JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Viktor Maximilian Loreth
k12006268

Submitted at
Institute of Computational Perception

Supervisor
Katharina Hoedt, PHD

November 25, 2023

EVALUATION OF INTERPRETABILITY METHODS IN IMAGE RECOGNITION

Bachelor Thesis

to obtain the academic degree of

Bachelor of Science

in the Bachelor's Program

Artificial Intelligence

JOHANNES KEPLER UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

JOHANNES KEPLER
UNIVERSITY LINZ

Sworn Declaration

I hereby declare under oath that the submitted Bachelor Thesis has been written solely by me without any third-party assistance, information other than provided sources or aids have not been used and those used have been fully documented. Sources for literal, paraphrased and cited quotes have been accurately credited.

The submitted document here present is identical to the electronically submitted text document.

Linz, November 25, 2023

JOHANNES KEPLER
UNIVERSITY LINZ

Abstract

This thesis focuses on the evaluation of interpretability methods. The primary challenge of this research area is the absence of a benchmark for evaluating interpretability methods. Although intrinsic interpretative models exist, the rise of black-box networks and their development present substantial challenges to interpretability. Despite various interpretability methods existing, the issue of evaluation remains unresolved. Existing evaluation methods, including "RemOve And Retrain" (ROAR), "Keep And Retrain" (KAR), "Benchmarking Attribution Methods" (BAM) and "Real-Time Image Saliency for Black Box Classifiers," offer numerical evaluations. Nevertheless, they remain ambiguous. In this thesis, we provide an overview of the current state of interpretability methods and their evaluation. Furthermore, we look at reconstructing the ROAR method using two attribution approaches.

JOHANNES KEPLER
UNIVERSITY LINZ

Contents

List of Figures

1 Introduction

Artificial Intelligence (AI) has undergone rapid development in the last few years. In today's modern era of mobile phones and computers, algorithms are used on a daily basis to have quick access to information and improve the efficiency of daily life. To ensure optimal performance of these algorithms, it is necessary to interpret their inner workings and functionality [3]. Several inherently interpretable algorithms exist (e.g.: Decision Trees, Linear Regression, Support Vector Machines, etc.), but deep neural networks (DNNs) achieve better performance in various tasks.

This transition in DNNs is attributed to the increase in computational power and the exponential increase in accessible data. Despite their remarkable accuracy, a major challenge is that deep neural networks remain opaque black boxes which we struggle to understand [34]. Nevertheless, the immense improvement in performance and their ability to handle massive datasets have led to widespread adoption in various sectors [44]. It is predicted that algorithms based on neural networks will become increasingly popular in the next few years [44].

However, one of the main difficulties with neural networks is the lack of reliable interpretation techniques. Understanding the models is required for several reasons. Regulatory laws require the data to be understandable. Additionally, the lack of interpretability in machine learning models leads to the presence of biases, such as gender discrimination and racial disparities [3]. Without a thorough understanding of how a model works, it becomes extremely difficult to confirm its functionality. This leads users to distrust and avoid machine learning systems [3]. In addition to these user-centric benefits, it also supports the model development process and allows experts to extract valuable insights [34].

To achieve this interpretability, many methods have been developed. However, a universally reliable method for neural networks is still missing [25]. Efforts to address this issue have led to several gradient methods that aim to assign significance values to pixels and represent their importance for neural network decisions [34]. Another alternative option to mitigate the black-box nature of algorithms involves employing model-agnostic methods. These methods offer a computational linkage between features and labels, irrespective of which model is used. Although highly effective for smaller datasets, they begin to struggle as the data size and complexity increase. Because of this, they do not offer a reliable method to quickly make neural networks interpretable [25].

In light of these prevalent problems, the object of this thesis is to recapitulate interpretability methods for neural networks in computer vision. Emphasis is placed on the evaluation of post-hoc interpretability techniques, focusing on the strengths and weaknesses of distinct techniques. Concluding the theoretical segment, a practical demonstration showcasing the application of RemOve And Retrain (ROAR) [19] is shown.

1.1 Structure of the Thesis

1. The first chapter 2 "Objectives of Interpretable Machine Learning" provides a definition of interpretable machine learning (IML). It also explains the various reasons why we strive for interpretability in machine learning models.
2. In the second chapter 3 "Machine Learning and their Interpretability", an overview of contemporary machine learning algorithms is made, categorizing them into two main groups: algorithms with inherent interpretability and those without. The goal is to make clear how supervised methods can be applied to image recognition.
3. In chapter 4 "Interpretability of Supervised Machine Learning Algorithms in Image Recognition" we dig into interpretability, emphasizing global and local model-agnostic techniques. These methods offer insights into overall model behaviour, regardless of algorithm specifics.
4. In subsequent sections 4.3 "Neural Network specific Interpretability Methods", model-specific post-hoc methods for Neural Networks are introduced. Feature visualization and gradient-based methods are explained.
5. In chapter 5 "Evaluation of post-hoc Interpretability methods", we focus on evaluating post-hoc interpretability methods. Various approaches to assess the effectiveness and dependability of these methods in offering meaningful insights into intricate models are introduced and discussed. Additionally, the advantages and disadvantages of these approaches are carefully examined to provide a comprehensive understanding of their applicability.
6. In chapter 7 "Which evaluation method and attribution method to use?", a conclusion is presented and advice on which evaluation method to use is given.
7. In the last chapter 6 "Reconstructing ROAR", the practical application of the ROAR methodology using the Food-101 dataset [7] and MNIST dataset [11] is presented to exemplify the discussed concepts. This real-world instance illustrates the current state of art of evaluation techniques in image recognition.

2 Objectives of Interpretable Machine Learning

In the rapidly evolving landscape of machine learning, interpretability has emerged as an important concept. Before going in-depth into various algorithms and methods, the fundamental question of: "What is interpretable Machine Learning (IML) and why do we need it?" is answered. A broad definition of IML given by [3]: "Interpretable machine learning is the use of machine learning techniques to generate human-understandable insights into data, the learned model, or the model output." Interpretability is important for various reasons. Allen[3] summarizes several objectives for interpretable machine learning:

Model Validation: Interpretable models are essential for validating (by a human) whether a learned model behaves as expected and consistently aligns with prior expectations and knowledge about the system.
Model Debugging: When unexpected behaviour occurs, finding the reasons for fault is impossible without understanding the system. To debug systems, an inner understanding of the machine learning system is necessary.
Transparency, Accountability & Trust: IML transforms black-box machine learning systems into understandable systems. Without trust and accountability, using them in high-stakes societal applications is not recommendable.
Ethics: Machine learning algorithms can be trained on biased data leading to unfair predictions that are discriminatory. To improve the fairness of machine learning algorithms interpretable methods need to be deployed to assess the fairness.
Data Exploration and Discovery: Insights into major patterns, trends, groups, or artefacts of the data are achieved by applying human-interpretable techniques. These data exploration insights influence the data pre-processing and model decisions.

Before going into detail about the different methods of interpretability an overview of some currently used supervised machine learning methods is given in the next chapter. Unsupervised machine learning is arguably intrinsically understandable, as the goal is to find a structure in the data [3]. With those examples, the necessity of interpretability for neural networks should be made evident.

3 Interpretability in Supervised Machine learning Methods

The interpretability of algorithms depends on two factors [25]. When determining the transparency of an algorithm, we evaluate the human interpretability of how the model learns from the underlying structure of the data. Is it possible for a human to understand the implications of the mathematical operations? The second factor is the interpretability of the learned parameters. Understanding the factors of a linear regression model is easy. Grasping the millions of weights in a neural network is impossible.
In this section, we present a selection of frequently employed supervised classification methods. They are briefly introduced and analyzed for their interpretability. As will become clear in the following sections, before the rise of Deep Neural Networks, interpretability was predominantly achieved through methods with inherent interpretability.

3.1 Linear Models

When predicting outcomes, one of the simplest methods is to use a linear regression model [31]. This model predicts by adding up n features (x_n) multiplied by a respective individual weight (α_n). The predictive output $\hat{y}_i$ is calculated:

$$\hat{y}_i = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n + \epsilon$$

The alphas α_i indicate the correlation of a feature with the target. The initial coefficient α_0 is known as the intercept, signifying the baseline. The noise ϵ describes the inevitable errors from inherent non-linearity in real-world dynamics or measurement inaccuracies.
To train the model, the MSE-Loss (Mean Squared Error) or the ABS-Loss (Absolute Loss) is applied between the true label y_i and the predicted label $\hat{y}_i$. The goal is to minimize this loss function.

$$\text{MSE-Loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\text{ABS-Loss} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

The interpretation of the model is simple. The factors are described through the coefficient matrix. Each feature is distinctive to the model and the weighting is visible in the vector α (assuming normalization).

$$\alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \dots \\ \alpha_n \end{bmatrix}$$

While linear models offer comprehensibility and a simple method of prediction, their application is restricted to data sets where there is a linear relationship between one or more variables and the target. In the domain of image recognition, it is not applicable because the features and the target are not linearly correlated.

3.2 Distance-based Methods

K-Nearest Neighbours (KNN) [27] serves as a classification method by considering the nearest neighbours.

Assume you have a dataset $D = (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where x_i represents the feature vectors, and y_i represents the corresponding class labels. Each x_i is a point in a multidimensional feature space, and y_i belongs to one of the classes (e.g., $y_i \in \{0, 1\}$ for binary classification). Given a new data point x_{new} that you want to classify, KNN works as follows:

1. Calculate the distance between x_{new} and all other data points in the training dataset.
2. Select the K data points (nearest neighbours) with the smallest distances to x_{new}.
3. Count the number of data points in each class among the selected KK neighbours.
4. Assign x_{new} to the class that has the majority of neighbours.

During training, KNN does not store any parameters but rather constructs an underlying data structure which is interpretable, for example, a ball tree or k-d tree. This makes KNN a widely used technique when interpretability is needed. However, when it comes to dealing with large datasets and many features, KNN faces considerable challenges. Therefore, it is not recommended to use KNN for image recognition purposes.

3.3 Support Vector Machines

?? Support Vector Machines (SVMs) [6] are used to find a hyperplane that best separates different classes y_i of data points. The primary objective is to maximise the distance between the hyperplane, defined by the weight vector w, and the closest data points x_i from each class. This is done to ensure that the datasets are separated as much as possible. Because there may be classification errors and noise in the dataset, it is possible to set an initial regularization parameter C to handle outliers.

The optimization problem SVM (for binary classification) in mathematical notion:

$$\text{Minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{Subject to } y_i(cw \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \text{ for } i = 1, \ldots, n$$

$\sum_{i=1}^{n}\xi_i$ represents the sum of slack variables (ξ_i), which measure how much a data point violates the margin constraint. While maximising the margin, Support Vector Machines (SVM) seek to reduce this sum, minimizing the violations. b is the bias term, which shifts the hyperplane away from the origin. For multiclass classification, a "One vs One" or "One vs Rest" approach can be used, in which multiple SVMs are trained.

SVMs can handle non-linear data through the kernel trick, in which a kernel is applied to the data, transforming it to a different space before computing the hyperplane. However, as the dimensionality increases, SVMs become non-interpretable, making it challenging to visualize and understand the weight matrix. Moreover, SVMs struggle with larger datasets and high-dimensional feature spaces, making them less effective for image classification.

3.4 Decision Trees

Decision trees are a commonly used machine learning algorithm that proves effective in both classification and regression tasks. The reason for their popularity can be attributed to their user-friendly structure, which can handle both categorical and continuous data. A decision tree is made up of three main types of nodes: root, internal, and leaf nodes, which combine to form a representation often depicted as a tree. In figure 3.1, a decision tree is plotted.

Example Dataset

#	Feature 1	Feature 2	Class
0	2.5	1.0	0
1	1.5	5.0	0
2	4.0	2.5	1
3	3.5	4.0	1
4	5.0	4.5	0
5	4.5	1.5	1

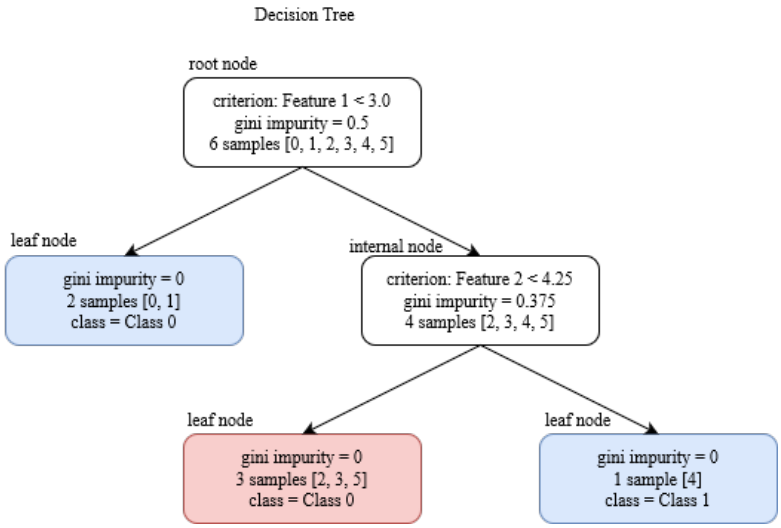

Figure 3.1: Decision Tree Example

An example dataset with 2 features and 2 classes is depicted to explain the behavior of a decision tree. The classes are separated in each root and internal node by a decision criterion: F_i <= criterion. The Gini impurity is commonly used for calculating the splitting criterion, which aims to best separate the data in the node. This process is repeated and new nodes are defined until the dataset is perfectly separated. However, this makes decision trees prone to overfitting as the model becomes overly complex and captures noise or random fluctuations in the training data, rather than the underlying patterns or relationships. Consequently, the resulting tree fits the training data exceptionally but does not generalize well to new or unseen data.

The depth of the tree can be restricted to avoid overfitting. Then, the majority is utilised to assign a label to the leaf. Nevertheless, as the dimensionality grows, determining the depth becomes more and more challenging.

Random forests[18], comprise many decision trees and are known to prevent overfitting. However, they tend to be more challenging to interpret because of the use of multiple algorithms. Techniques like SHAP values [23] or partial dependence plots [15] can be employed to make them more understandable. Gradient boosting like XGBoost [8], LightGBM [20] and CatBoost [30] share similarities with random forests and decision trees, but they assign differential learned weights to each decision. They suffer from the same interpretability issues as random forests. While decision tree-based methods can be applied for image classification, their accuracy tends to be worse than neural networks.

3.5 Neural Networks

The rise of neural networks and their powerful predictive capabilities has made them a popular choice for classification tasks. However, as these networks become increasingly complex, the traditional approach of understanding them through weight examination becomes challenging. With the emergence of Convolutional Neural Networks (CNN) in 2012 [22], neural networks have become state-of-the-art for image prediction.

In a neural network, each neuron calculates a weighted sum of its inputs and applies an activation function to produce an output. This output serves as the input for the subsequent layer. While various neuron types exist, including convolutional and recurrent neurons, foundational neurons are present in nearly all neural network architectures. A visual representation of a simple neural network can be seen in Figure 3.2.

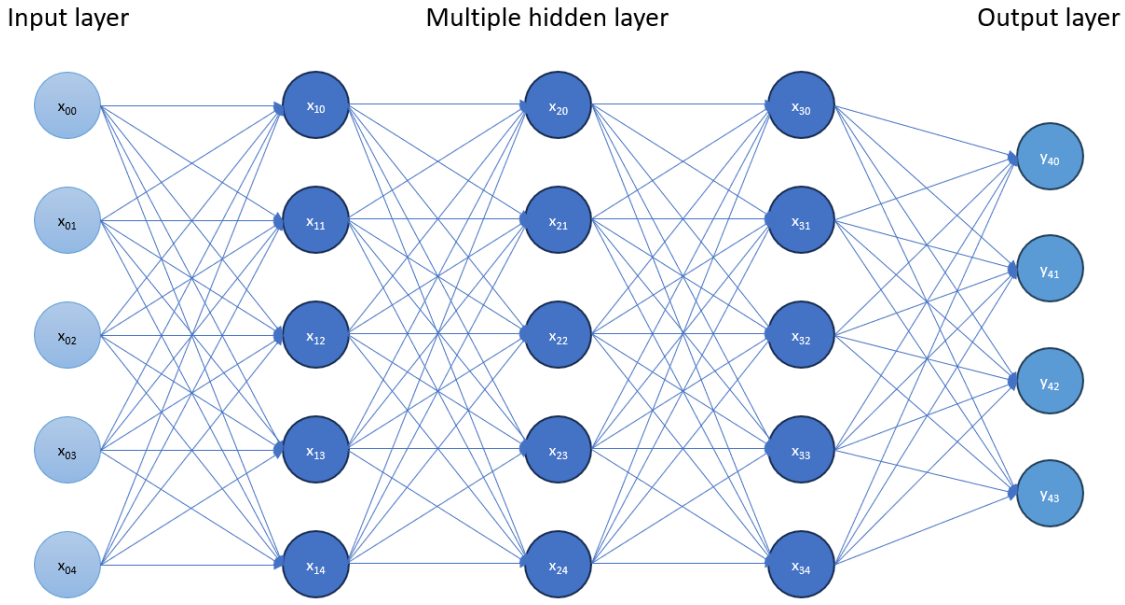

Figure 3.2: Neural Network Example

In this neural network, each edge, except those in the input layer, represents a weight matrix denoted as w. In this example, 'j' signifies the hidden layer number, and 'i' represents the weight number within that layer. The output of a neuron is determined by summing the products of the inputs x_{ji}, their corresponding weights, and a bias term b_{ji}, expressed as $y_{(j+1)i} = \sum_{i=1}^{n} w_{ji} * x_{ji} + b_{ji}$. Depending on the architecture, an activation function, such as Sigmoid or ReLU, is applied to y_{ji}, transforming it into the new input for the next layer $x_{ji} = f(y_{ji})$ or the output $f(y_{ji})$.

Labeled data is used to train a neural network. The prediction error is minimized by calculating the prediction error and the gradients. This is typically done by backpropagation over several epochs until the network can accurately predict new data.

Modern neural networks, such as ResNet50 [17] comprise over 50 layers and utilize more than a thousand kernels. Although they produce the greatest results, they lack an intuitive explanation of the learned parameters, unlike other types of supervised machine learning algorithms. Nonetheless, interpreting them to some degree is a necessity. Therefore, methods were developed that attempt to make any supervised machine learning algorithm interpretable.

4 Interpretability of Supervised Machine Learning Algorithms in Image Recognition

Generating interpretability for image recognition poses a challenge due to its high dimensionality. Additionally many interpretable methods are too computationally expensive [26]. Additionally, intrinsic interpretable methods yield inferior outcomes compared to black-box models [10].

Before going into the specifics of interpretability methods for image recognition, we present a framework [3], which is displayed in figure 4.1, for categorising supervised machine learning and interpretability methods.

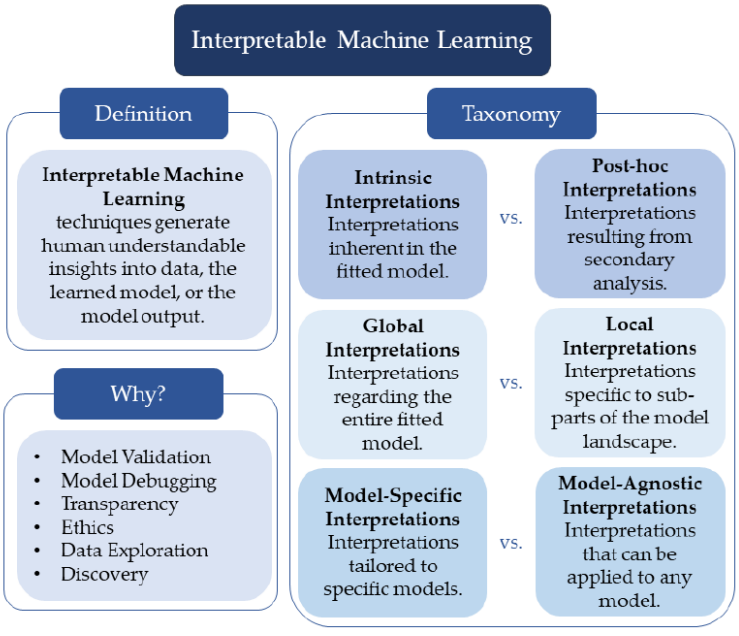

Figure 4.1: Interpretability categorization (Figure from [3])

Intrinsic vs Post-hoc: Methods can be broadly classified into intrinsic and post-hoc techniques. Intrinsic supervised methods are inherently understandable by design, while post-hoc methods involve analyzing the model's behaviour after its creation. Typical intrinsic understandable methods are Decision Trees 3.4 and KNNs 3.2. As methods for classifying data become more difficult, the methods themselves become less interpretable and lose their intrinsic interpretability. Therefore, post-hoc methods must be applied to them to make them interpretable again.
Global Interpretations vs Local Interpretations: Global interpretability methods encompass the entirety of a fitted model. In contrast, local interpretations zoom in on specific portions of the model landscape, such as class boundaries or single interpretations. Local interpretations exist because for some models it becomes too complex to generate a global interpretation. In order to still have some interpretability in the model, local interpretations are used.
Model-Specific Interpretations vs Model-Agnostic Interpretations: Model-specific interpretations are tailored to a particular class of algorithms, like gradient methods for neural networks. Model-specific interpretations make use of the inherent information as weights or gradients in a model. In contrast, model-agnostic interpretations, such as LIME or SHAP, can be applied to any classification algorithm because they do not require model-specific knowledge.

To address the issue of interpretability in a comprehensive manner, a universally applicable interpretability method is necessary. This method must have low computational costs while providing a reliable truth for evaluation. However, such a method does not presently exist.

The goal of this section is to present several commonly used interpretability methods and evaluate their suitability for image classifications. As Neural Networks are the best-performing methods for image recognition, the focus lies on the suitability of the interpretability methods for Neural networks. After going over model-agnostic interpretability methods in the next sections, we go in-depth about model-specific interpretations for neural networks in section 4.2.

4.1 Model-Agnostic Methods and their applicability to Neural Networks

Model-agnostic methods can be applied to any method. However, it does not make sense to use the same method for every algorithm. In this section, examples of global and local agnostic methods are summarised and their applicability to neural networks in image classification is discussed.

4.1.1 Global Model-Agnostic Methods

Global Methods aim to provide a complete overview of the functioning of a model. However, it is difficult to represent the non-linearity in image classification for multiple classes. While some methods work reasonably well for the computation required, there is no global agnostic method that fully covers the need for interpretability.

1. **Partial Dependency Plots:** Partial Dependency Plots (PDPs)[15] display the relationship between individual or multiple features and an outcome. However, when dealing with a bigger feature size, there are too many plots to make sense of the data. Therefore they are not feasible for Neural Networks and image recognition tasks.

2. **Accumulated Local Effects:** Accumulated Local Effects (ALE)[4] are an advancement of PDP. While it overcomes certain difficulties from PDPs, they struggle with the same problem when it comes to a bigger feature size.

3. **Feature Interaction:** Feature Interaction analyzes the interaction between features inside the model. The underlying Friedman's H-statistic[12] is a method to evaluate the correlation and variance. Because of the complexity of image tasks and the high computational costs, this method is not applicable to Neural Networks in a meaningful matter.

4. **Functional Decomposition:** Functional Decomposition is used in Neural Networks. In section 4.2.2 Network Dissection a method to disassemble networks is presented.

5. **Permutation Feature Importance:** Permutation Feature Importance is commonly used in visual machine learning tasks. It can be used for the evaluation of a feature importance map constructed by another interpretability method. An example is given in section 5.2.

6. **Prototype and Criticism:** The creation of clusters using the prototype and criticism method involves categorizing well-presented data instances as 'prototypes' and sparsely presented ones as 'criticism'. This approach aims to improve the interpretability of these clusters by verifying their classification accuracy. However, a significant challenge with this method arises when applying it to images, as it can be difficult to generate reliable clusters that effectively group the data.

4.1.2 Local Model-Agnostic Methods

The aim of local model-agnostic methods is to reveal knowledge of how individual predictions of a model behave. In the field of image classification, where non-linearity and multiple classes contribute to added complexity, achieving accurate interpretability for a single sample is already challenging. Although some techniques effectively manage the trade-off between computational efficiency and interpretability in certain scenarios, there is currently no universally accepted model-agnostic method for achieving this. Nevertheless, they can be applied to Neural Networks and potentially give meaningful interpretations.

1. **LIME: Local Interpretable Model-agnostic Explanations:** LIME[33] generates explanations in a local scope by training interpretable models on the predictions of a model. LIME only covers a single local context of the model. LIME can be applied to various types of data, including image data.

2. **Scoped Rules (Anchors):** Anchors [32] are distinctive patterns or conditions which guarantee a prediction. Finding anchors becomes increasingly expensive with more features present and is not normally used in neural networks.

3. **Individual conditional expectation curves:** ICE [13] display one line per data sample and display how the sample changes when a feature changes. It is an individual version of PDP. It is not used in image classification or to evaluate neural networks.

4. **Counterfactual explanations:** Counterfactual explanations [41] of a prediction explain the smallest change of feature values which is necessary to change the prediction of an output. The problem in using this method is that for each instance multiple counterfactual explanations exist. The use of counterfactual explanations in image recognition as a standalone method is not common.

5. **SHAP (SHapley Additive exPlanations):** SHAP values [23] calculate how much each feature contributes to the difference between a models prediction for a specific instance and the average prediction across all instances. Positive SHAP values indicate features that increase the predicted likelihood for a particular class, while negative values indicate features that decrease it. It is used commonly in neural networks.

While the summarized techniques can be applied to neural networks, they do not provide a universal solution for interpretability. For this reason, Model-specific methods for neural Networks were developed which make use of the inherent parameters of neural networks.

4.2 Model-Specific methods for Neural Networks

In the domain of Natural Language Processing (NLP) and Computer Vision, Deep Learning has proven very successful [10]. By passing the features through a sequence of layers, characterized by matrix multiplications with kernel weights and nonlinear transformation functions, a prediction is computed. Depending on the specific task, additional elements like Long Short-Time Memory(LSTM) layers and Convolutional layers are utilized. Given the immense amount of mathematical operations and the non-linearity underlying a single prediction, humans are not fit to apprehend the mapping. To interpret predictions, we would have to decipher the intricate learned knowledge of numerous different kernels and weights. Recognizing that humans cannot grasp millions of weights, the demand for interpretability methods is high. To assess the behaviour and predictions of Deep Neural networks, specific interpretability methods were developed. These methods calculate the likelihood of a feature being responsible for the result.

While model-agnostic methods offer an approach to understanding Neural Networks, the high-dimensionality of the features and the high computational effort makes it difficult to use model-agnostic methods [26]. For instance, in an image with the dimensions of 3x224x224, as commonly encountered in Food-101, the data features exceed 150.000. In NLP tasks, where vocabularies often encompass around 20,000 words, the computational complexity renders most model-agnostic techniques difficult to employ.
In the pursuit of comprehending the complexity of Deep Neural Networks, it makes sense to utilize the weights and gradients in the model. The information saved in the hidden layers as learned weights can be used to evaluate the network. Moreover, the gradients can be taken into consideration as well. In the following subsections, several concepts for understanding Deep Neural Networks are introduced.

4.2.1 Feature Visualization

Feature Visualization aims to make parameters in single layers in neural networks understandable. The goal is to give an understanding of a deep neural network by dissecting each layer and understanding the underlying parameters through visualization.

The higher-level features in these networks relate to clear concepts, shown in Figure 4.2. As the input (image-pixels) passes through the layers, the resembled structure changes at each layer. In each convolutional layer, the network gains new and more complex features. The joining of fully connected layers then converts image-based data into predictions.

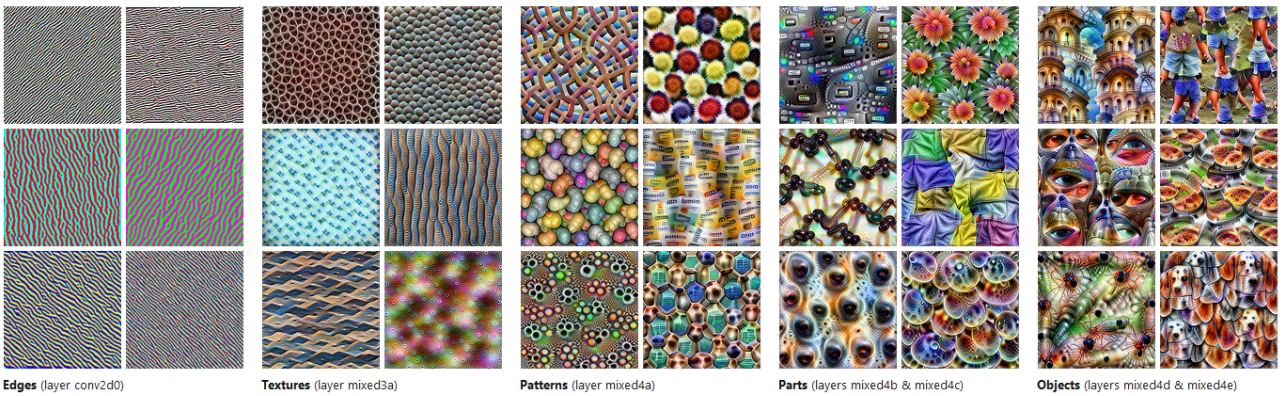

Figure 4.2: Feature Visualization (Figure from [29])

The figure visualizes this process. The first convolutional layers find simple features like edges and basic textures. Later, they recognize more detailed patterns. The deepest layers learn about parts and objects. This object information passes to the other hidden layers, which then finally make a prediction.

Feature visualization is based on activating one kernel in the network. This involves maximizing the activation of a specific neuron (Visible in figure 4.3). There are two methods for achieving this. First, we can make use of the training image that triggers the highest activation. Yet, this approach faces a significant problem. When an image contains multiple objects, it is hard to pinpoint which object causes the activation. Because of this, an alternative route is adopted: generating new images from random noise. This is accomplished through methods like Generative Adversarial Networks (GANs) [14] or other diffusion-based techniques[46].

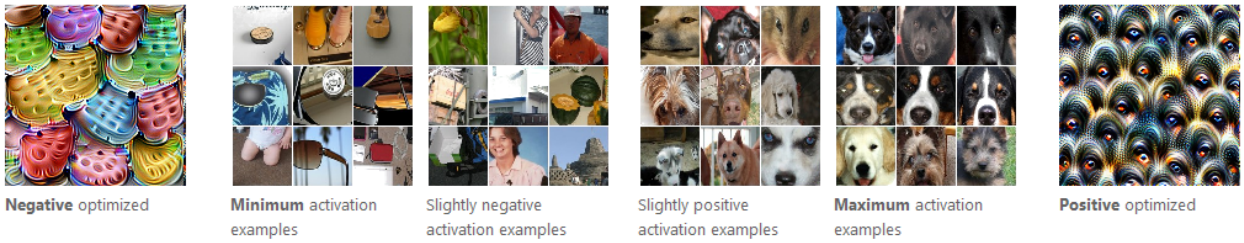

Figure 4.3: Activation Maximization (Figure from [29])

Feature visualization provides a first insight into the behaviour of a model, improving the understanding of its inner layers. It also has the potential to enrich domain understanding by aligning learned features with domain-specific knowledge. Furthermore, it can assist in debugging and refining models, contributing to their overall performance improvement. However, interpretation of the visualized features is difficult and decision making is challenging to understand.

4.2.2 Network Dissection

Network dissection [5], builds on the principles of feature visualisation. First, they identify a broad set of human-labelled visual visual concepts. Then, they collect the responses of hidden variables to these known concepts using feature visualization. Finally, they quantify the correspondence between hidden variables and concepts.

Network Dissection [5] operates based on feature visualization. Initially, a broad set of visual concepts labeled by humans is built. Subsequently, the response to these recognized concepts are generated through the utilization of feature visualization techniques. Afterwards, the alignment between hidden variables and concepts is measured, providing a quantitative assessment of their correspondence.

The method establishes a link between the individual concept and the prediction of specific features. Although it is effective at capturing low-level features, capturing high-level features remains a challenge. In addition, the analysis of a single sample is very time consuming. However, this interpretability method has significant potential for improving the transparency of neural networks.

4.3 Attribution Maps

Attribution maps are visualizations that highlight the regions of an input image that have the most significant impact on a model's output. By revealing the areas that strongly influence a prediction, saliency maps bridge the gap between the model's "black-box" nature and human understanding. An example attribution map is visible in figure 4.4.

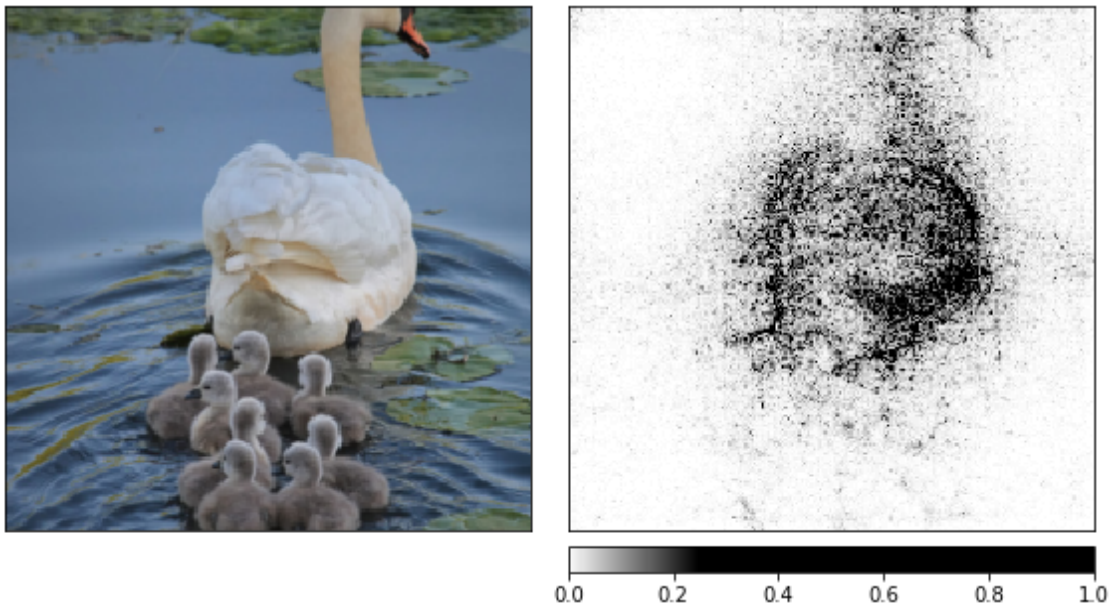

Figure 4.4: Attribution Map: Classification of a swan (Figure from [1])

Attribution maps are typically calculated using SHAP[23] or gradient methods. Attribution maps provide a direct and intuitive way to understand which parts of input data influence a model's decision. The main difficulty is in generating reliable attribution maps [21]. The remaining subsections introduce some commonly used methods. The key objective for all the following methods is the same: Generating a reliable and easily understood attribution map.

4.3.1 Vanilla Gradient and Deconv-Net

Vanilla Gradient [37] focuses on the computation of gradients within neural networks. It involves generating a forward pass of an image and then computing gradients for the class values. These gradients are then visualised. However, there are two challenges with this method. First, when using Rectified Linear Units (ReLU), negative gradients are set to zero. This results in a loss of information in the gradients. Second, there are no gradients in the pooling layers.
The Devonc-Net from Zeiler [24] addresses these problems: A Deconv-Net-layer [43] is attached to each convolutional layer providing a path back to the image pixels. The Devonc-Net learns

weights to reverse the process from input to the last convolutional layer. Unpooling in Deconv-Net addresses the non-invertibility of pooling operations by preserving the original maxima locations through switch variables, which are used to reconstruct the activations. To maintain positive signals during reconstruction, a reverse ReLU non-linearity is applied in each layer, similar to the forward pass. Additionally, Deconv-Net utilizes transposed filters, effectively mirroring each filter both vertically and horizontally to achieve filtering in the reverse direction.

This makes it possible to highlight the responsible features for a classification output.

4.3.2 Gradient-weighted Class Activation Map: Grad-CAM

Grad-CAM [36] ignores low-relevancy classes, which means that percentages for predictions below a certain threshold are not taken into account. The method then backpropagates gradients specifically for the classes of interest. These gradients are rescaled to fall within the [0,1] range and visualized. Grad-CAM takes into account all layers except for the convolutional layers. Evaluating the importance of each kernel prior to the convolutional layers provides insight into the importance of each input feature. This leads to a coarse accuracy, as pooling layers and convolutional layers are not handled and the resolution is up-scaled to the required format.

Guided Grad-CAM [36] combines Grad-Cam with guided backpropagation [39] to have a better localization. The up-sampled attribution map from Grad-Cam is multiplied pixel-wise with another attribution method. This acts as a lens for other attribution methods to focus on specific parts of the attribution map.

4.3.3 Integrated Gradient

Integrated Gradient [40] calculates the contribution of each feature by integrating the gradients of the model output with respect to the input along a path from a baseline input to the current input. The baseline input (a black image for images) serves as a reference point, and the integral approximates the accumulated effect of each feature on the model output. By considering a path rather than a single point, Integrated Gradient is able to visualise the features of an image more exactly.

4.3.4 Ensemble Methods: Smooth Gradient, Smooth Grad² and VarGrad

All the following methods can be applied to gradient-based techniques to adjust their behaviour and enhance their robustness in various applications.

SmoothGrad [38] is a smoothing approach to mitigate noise in gradient calculations. To achieve this, it generates a set of J noisy estimates by independently adding Gaussian noise θ to the input x.

These noisy estimates are then averaged to obtain a more reliable and accurate gradient estimate, as represented by the formula. g is describing the calculation of an attribution map, A, is the attribution map:

$$A = \frac{1}{J}\sum_{i=1}^{J}(g(x+\theta))$$

SmoothGrad² [19] builds upon the concept of SmoothGrad by squaring all estimates before averaging them. This modification is aimed at emphasizing the contributions of gradients while further reducing the impact of noise. The formula is as follows:

$$A = \frac{1}{J}\sum_{i=1}^{J}(g(x+\theta)^2)$$

VarGrad [2] offers an alternative approach to aggregating gradient estimates. Instead of summing them up or averaging them, VarGrad focuses on estimating the variance of these estimates. This aggregation provides insights into the variability of gradient information across different perturbations, offering a unique perspective on the model's behaviour. The formula for VarGrad is expressed as:

$$A = Var(g(x+\theta))$$

4.3.5 Summarizing Attribution Maps

There exist many more attribution maps but there exists one major problem: There is no consensus on which method should be used. While attribution maps have advantages [25], such that they are fast to compute and easy to interpret, they are also unreliable [26] and can be insensitive to models and data. Furthermore, it is difficult to know whether an explanation is correct. In the next chapter, the evaluation of interpretability methods is discussed.

5 Evaluation of post-hoc Interpretability Methods

This chapter discusses the evaluation of attribution maps and the underlying gradient methods. Finding appropriate evaluation methods for attribution maps is difficult. A significant challenge is the lack of ground truths in attribution maps. Nevertheless, several evaluation methods have been developed that attempt to assess the performance of saliency methods. While they do not provide an objective truth as to whether one saliency method outperforms another in all tests, they can provide some indication of the effectiveness of saliency methods. Gupta[16] suggests that these evaluation methods can be broadly categorized into two types:

Extrinsic Evaluation: The evaluation is compared against predefined ground truths explanations, established by a human annotator or by another network e.g.([45]). However, the sheer amount of images makes the evaluation challenging. It is difficult to establish a ground-truth for attribution. Also, Adebayo [2] underscores the issue of confirmation bias, where an explanation may be considered correct because it highlights the same regions as humans might choose, but it is not sure that the model employs the same learned concepts.

Intrinsic Evaluation: Intrinsic methods rely on computational analyses involving the network to evaluate saliency methods, without requiring human judgments. These methods often are based on creating a new composite input using the heat map and the original input. Examples for such methods are Roar [19] and the saliency metric [9].

Many evaluation methods exist, but there is only little correlation[28] in the results for different datasets. If a correlation exists, it is model and task-dependent, which furthermore weakens this assumption. In the upcoming sections, potential evaluation metrics and methodologies are introduced to provide a more meaningful assessment. These metrics and methods do not offer an objective evaluation, but they offer some guidance for evaluation.

5.1 Completeness and Soundness

The Soundness and Completeness metrics [16] are used to evaluate saliency methods.

Soundness: A certain part of the image is responsible for the output of the classifier. Some parts of the pixels are responsible for the classification of the label. In practice, this means that if you

apply any mask to the image that still makes the x% of the most important pixels visible, then the classification will be the same. This also means that if some of these important pixels are not visible, then the confidence of this classification will decrease.
Completeness: No matter what mask is chosen, as long as the x% most important pixels are visible, the classification will be correct. In practice, this means that if the important pixels are visible, the classification will remain correct, regardless of the appearance of the other pixels.

These two metrics are useful for assessing the correctness of masking. However, it is only one metric for other evaluation frameworks. Gupta suggests that saliency methods should test for both completeness and soundness.

5.2 Evaluating the Visualization of What a Deep Neural Network has learned

In the paper by Samek[35] a method is described, where the most important pixels detected by an attribution method are systematically replaced with randomly sampled values using a greedy algorithm. The objective is to measure the impact on the classification value f(x). The method can be applied by only removing the important pixels or by defining a local neighborhood around the pixel which is replaced as well. This neighborhood is defined because, from the context of the nearby pixels, the same information can be deduced.

While the method produces results which confirm that removing important pixels leads to a performance drop, the models are not retrained [19]. This leads to an evaluation of a different data distribution and violates the assumption of a similar data distribution for the train and test data.

5.3 A Benchmark for Interpretability Methods in Deep Neural Networks

This paper [19] presents two methods for numerically evaluating attribution maps. "RemOve And Retrain (ROAR)" and "Keep And Retrain (KAR)". This is done by removing or keeping supposedly informative features from the input and observing the response of the neural network. The section 6 gives an example of a practical application for MNIST[11] and Food-101[7].

ROAR can be applied to any method that produces attribution maps. It identifies 10, 30, 50, 70 or 90 % of the most important pixels in an attribution map and ranks them according to their numerical importance. In order to have a baseline against which to compare, a random baseline is generated which randomly assigns the importance of the pixels.

After the most important pixels have been identified, new datasets are generated for each threshold. In this step, x% of the most important pixels are replaced by the mean value of the data set. Once the new data sets have been generated, new randomly initialized models with the same architecture and learning parameters are trained on this modified data set. The same train/test split should be used as for the original model. This is done for several runs to ensure consistency of results. After retraining for each set of removed pixels and attribution maps, the results are compared. A model trained on a dataset with random % pixels removed should have a slower loss of accuracy than a model with a dataset using an attribution map to remove the pixels. The greater the decrease in accuracy compared to the random baseline, the better the attribution map.

KAR works in a similar way to ROAR, but instead of removing the most important pixels, it removes the least important pixels. As KAR performs worse than ROAR, it is not described further.

The conclusion of Hooker [19] is that some commonly used attribution estimators like Gradient[37], Integrated Gradient[40] and Guided Backpropagation[39] perform worse than random assignments in ROAR. On the other hand, the effectiveness of Smooth-Grad-Squared and VarGrad was confirmed. However, ROAR is sensitive to the dataset used. While it does provide a numerical evaluation, it is sensitive to the model and dataset used and therefore cannot universally evaluate an algorithm.

5.4 Benchmarking Attribution Methods (BAM)

As already mentioned before, determining the ground truth of an attribution map is impossible. However, it is possible to compute the relative importance of specific features when comparing it between two different models[42]. BAM[42] uses this to evaluate different attribution methods.
Visualize an image consisting of two distinct pixel sets:
Each image contains one object, represented by the p_o pixels, and the remaining pixels p_s make up the background scene. By training two distinct classifiers, one trained to detect objects and the other trained for scene identification, we expect that the relative feature importance for the p_o pixels should be higher for the object classifier than for the scene classifier. Conversely, the same holds for the p_s pixels for the scene classifier.
In BAM, such image-scene pairs are created systematically. To address the distribution shift problem, objects with a mean similar to the mean of the dataset are chosen so that they do not significantly alter the image distribution. Attribution maps are calculated for each image and both trained classifiers. Those attribution maps are then compared using different metrics. However, the results show different results for various metrics and no clear result can be given for any method.
The advantage of BAM over ROAR [19] and other methods is the lower computational cost. No retraining or perturbation is required. However, the method has the usual problems of evaluation algorithms: sensitivity to the dataset and model and not offering a universal truth.

5.5 Sanity Checks for Saliency Maps

Adebayo [2] proposes two "sanity" tests to evaluate whether attribution methods are meaningful. It reveals that visual inspection by humans does not determine if the explanation is sensitive to the underlying model and data. Two instances of the framework are tested:
The model **parameter randomization test** involves comparing the output of a saliency method on a trained model with the output of a randomly initialized untrained network of the same architecture. The output should differ, otherwise, the saliency map is considered ineffective. An example of this test is visible in figure 5.1. The **data randomization test** randomly shuffles the labels of the data. Afterwards, the model is trained on the altered dataset. If the saliency maps do not differ from a normally trained model, then the method does not depend on the relationship of the images and labels.

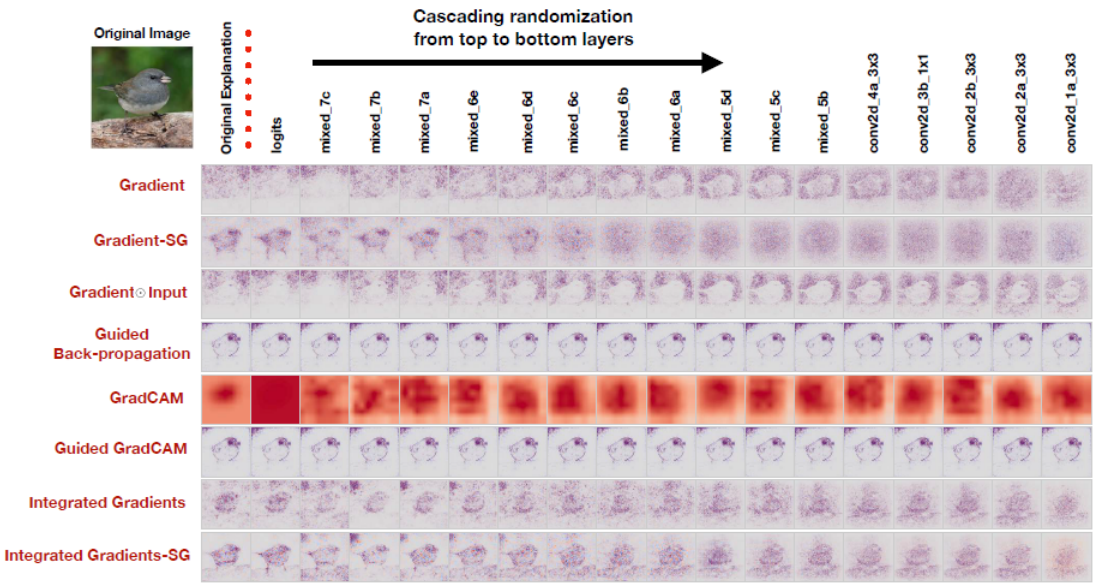

Figure 5.1: Cascading Randomization on Image Net. The figure shows the original explanations. Progression from left to right indicates complete randomization of network weights up to that block inclusive. The last block corresponds to a network with completely reinitialized weights. (Figure from [2])

If either of these two hypotheses fails during testing, the method under evaluation can be rejected, indicating that it fails a critical sanity check. On the tested methods, Guided BackProp & Guided GradCAM fail these sanity checks.

6 Reproduction of RemOve And Retrain (ROAR)

6.1 Scientific Motivation and Goal

After looking at different methods in theory, we look at one approach in detail. We decided to look at ROAR[19] because it is a relatively recently published study. We try to reproduce some of the experiments. Only a small part of the original experiment was reconstructed because the complexity and the required computing power were not available. In addition, we also want to look at the fluctuations between the different runs, as these were not included in the original paper.

Due to resource constraints, the research was limited to evaluating Food-101 [7] using two interpretability methods along a baseline model. Instead of five control runs, only four were performed. As an initial proof of concept, the evaluation was performed on the MNIST dataset[11].

6.2 Graying out Pixels

In ROAR, pixels are grayed out to hide potentially informative pixels. Graying out pixels does not necessarily mean that the information is completely hidden. To provide a theoretical explanation for the implications of removing input dimensions on the resulting accuracy, the following two factors[19] should be considered:

1. Input dimensions are removed and the accuracy of the model drops:
   The removed input was informative for the model and their absence reduces the ability of the model to identify the correct class.

2. We remove inputs and the accuracy of the model does not drop.
   (a) It is possible that the removed input dimensions were not important for the model's decision-making process. The attribution map failed to identify important features, such as background pixels. (b) The input could be redundant and the information can be reconstructed using other available inputs.

Removing pixels in RGB images can create new patterns for models to predict. Therefore, human assumptions about the effect of removing pixels should be made only after careful consideration.

6.3 Project Setup MNIST

A. Choice of the MNIST dataset

Before starting a large computation on the Food-101 dataset, the method was tested on a smaller dataset (MNIST) and with only one interpretability method to test the programmed pipeline. The MNIST dataset consists of 10 classes and 70,000 different images. The images are 28x28 pixels and a pixel has a value in the range [0,1] in greyscale. It was chosen because it is a small dataset and the reconstruction of the method can be achieved easily. Furthermore, the dataset is only 50 MB in size and a model can be trained without significant computational effort. For the proof of concept, only integrated gradient [40] and a random baseline were computed.

B. Training the model

The model used in this thesis is a convolutional neural network architecture consisting of two convolutional layers, two pooling layers, and three linear layers. The model was trained for ten epochs with a learning rate of 1e-3 and an SGD optimizer with a momentum term of 0.9. The batch size was 64. Across five separate runs, the model achieved an average accuracy of $97.4\% \pm 0.4\%$ on the test data. The cross-entropy loss was selected for training.

C. Dataset preparation and Splitting

The dataset used in this thesis was normalized using the mean and standard deviation and was split using the PyTorch library's default splitting method (60,000 images for the training data and 10,000 images for the test data). This split has a fixed test and train set which includes a homogeneous distribution of the classes. The training data was shuffled randomly for the training runs. The same dataset split was used for all training runs to ensure consistency in the results.

D. Example of the modified dataset

The modified dataset is the result of combining the attribution map with the original image pixels $A \circ p$. In this operation, the most x% important pixels are replaced with the mean value of the dataset. In figure 6.1 the difference between the two altered datasets is visible. In the first row, the random baseline is visible, and in the second row, the evaluation of graying out pixels based on the attribution values of integrated gradient is shown. 10, 30, 50, 70 and 90% of the pixels are grayed out (left to right).

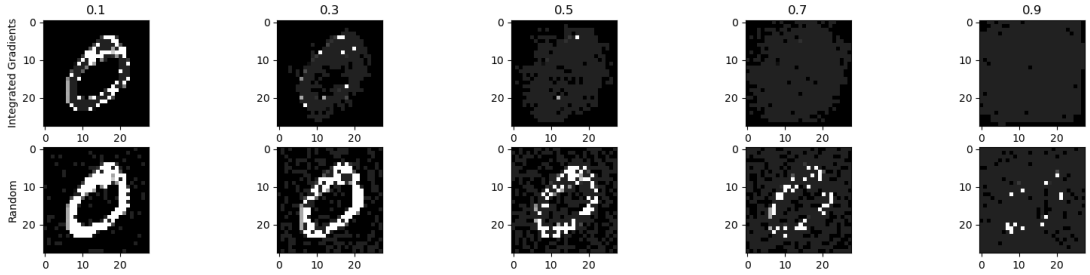

Figure 6.1: Integrated Gradients and Random Baseline Comparison

Pixels blocked	10%	30%	50%	70%	90%
Random Baseline	98.64%	97.76%	96.4%	90.9%	75.5%
std	0.42%	0.28%	0.39%	0.58%	0.65%
Integrated Gradient	95.4%	91.48%	86.64%	55.52%	10%
std	0.44%	0.83%	2.1%	31.64%	0%

Table 6.1: Accuracy and standard deviation of the retrained models per threshold.

Judging by human intuition, Integrated Gradients [40] seems to identify information more reliably. The model is expected to perform worse after retraining. However, human intuition can be problematic in neural networks as mentioned in section 6.2.

E. Retraining on the new datasets

To ensure that the model learns consistently, the same split and learning parameters as in the standard case were used to retrain the model on the newly modified dataset. An untrained network was created and then retrained for each training run. To validate the results and to minimize the effect of random seed initialization, the training was performed with 5 different seeds and 5 independent models were created. The exact results can be found in the appendix.

F. Advantages and Disadvantages of the setup

Due to the small size of the dataset and the limited number of classes, training a model is fast and effective. A disadvantage is the limited meaningfulness of this setup for other datasets and models. The majority of use cases primarily involve working with images in the RGB colour space, using deep learning models that are often more complex. In addition, RGB images combine 3 channels, making attribution maps more difficult to compute and understand.

G. Results

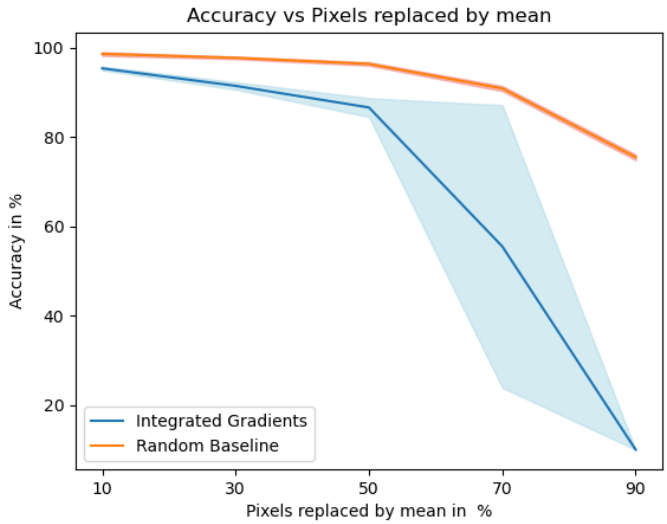

Figure 6.2: Accuracy: Random Baseline and Integrated Gradient

Figure 6.2 shows the decay in the accuracy of the random baseline and the models trained on pixels grayed out by integrated gradient. Integrated Gradient seems to have a faster accuracy decay, and at 90% of the grayed-out pixels, the classification is 10%, which is the same as a random classification. Note that Integrated Gradient has a very high standard deviation at 70%. This is because two control runs happened to learn a random classification, while the other three still managed to learn a classification. In this experiment, Integrated Gradient seems to identify more informative pixels than the random baseline for the MNIST dataset.
A possible explanation for this result, which differs from the ROAR experiments in that the integrated gradient identifies more important pixels for the network than the random baseline, is that for simple models, such as the one used in this experiment, and for simple datasets like the MNIST dataset, it is easier for attribution methods to find informative pixels. Additionally, the structure of a greyscale image is less complex than that of an RGB image.

6.4 Project Setup Food-101

After building a functional, fast pipeline for the ROAR algorithm, some of the experiments of the original paper were reconstructed. Originally, the goal of the thesis was to reconstruct Image Net. However, the computation time was too long with the available setup, so Food-101, which has only about 5 GB, was chosen.

A. Training the model

The model used in this thesis is ResNet50 [17]. The Food-101 [7] dataset was chosen for replication. The learning rate was set to 0.175 and the batch size was set to 64. Using an SGD optimiser

with momentum of 0.9, weight decay of 0.0001 and a scheduler at epoch 30 with gamma=0.1, an average accuracy of 70.5% $\pm$ 1.6 was achieved over 5 separate training runs on the test score (no validation split was used). Cross-entropy loss was used. A total of 31 epochs were run. By using this particular training setup, the model learns more about the specific data in the last epoch. In the last epoch, an accuracy of 95% was achieved on the training data.
In the original paper, an accuracy of 84.54% was achieved on the Food-101 dataset. A lower accuracy was achieved because different training parameters were used in our experiment. We used a smaller batch size of 64 instead of 256 and fewer epochs, 31 instead of 90. More fine-tuning could have been done to achieve a better test score, but the results were considered good enough for our experiments.

B. Dataset preparation and splitting

The data was resized using centre-crop to make all images 3x224x224. They were then normalised. No data enhancement such as flipping and mirroring was applied. The standard 75:25 split was used. In the original dataset, the training and test data have a fixed split, which we also used for this thesis. The saliency maps used to generate the datasets were computed using a single model. The original paper does not clearly state how the pixels are replaced. In these experiments, the input layers were considered as 3x224x224 maps and each channel of a pixel was replaced individually by the mean of that channel.

C. Retraining the models

The same procedure was followed as in the MNIST section 6.3. However, only 4 control runs were performed instead of 5. The exact results can be found in the Appendix.

B. Results

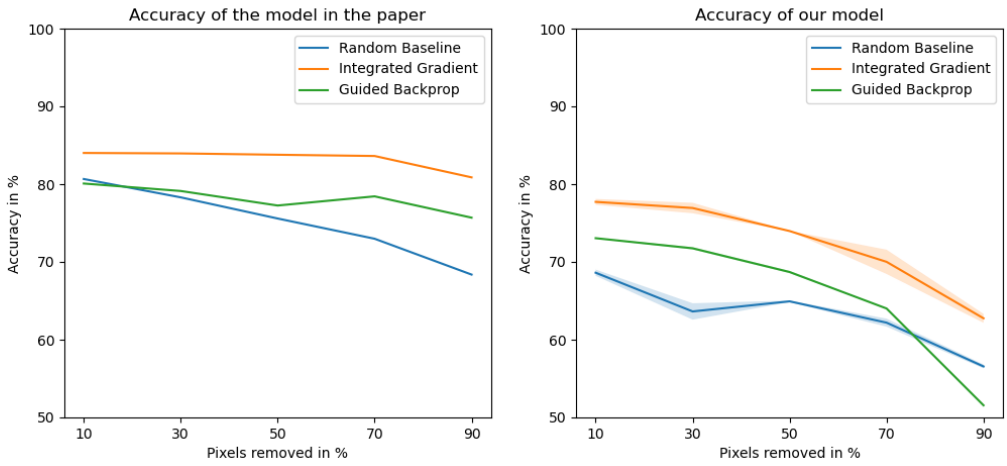

Figure 6.3: Comparison of the accuracy behaviour of the original model vs the modified model.

Pixels blocked	10%	30%	50%	70%	90%
Random Baseline	68.61%	63.63%	64.93%	62.19%	56.56%
std	0.40%	1.05%	0.10%	0.51%	0.25%
Integrated Gradient	77.72%	76.93%	73.97%	70.01%	62.75%
std	0.37%	0.67%	0.01%	1.56%	0.57%
Guided Backprop	73.05%	71.75%	68.70%	64.01%	51.56%
std	0.06%	0.15%	0.14%	0.01%	0.02%

Table 6.2: Accuracy and standard deviation of the retrained models per threshold

Figure 6.3 shows the accuracy of the retrained models in the original paper and the accuracy of our retrained model. Our experiments show similar results as the original paper for integrated gradient and guided backpropagation. The random baseline seems to block out more important pixels than the other attribution methods. The fact that the accuracy drops so slowly makes it unclear whether the method can be used to reliably test attribution methods. Our accuracy drops more quickly and this is probably due to the different training setups as well as the center-crop we employed. The standard deviations are small and it appears that the results can be replicated consistently.

7 Summary and Interpretation

Numerous attribution maps and possible evaluation methods have been looked at and it leaves data scientists with one question: Which evaluation method and attribution method should be used?
The unsatisfactory answer is: We do not know. Depending on the model and dataset used, different evaluation methods give varying outcomes. In Sanity check[2] and ROAR [19] some methods were seen as unsuitable for the task considered. However, this was dependent on the model and the dataset. Considering there exists no ground truth for an evaluation method, there also is no ground truth for the attribution method. Opting for an SG - GRAD or a VAR-GRAD approach might result in favourable results for ROAR and Sanity checks. However, it remains difficult to evaluate visual explanations and there is no established ground truth to determine which evaluation method is the best. Consequently, further research is required to provide clearer guidance in this area.

Additionally, ROAR has been reproduced on MNIST as an example. Similar trends have been found as in the original paper[19], despite a simplified setup. However, the performance of interpretability methods seems to vary for different datasets and models. This sensitivity to dataset and model makes it difficult to judge what is causing the result. It could be the attribution method, the dataset, the model, or the specific training setup of the model. It is difficult to make a clear statement. We conclude that choosing a different model and dataset leads to different results.

Although the underlying logic of ROAR makes sense from a human perspective, it seems difficult to apply in practice. Depending on the training setup, different results are obtained. Nevertheless, it shows how little we know about evaluating interpretability methods for deep neural networks.

Bibliography

[1] Captum Library. URL: Source:%20https://captum.ai/tutorials/Resnet%5C_TorchVision%5C_Interpret.

[2] Julius Adebayo et al. "Sanity checks for saliency maps". In: *Advances in neural information processing systems* 31 (2018).

[3] Genevera I. Allen, Luqin Gan, and Lili Zheng. "Interpretable Machine Learning for Discovery: Statistical Challenges \& Opportunities". In: *CoRR* abs/2308.01475 (2023). DOI: 10.48550/arXiv.2308.01475. arXiv: 2308.01475. URL: https://doi.org/10.48550/arXiv.2308.01475.

[4] Daniel W Apley and Jingyu Zhu. "Visualizing the effects of predictor variables in black box supervised learning models". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82.4 (2020), pp. 1059–1086.

[5] David Bau et al. "Network Dissection: Quantifying Interpretability of Deep Visual Representations". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 3319–3327. DOI: 10.1109/CVPR.2017.354. URL: https://doi.org/10.1109/CVPR.2017.354.

[6] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.

[7] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101–mining discriminative components with random forests". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*. Springer. 2014, pp. 446–461.

[8] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145%2F2939672.2939785.

[9] Piotr Dabkowski and Yarin Gal. "Real Time Image Saliency for Black Box Classifiers". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 6967–6976. URL: https://proceedings.neurips.cc/paper/2017/hash/0060ef47b12160b9198302ebdb144dcf-Abstract.html.

[10] Manuel Fernández Delgado et al. "Do we need hundreds of classifiers to solve real world classification problems?" In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 3133–3181. DOI: 10.5555/2627435.2697065. URL: https://dl.acm.org/doi/10.5555/2627435.2697065.

[11] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[12] Jerome H Friedman and Bogdan E Popescu. "Predictive learning via rule ensembles". In: (2008).

[13] Alex Goldstein et al. "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation". In: *journal of Computational and Graphical Statistics* 24.1 (2015), pp. 44–65.

[14] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[15] Brandon M. Greenwell, Bradley C. Boehmke, and Andrew J. McCarthy. "A Simple and Effective Model-Based Variable Importance Measure". In: *CoRR* abs/1805.04755 (2018). arXiv: 1805.04755. URL: http://arxiv.org/abs/1805.04755.

[16] Arushi Gupta et al. "New Definitions and Evaluations for Saliency Methods: Staying Intrinsic, Complete and Sound". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 33120–33133.

[17] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[18] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282.

[19] Sara Hooker et al. "A benchmark for interpretability methods in deep neural networks". In: *Advances in neural information processing systems* 32 (2019).

[20] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 3149–3157. ISBN: 9781510860964.

[21] Pieter-Jan Kindermans et al. "The (Un)reliability of Saliency Methods". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Ed. by Wojciech Samek et al. Cham: Springer International Publishing, 2019, pp. 267–280. ISBN: 978-3-030-28954-6. DOI: 10.1007/978-3-030-28954-6_14. URL: https://doi.org/10.1007/978-3-030-28954-6_14.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60.6 (2017), pp. 84–90.

[23] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[24] D Matthew Zeiler and Fergus Rob. "Visualizing and understanding convolutional neural networks". In: ECCV. 2014.

[25] Christoph Molnar. *Interpable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. 2022. URL: https://christophm.github.io/interpretable-ml-book.

[26] Christoph Molnar et al. "General Pitfalls of Model-Agnostic Interpretation Methods for Machine Learning Models". In: *xxAI - Beyond Explainable AI - International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*. Ed. by Andreas Holzinger et al. Vol. 13200. Lecture Notes in Computer Science. Springer, 2020, pp. 39–68. DOI: 10.1007/978-3-031-04083-2_4. URL: https://doi.org/10.1007/978-3-031-04083-2%5C_4.

[27] Antonio Mucherino, Petraq J. Papajorgji, and Panos M. Pardalos. "k-Nearest Neighbor Classification". In: *Data Mining in Agriculture*. New York, NY: Springer New York, 2009, pp. 83–106. ISBN: 978-0-387-88615-2. DOI: 10.1007/978-0-387-88615-2_4. URL: https://doi.org/10.1007/978-0-387-88615-2_4.

[28] Michael Neely et al. "Order in the Court: Explainable AI Methods Prone to Disagreement". In: *CoRR* abs/2105.03287 (2021). arXiv: 2105.03287. URL: https://arxiv.org/abs/2105.03287.

[29] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature visualization". In: *Distill* 2.11 (2017), e7.

[30] Liudmila Ostroumova Prokhorenkova et al. "CatBoost: unbiased boosting with categorical features". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al. 2018, pp. 6639–6649. URL: https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html.

[31] Simo Puntanen. "Methods of multivariate analysis, by alvin c. rencher, william f. christensen". In: *International Statistical Review* 81.2 (2013), pp. 328–329.

[32] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-Precision Model-Agnostic Explanations". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (Apr. 2018). DOI: 10.1609/aaai.v32i1.11491.

[33] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should i trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[34] Wojciech Samek and Klaus-Robert Müller. "Towards Explainable Artificial Intelligence". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer International Publishing, 2019, pp. 5–22. DOI: 10.1007/978-3-030-28954-6_1. URL: https://doi.org/10.1007%2F978-3-030-28954-6_1.

[35] Wojciech Samek et al. “Evaluating the Visualization of What a Deep Neural Network Has Learned”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.11 (2017), pp. 2660–2673. DOI: 10.1109/TNNLS.2016.2599820.

[36] Ramprasaath R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74. URL: https://doi.org/10.1109/ICCV.2017.74.

[37] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014. URL: http://arxiv.org/abs/1312.6034.

[38] Daniel Smilkov et al. “SmoothGrad: removing noise by adding noise”. In: *CoRR* abs/1706.03825 (2017). arXiv: 1706.03825. URL: http://arxiv.org/abs/1706.03825.

[39] Jost Tobias Springenberg et al. “Striving for Simplicity: The All Convolutional Net”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6806.

[40] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 3319–3328. URL: http://proceedings.mlr.press/v70/sundararajan17a.html.

[41] Sandra Wachter, Brent Mittelstadt, and Chris Russell. “Counterfactual explanations without opening the black box: Automated decisions and the GDPR”. In: *Harv. JL & Tech.* 31 (2017), p. 841.

[42] Mengjiao Yang and Been Kim. “Benchmarking attribution methods with relative feature importance”. In: *arXiv preprint arXiv:1907.09701* (2019).

[43] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. “Adaptive deconvolutional networks for mid and high level feature learning”. In: *2011 International Conference on Computer Vision* (2011), pp. 2018–2025. URL: https://api.semanticscholar.org/CorpusID:975170.

[44] Daniel Zhang et al. “The AI Index 2022 Annual Report”. In: *CoRR* abs/2205.03468 (2022). DOI: 10.48550/arXiv.2205.03468. arXiv: 2205.03468. URL: https://doi.org/10.48550/arXiv.2205.03468.

[45] Jianming Zhang et al. “Top-down neural attention by excitation backprop”. In: *International Journal of Computer Vision* 126.10 (2018), pp. 1084–1102.

[46] Tianyi Zhang et al. "A Survey of Diffusion Based Image Generation Models: Issues and Their Solutions". In: *arXiv preprint arXiv:2308.13142* (2023).