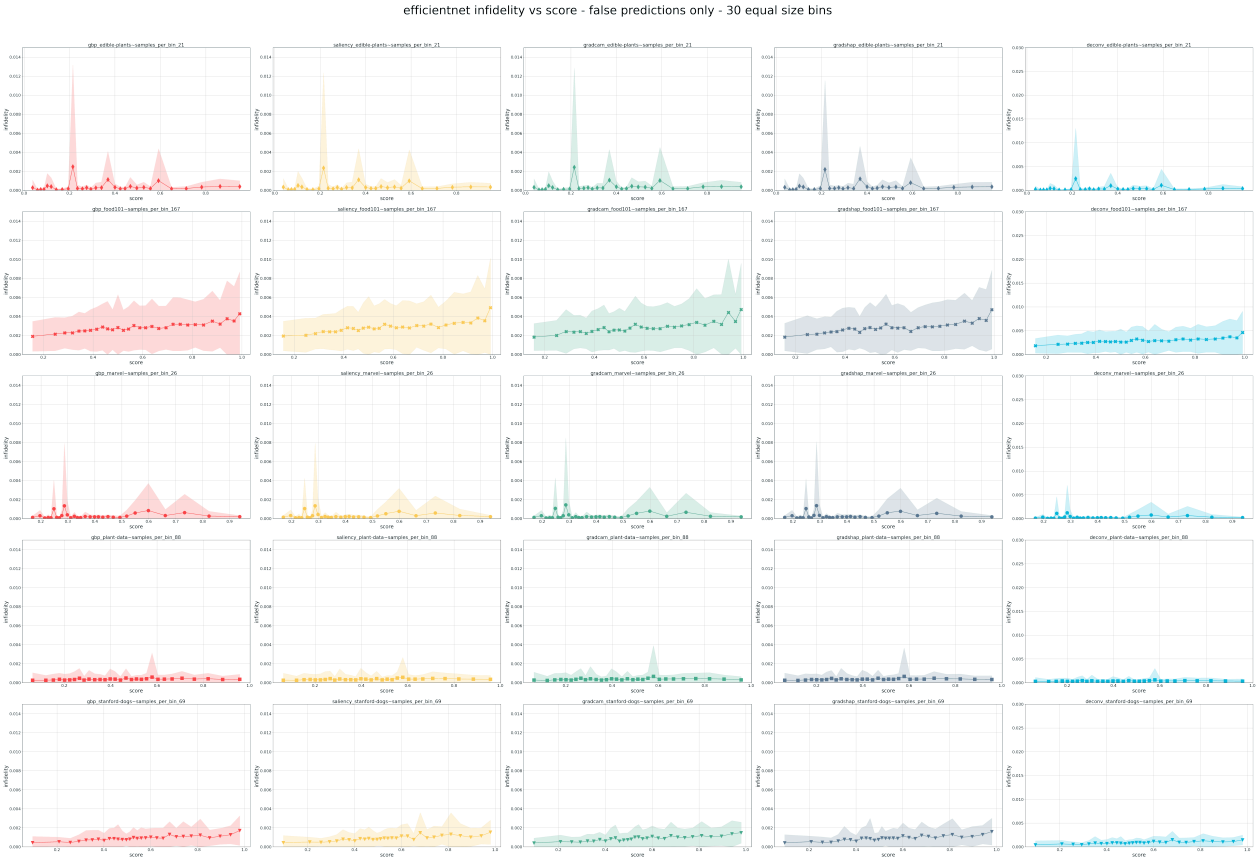


Figure B.6: Infidelity scores (with standard deviation) on *EfficientNet B0* architecture. All scores are the mean value for the particular model and are related to that models' predicted scores (x-axis). Each data point is a mean value of the same amount of samples per dataset. Amounts differ between datasets to always split the results into 30 equally-sized bins.

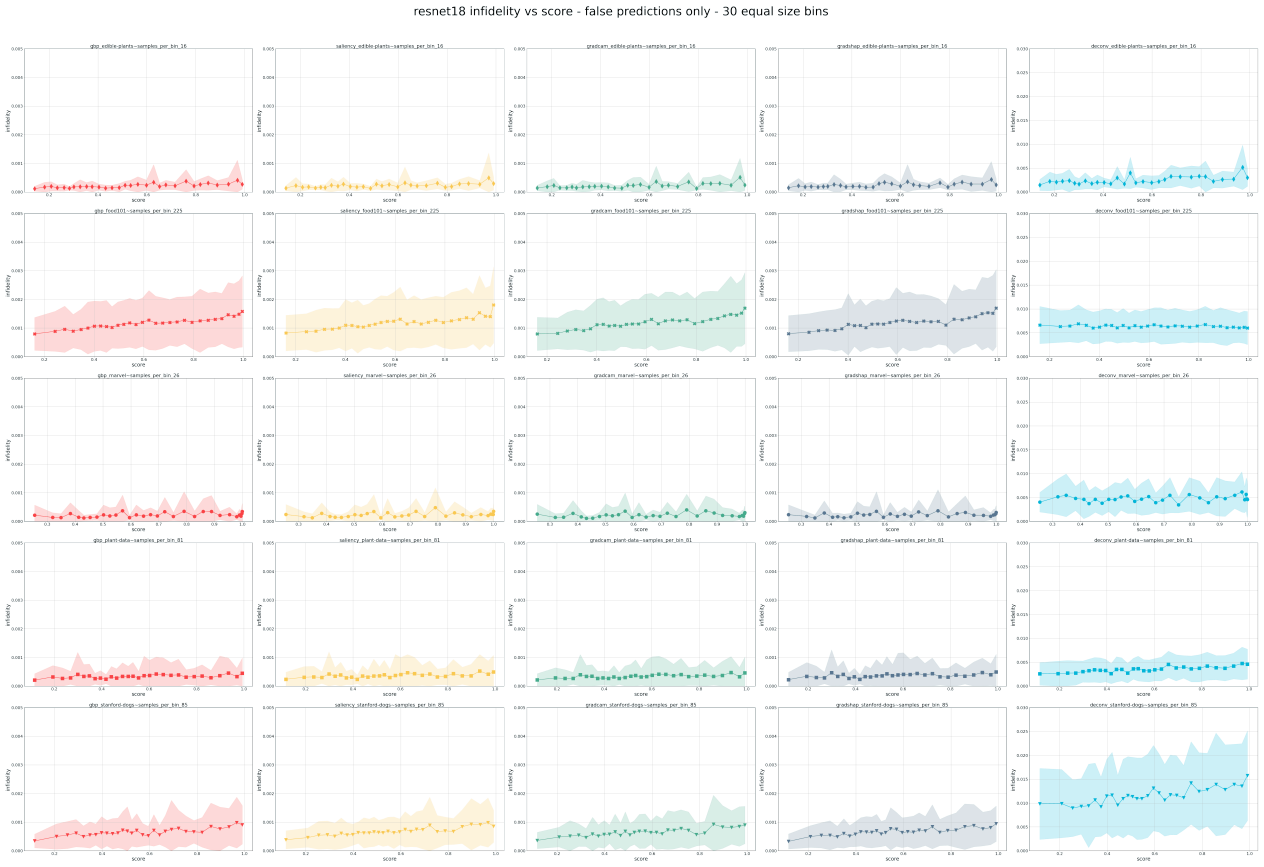


Figure B.7: Infidelity scores (with standard deviation) on *ResNet18* architecture. All scores are the mean value for the particular model and related to that models' predicted scores (x-axis). Each data point is a meas value of the same amount of samples per dataset. Amounts differ between datasets to always split the results into 30 equally-sized bins.

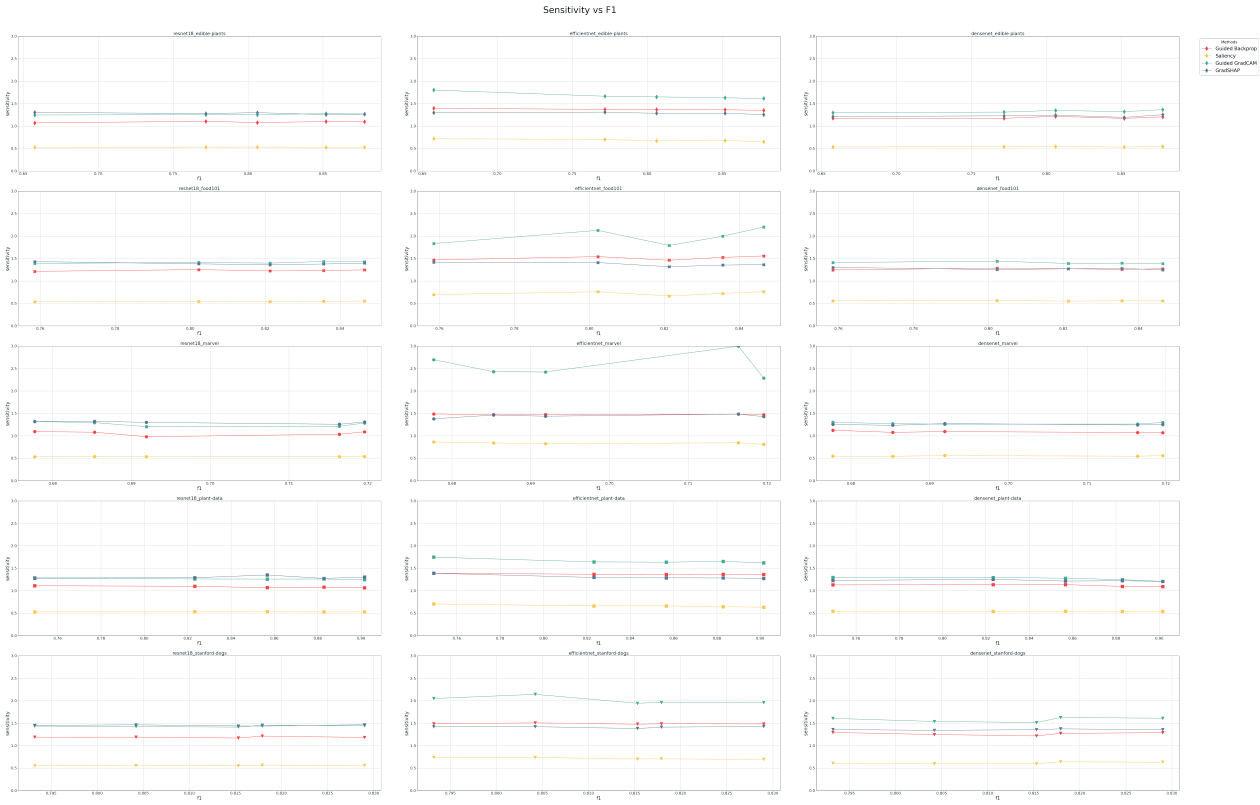


Figure B.8: Combined Sensitivity scores against F1 scores. Each row represents a dataset, each column represents an architecture (see individual subplot titles)

Figure B.9: Sensitivity scores (with standard deviation) on *DenseNet121* architecture. All scores are the mean value for the particular model and are related to that models' predicted scores (x-axis). Each data point is a mean value of the same amount of samples per dataset. Amounts differ between datasets to always split the results into 30 equally-sized bins.