UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA RAČUNARSTVO I INFORMATIKU

Primjena LIME metode za objašnjivost u algoritmima dubokog učenja

ZAVRŠNI RAD

- PRVI CIKLUS STUDIJA -

Student:
Ahmed Raščić

Mentor:
Doc. dr Senka Krivić

Sarajevo,
Juli 2024

UNIVERSITY OF SARAJEVO
FACULTY OF ELECTRICAL ENGINEERING
DEPARTMENT OF COMPUTING AND INFORMATICS

Application of the LIME method for explainability in deep learning algorithms

FINAL THESIS

- BACHELOR STUDIES -

Candidate:
Ahmed Raščić

Supervisor:
Doc. dr Senka Krivić

Sarajevo,
July 2024

Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za Računarstvo i informatiku

Izjava o autentičnosti radova

Završni rad
I ciklusa studija

Ime i prezime: Ahmed Raščić
Naslov rada: Primjena LIME metode za objašnjivost u algoritmima dubokog učenja
Vrsta rada: Završni rad Prvog ciklusa studija
Broj stranica: 39

Potvrđujem:

- da sam pročitao dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- da sam svjestan univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- da sam jasno naznačio prisustvo citiranog ili parafraziranog materijala i da sam se referirao na sve izvore;
- da sam dosljedno naveo korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- da sam odgovarajuće naznačio svaku pomoć koju sam dobio pored pomoći mentora i akademskih tutora/ica.

Sarajevo, Juli 2024

Potpis:

_____

Ahmed Raščić

Elektrotehnički fakultet, Univerzitet u Sarajevu
Odsjek za Računarstvo i informatiku.
Doc. dr Senka Krivić, dipl.el.ing
Sarajevo, Juli 2024

Postavka zadatka završnog rada I ciklusa:

Naziv teme: Primjena LIME metode za objašnjivost u algoritmima dubokog učenja

Tema ovog diplomskog rada je istraživanje i razumijevanje primjenjivosti metode LIME (Local Interpretable Model-agnostic Explanations) u području algoritama dubokog učenja. Ovo istraživanje ima za cilj da se udubi u tehnike i principe LIME-a kako bi složene AI modele učinili razumljivijim i transparentnijim za ljudske korisnike. Studija će uključivati dubinsku analizu LIME-ovih mogućnosti i ograničenja, praktičnu implementaciju u AI sistemima i sveobuhvatnu procjenu njegove djelotvornosti u pružanju smislenih objašnjenja za odluke AI modela.

Mentor: Doc. dr Senka Krivić

Polazna literatura:

- "Why Should I Trust You?": Explaining the Predictions of Any Classifier Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin
- Ribeiro, M. T., Singh, S., Guestrin, C., "Anchors: High-precision model-agnostic explanations", Proceedings of the AAAI conference on artificial intelligence, Vol. 32, No. 1, 2018

Doc. dr Senka Krivić, dipl. ing. el.

Sažetak

Ova teza se bavi ključnim pitanjem interpretabilnosti u mašinskom učenju (ML), temom od rastućeg značaja kako se ML modeli a posebno duboke neuronske mreže, sve češće koriste u kritičnim sektorima poput zdravstva i finansije. Netransparentna priroda ovih modela predstavlja značajne rizike, sprečavajući njihovo šire prihvatanje i naglašavajući potrebu za efikasnim tehnikama interpretabilnosti kako bi se garantovala njihova pouzdanost i pravednost. Ovo istraživanje se fokusira na LIME (Local Interpretable Model-agnostic Explanations) tehniku, upoređujući je sa SHAP-om (SHapley Additive ExPlanations) i Anchors-ima kako bi se procijenila njihova efikasnost u objašnjavanju procesa donošenja odluka u modelima mašinskog učenja. Primijenjena na ResNet50 neuronsku mrežu obučenu na ImageNet skupu podataka, studija provodi detaljnu analizu, usredsređujući se na sposobnost metode da učini razmišljanje mreže transparentnijim. Cilj je izvršiti evaluaciju ovih metoda pod različitim uvjetima korištenjem kvalitativnih i kvantitativnih pristupa, uključujući korisničku studiju, kako bi se utvrdilo koja tehnika nudi najrazumljivija objašnjenja i pod kojim okolnostima. Očekuje se da će očekivani rezultati otkriti različite snage i slabosti svake metode, poboljšavajući primjenu i razvoj transparentnijih i odgovornijih AI sistema. Ovaj rad naglašava ulogu interpretabilnosti u izgradnji povjerenja i osiguravanju etičke upotrebe AI tehnologija.

Ključne riječi: interpretabilnost mašinskog učenja, objašnjivost, LIME, SHAP, Anchors

Abstract

This thesis addresses the crucial issue of interpretability in machine learning (ML), a topic of increasing importance as ML models, particularly deep neural networks, are employed in critical sectors e.g. healthcare and finance. The opaque nature of these models poses substantial risks, impeding their broader acceptance and underscoring the need for effective interpretability techniques to guarantee their reliability and fairness. This research focuses on the LIME (Local Interpretable Model-agnostic Explanations) technique, comparing it with SHAP (SHapley Additive exPlanations) and Anchors to evaluate their effectiveness in explaining the decision-making processes of machine learning models. Applied to the ResNet50 neural network trained on the ImageNet dataset, the study conducts a detailed analysis, centering on the method's capability to make the network's reasoning more transparent. The goal is to assess these methods under various conditions using qualitative and quantitative approaches, including a user study, to determine which technique offers the most understandable explanations and under what circumstances. Anticipated results are expected to reveal distinct strengths and weaknesses of each method, enhancing the deployment and development of more transparent and accountable AI systems. This work emphasizes the role of interpretability in building trust and ensuring the ethical use of AI technologies.

Keywords: machine learning interpretability, explainability, LIME, SHAP, Anchors

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement and Rationale

For any written program, the question of its testing and acceptance looms. How can we increase our confidence that a piece of software does perform as specified under a certain set of conditions? For practical purposes, as computer systems grow in complexity and capability, the question mentioned above of testing and acceptance evolves into an industrial undertaking of its own. We don't need to delve too deep into the field of software verification and validation to understand that the general question of software acceptance is anything but a trivial problem. Software, in general, is a very brittle technology, and the art and craft of software engineering at its best allows us to construct software that is robust under complexity.

Despite the complexity, we still manage. After all, nothing in any program is beyond statements, loops, and conditionals. Moreover, computer programs are sequences of clear, precise, and well-defined instructions to be executed by a computing platform. Therefore, it follows that at least in principle, we are always able, independent of runtime details, with more or less effort, to reproduce the execution of a certain algorithm under given conditions and gain detailed insight and understanding about the behavior of a program under study. In other words, we can understand why the algorithm behaves the way it does, and starting from that understanding, we can perhaps modify, fix, or extend the algorithm in ways we deem fit.

Therefore, if we have the code, the ability to run the code under various conditions, and the ability to inspect the execution of the code at various points of interest - the whole program becomes a white box to us, meaning a system that we can thoroughly analyze from the inside out to fully understand its behavior. Without this ability, programming would be a hopeless practice, and we would be able to construct only minimal programs of severely limited capability. The fact that we can in principle gain such detailed insight into our programs is of surprise to no one. Again, after all, it's all just statements, loops, and conditionals [2].

But there is a complication. We have hitherto assumed that the code's author was a human, thinking in ways and patterns recognizable to us and considering a certain problem to solve. We expect the code to be well written by its author, specifying the instructions for the computer to execute for it to achieve a known goal. We expect the code to be a "how-to" recipe for a computer to perform a certain task.

In machine learning, that cannot be the case. We want to avoid telling the computer how to do something as much as possible. We do not know how we do many of our everyday tasks. For example, we do not know how we recognize familiar faces or voices. It is a matter of "I know it when I see it". We are, therefore, utterly incapable of telling a computer how to recognize a face or a voice in the classical manner of programming. There are many examples of such problems

where an explicit program is impossible to construct classically. Machine learning therefore is not as much concerned with specifying a "how-to" of a specific task, rather the focus of the craft is all about figuring out and constructing computational paradigms of programmatic learning which is a very different task.

We, as humans, learn tasks that cannot be easily reduced to a specific sequence of algorithmic steps. Most of the interesting computations we perform are of this nature. This algorithmic irreducibility notwithstanding the idea is that maybe we can develop statistical learning algorithms capable of tackling tasks like these, which no simple algorithm can develop directly. The chain of abstraction can then be repeated ad infinitum. If we can learn a task algorithmically, maybe we can also learn how to learn algorithmically, etc. This, however, robs us of a fundamental assumption we made about programs. The results of these learning algorithms are not going to be well-written, concise, readable lines of code that we can then run under various conditions to gain insight into their performance. Instead, they are going to be some kinds of statistical models of a given task.

Machine learning recognizes many different learning algorithms and models [3]. Whereas a learning algorithm is the specific way in which a particular model is trained for a given task based on some data, a machine learning model is an abstract mathematical structure that can be finetuned by a learning algorithm so that after training we have a mathematical structure that specifically models the task we are trying to learn. Bayesian networks, support vector machines, and decision trees are some examples of general learning models but the most advanced family of models (in terms of performance) is usually the artificial neural network.

It is said that neural networks are black box models in the sense that their functioning is not particularly transparent. This presents a verification and validation challenge concerning utilizing neural networks, which we are highly motivated to employ in various tasks because of their sophistication and precision. At the same time, we are uncomfortable with the fact that we cannot really, at this point, make the functioning of the neural net completely transparent to our inspection. That opens up a significant knowledge gap about the network itself and makes us less confident about deploying the technology, especially in domains where the cost of a mistake is high, such as healthcare [1]. This is a crucial point and the fundamental problem that is the backdrop of our discussion. To see why neural networks are black boxes and not transparent, we will provide a brief overview of their fundamental working concept.

The basic structure of artificial neural networks is a sequence of sets of artificial neurons (neuron layers) that can be loosely thought of as substeps of a target computation that we want to learn. The adjacent layers communicate with each other in a complex manner. The starting point is the first layer that corresponds to an immediate representation of the input to the network - say, the brightness values of every pixel in a grayscale picture of a hand-written digit to be recognized. Then the computation moves through layers of neurons, eventually reaching the final layer - which, in the case of our example with digits, could be ten neurons where the value of each represents the probability that our neural net estimates the input picture to be a digit of zero, one, two, etc.

The interesting part happens in the layers between. To understand how the layers compute the result, we need to understand the basic idea behind a single artificial neuron [4]. The artificial neuron can be thought of as a function. The idea is that the input of a neuron is going to be the output of many other neurons (from the layer before) - this is a vector of inputs. Then, the specific function corresponding to a given neuron is defined by a set of parameters - the neuron's weights (a weight vector) and the neuron's offset.

The neuron calculates the dot product of input and weight vectors and adds the offset. The resulting number is then passed through an activation function. Up to this point, everything we

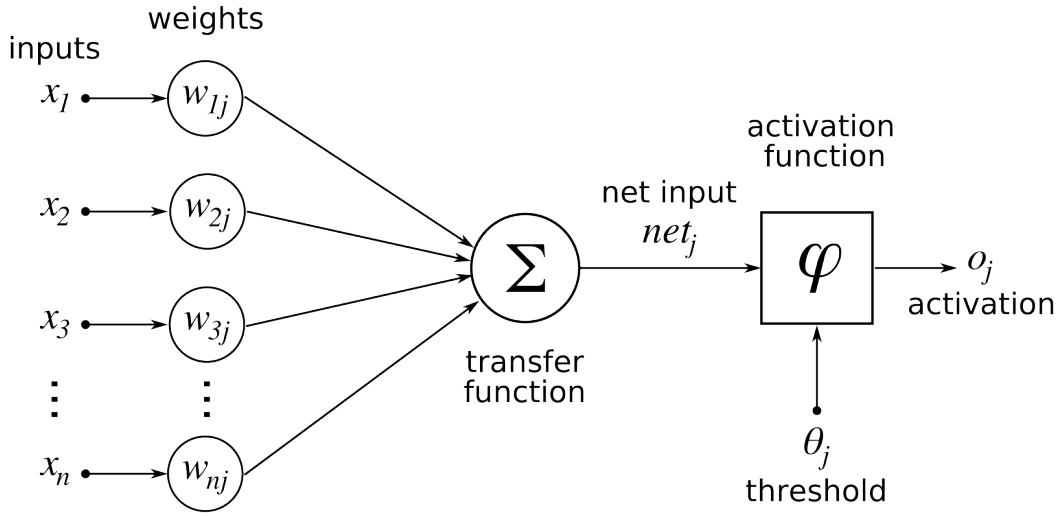

Figure 1.1: A graphical representation of an artificial neuron.

have computed is linear, which would make for quite a limited model since it would be unable to compute many interesting non-linear tasks. Therefore, the activation function is a non-linear function, the specific choice of which has both theoretical and engineering implications. The simplest choice would be a step function, whereas the most used functions are the sigmoid function and its derivative softmax, as well as the rectified linear unit (ReLU).

In any case, the result of the dot product plus the offset is forwarded to the activation function, which then is taken as the final output of the neuron and then forwarded again toward the next layer until we get to the final layer, which represents the output of the neural net as a whole. The computation of neural nets, therefore, is determined by their weights and offsets and also by the way their neurons are interlinked. This is of critical importance since precisely these parameters are determined by a learning algorithm suited for training neural networks.

The learning algorithm takes many examples as inputs (properly labeled in the case of supervised learning) and then strives to finetune the network's weights and offsets so that the network's output converges towards the correct result as closely as possible. It is known that neural networks can be universal function approximators and also can be Turing-complete [5, 6]. The fact that neural networks are universal function approximators is remarkable since they have tremendous expressive power - even to compute black-box functions we don't know a specific algorithm for despite knowing the correct input-output pairs, which is exactly what we are interested in.

On the other hand, the fact that neural networks can be Turing-complete means that, at least in theory, the computation of the neural net could be reduced to a formal algorithm, which has important implications for the interpretability of neural networks. Namely, suppose we can extract an algorithm from a working neural network. In that case, we can express its learned logic in terms of conditionals, statements, and loops - which is familiar territory that is, in relative terms, organically interpretable. Given the structure of neural nets, the total number of weights is described as growing quadratically with the number of neurons per layer. The famous AlexNet classifier possesses 60 million parameters [7]. The recently popular large language model GPT 3 has 175 billion parameters. The GPT 4 iteration has 1.76 trillion parameters [8].

These are massive models. Given that the capability of the network scales well with the number of neurons, and that the number of parameters therefore grows exponentially, we see the challenge of intepretability. For one, loops, conditionals, and statements are nowhere to be seen anymore - there are just millions, tens of millions, and these days billions of neurons that

perform some computation. All the parameters of the network obviously must mean something since they play some role in the computation, but we have no easy way of figuring out what exactly one parameter, one neuron and even one layer represent.

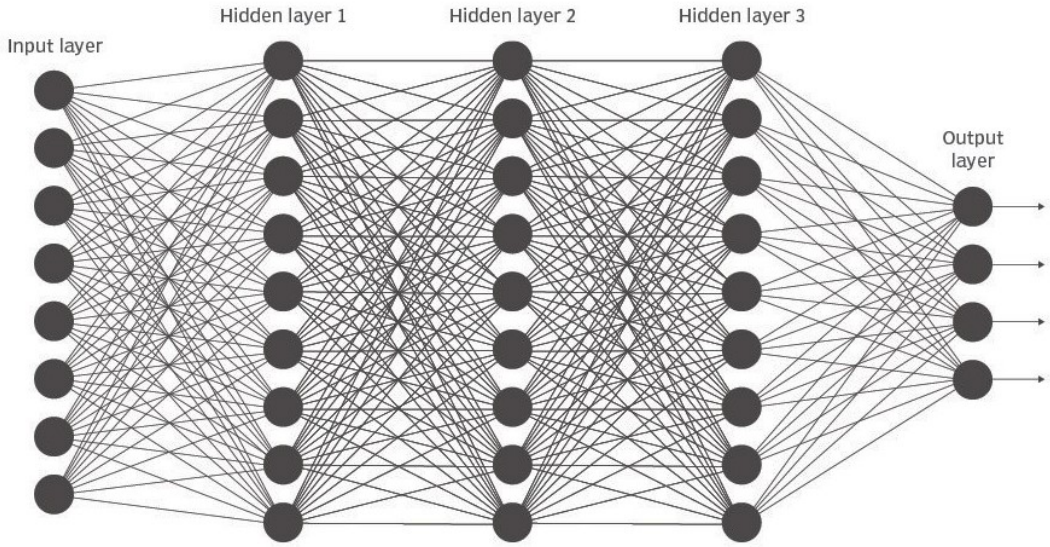

Figure 1.2: A tiny neural network.

Even in this tiny network it is difficult to visually discern all the edges between the layers let alone figure out what role they play in the computation of the network.

We are unable in any trivial sense just to inspect the weights and gain insight and understanding. How does it work? Is the reasoning behind seemingly correct answers correct in its own right? Does the network express unexpected biases that might have been hidden all along in the training data? When the network makes mistakes, what knowledge did it fail to capture? What corrective actions might be suitable to further improve the neural network's performance? All of these and many similar questions are questions to which the fields of ML (machine learning) interpretability and, in general, Explainable Artificial Intelligence (XAI) are of the utmost importance.

It is precisely because of this that we call neural networks black box models, meaning models whose operation is not transparent from the inside out. We can see the inputs and outputs of a neural network, and we might even have full access to all of its parameters, but we cannot easily deduce how or why the neural network computes a given output exactly. This is also the reason why simply releasing a model's weights is not enough to qualify it as open source in the traditional sense since the term takes on a different meaning in the context of machine learning [9]. The opaqueness of neural networks is one of the challenges of XAI and ML interpretability. We can build sophisticated AI systems that are of immense value. But how confident can we be in their functioning when they are deployed in the wild?

One definition of artificial general intelligence (AGI) can be found in the charter of OpenAI - the organization that developed the ChatGPT system that was declared the fastest-growing application of all time [1].

[1] `https://www.theverge.com/2023/11/6/23948386/chatgpt-active-user-count-openai-developer-conference`

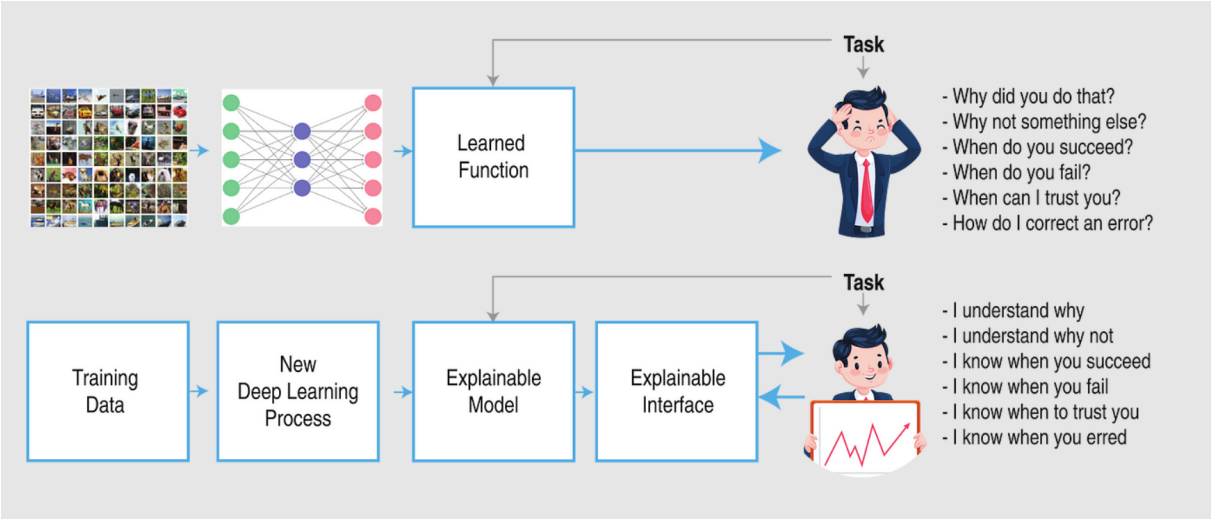

Figure 1.3: Problem and goal of XAI and interpretable ML.

They define AGI as "...highly autonomous systems that exceed human performance in most economically valuable tasks..."[2]. Authors and experts differ on whether AGI also implies superintelligence. One possible definition of superintelligence reads "By superintelligence we mean an intellect that is much smarter than the best human brain in virtually all areas, including scientific creativity, general wisdom, and social skills."[10]. In any case, the concept of superintelligence is most often mentioned in the context of discussions about the existential risks of general artificial intelligence.

There is a general understanding that the technologies of artificial intelligence and, therefore, machine learning have potentially very dangerous use cases. Cheap to produce AI-enhanced weapons, population surveillance and control, misinformation operations and election interference, scams via voice mimicry... The list goes on and on. In March of 2023, the Future of Life Institute published an open letter titled "Pause Giant AI Experiments: An Open Letter" in which they called for at least a six-month pause in the development of AI models more powerful than GPT 4, citing security concerns[3].

The same year, a "Statement on AI risk of extinction" was also published and signed by many prominent stakeholders. In it, the signatories beseech that mitigating the risk of AI-caused human extinction should be a global priority. On a more grim note, Eliezer Yudkowsky[4], an AI researcher is utterly pessimistic and is convinced that we are already most probably on the path to extinction since he believes that the development of AI capabilities vastly outpaces the speed of progress in solving the alignment problem and will continue to do so[5].

If we cannot interpret or understand the technology we are building - given that there seem to be significant risks involved - should we be building it at all? We of course will continue developing these systems, but the more seriously we take AI safety (and that seems to be the trend) there is going to be, and already is, a tremendous research incentive in the fields of XAI and ML interpretability to develop trust-building techniques so that we can put these sophisticated and powerful systems under our control and ensure that they are developed and put to use responsibly and beneficially.

[2] https://openai.com/charter/
[3] https://futureoflife.org/wp-content/uploads/2023/05/FLI_Pause-Giant-AI-Experiments_An-Open-Letter.pdf
[4] https://en.wikipedia.org/wiki/Eliezer_Yudkowsky
[5] https://time.com/6266923/ai-eliezer-yudkowsky-open-letter-not-enough/

1.2 Hypothesis

Interpretable machine learning models are crucial in validating the reliability and fairness of decisions made by complex algorithms, especially in high-stakes domains such as healthcare and finance. Methods like LIME, SHAP, and Anchors provide frameworks to approximate and explain the decisions of opaque machine learning models. This research hypothesizes that each interpretability technique offers distinct advantages depending on the nature of the data, runtime constraints, and preferred explanation characteristics. By comparing these methods, we aim to identify which technique most effectively elucidates model decisions across various scenarios, especially for deep neural networks. This could lead to more nuanced insights into how machine learning models operate, potentially guiding the development of more transparent and accountable AI systems that integrate tools in consideration of interpretability and explainability. Thus, this study seeks to determine the conditions under which each method best operates, hypothesizing that the effectiveness of LIME, SHAP, and Anchors varies based on the end-user characteristics, available computational resources, underlying model type, and the interpretability demands of the application context.

1.3 Description of the Proposed Solution, Objectives of the Work, and Methodology

1.3.1 Description of the Proposed Solution

This thesis proposes an in-depth analytical comparison of three advanced model-agnostic interpretability algorithms—SHAP (SHapley Additive exPlanations), LIME (Local Interpretable Model-agnostic Explanations), and Anchors. These techniques are designed to enhance the transparency of machine learning models by providing insights into the reasoning behind their predictions. These algorithms treat the model as a black box and are specifically engineered to decipher and explain individual predictions made by complex models, such as deep neural networks.

Our approach involves detailing and evaluating each method based on its ability to produce interpretable explanations for the output of a ResNet50 convolutional neural network, trained on the ImageNet dataset. The SHAP and LIME methods focus on attributing the prediction output to the most influential input features through a process of input perturbation and feature importance analysis. On the other hand, the Anchors method determines portions of input data that securely anchor the model's predictions, thus providing rule-based explanations. The GitHub repository that contains code used to generate explanations and test the methods can be found at `https://github.com/arascic1/ML-Interpretability`.

1.3.2 Objectives of the Work

The core objectives of this thesis center on a comprehensive evaluation and comparison of three principal model-agnostic interpretability techniques: SHAP, LIME, and Anchors. Our goal is to assess each method's capability to clarify the decision-making processes of the ResNet50 deep neural network trained on the ImageNet dataset. An integral part of this objective includes conducting a user study to gauge the satisfaction levels of layman end users with the explanations these methods provide. This will help us to determine the practical utility of these interpretability techniques in real-world applications in different use case circumstances. Addi-

tionally, this thesis aims to analyze and compare the computational efficiency and performance of these methods. A thorough review of the existing literature will be undertaken to encapsulate the current state of research on model interpretability, identifying gaps and suggesting potential future research directions.

1.3.3 Methodology

- **Dataset and Model Selection:** Utilize the ImageNet dataset and the ResNet50 convolutional neural network, which is well-documented and widely used in image classification tasks, ensuring that the findings are relevant and comparable to current standards in the field.

- **Implementation of Interpretability Methods:** Implement the three interpretability methods: SHAP, which uses game theory to attribute output to inputs; LIME, which perturbs input data and observes the changes in predictions to identify influential features; and Anchors, which determines conditions under which model predictions remain unchanged, thus providing rule-based explanations.

- **Experimental Setup:** The experiments were designed to rigorously assess the interpretability methods SHAP, LIME, and Anchors through a series of scenarios. These scenarios involved five pairs of images, each displaying the original image, its classification, confidence level, and the explanation provided by the selected interpretability method. This setup was chosen to cover a variety of typical challenges in image classification, including incorrect classifications, correct classifications with low confidence, and correct classifications with high confidence.

- **Qualitative Analysis:** Conduct a qualitative analysis of the explanations generated by each method. This will involve a detailed examination of the interpretability outputs to assess how comprehensible and informative they are.

- **User Study:** The user study involved undergraduate and master's students in computer science, serving as layman end users, who evaluated the clarity, detail, and usefulness of the explanations using the Hoffman Explanation Satisfaction Scale. This Likert-type questionnaire helped assess participant satisfaction with each method's explanations across different attributes. The study was crucial for understanding how non-experts perceive and value the information provided by each interpretability technique in real-world scenarios.

- **Performance Evaluation:** Perform a runtime analysis to evaluate the computational efficiency of each method. This will include measuring the time taken by each method to generate explanations and comparing their scalability and practicality in operational settings.

- **Literature Review:** A literature review was conducted to examine existing research on interpretability methods such as SHAP, LIME, Anchors, and others, focusing on their applications, research trends, and methodologies.

- **Comparative Analysis and Synthesis:** Compare the results obtained from the qualitative analysis, user study, and performance evaluation to synthesize findings that highlight the strengths and limitations of each method.

1.4 Expected Results

This research aims to provide a detailed comparative analysis of the interpretability methods SHAP, LIME, and Anchors within the context of image classification by deep neural networks. We anticipate demonstrating distinct strengths and limitations of each method in terms of explanation precision, detail, completeness, etc. as evaluated by both qualitative analysis and user satisfaction scores. Specifically, we expect to identify which method provides the most comprehensive and understandable explanations to layman end-users, thereby enhancing trust and transparency in AI applications. Additionally, performance evaluations will likely reveal differences in the computational efficiency of each method, providing insights into their behavior in various hyperparameter settings. A thorough synthesis of qualitative, quantitative, and user-based feedback is expected to offer a nuanced understanding of how each method functions under different scenarios. This will guide future improvements in model interpretability techniques and potentially propose new directions for research in explainable AI.

1.5 Thesis overview

The primary focus of this thesis is to perform a comparison of three prominent interpretability algorithms for machine learning models: SHAP (SHapley Additive exPlanations), LIME (Local Interpretable Model-agnostic Explanations), and Anchors [11, 12, 13]. All of the aforementioned methods are model-agnostic methods, which means that they treat the model under investigation completely as a black box and do not depend at all on knowledge of any model internals on their operation.

Apart from being model-agnostic, they are also local, meaning that they are used to explain specific predictions or classifications of a given model and are not tasked with providing a sort of general, top-level global overview of the model's behavior overall. That being said, despite their locality, by using these methods it is possible to purposefully assemble a collection of individual explanations for a given model in such a way that the generated set is representative of the overall model behavior on a global level. For example, the paper on LIME [12] contains a section where this approach is described by developing a submodular pick algorithm. Global explanations are however outside the scope of this thesis.

These models will be described in detail in the following chapter. Right now it will suffice to say that LIME and SHAP are attribution methods, which means that they analyze and calculate which features of the input affect the output in a relevant way and then they attribute the output relevance amongst detected input features. It will be seen that LIME and Anchor methods perform some well-defined perturbation of the input data and then analyze how the model responds to the perturbed data, and based on those perturbed input-output pairs models can deduce features and their respective relevance and influence on the output itself. Anchors and SHAP also rely on some concepts from reinforcement learning and game theory respectively.

The Anchors method is a bit different than feature attribution methods in that it extracts a portion of the input that locks the prediction or classification, in the sense that changes outside the detected portion do not affect the prediction, and for example, inserting the detected portion into other unrelated and differently classified images can shift their classification towards the one that corresponds to the extracted portion. In that sense, the Anchor method will find the portion of the input that "anchors" the prediction in place, or in other words it will extract a feature of the input which can then be used to frame the classification task in terms of an if-then rule with the detected feature thought of as an antecedent and the target classification representing the consequent.

This thesis is focused primarily on analyzing the aforementioned interpretability algorithms on the task of image classification by deep neural networks. The neural network used for this thesis was the ResNet50 convolutional neural network that is fifty layers deep with a total of more than 25 million trainable parameters [14]. We have used the variant of ResNet50 that was trained on the ImageNet dataset available in Keras [15]. The state-of-the-art (SOTA) accuracy of the ResNet50 model on the ImageNet dataset was accomplished in 2020 and equals 83.2% [16]. For comparison, the current SOTA for ImageNet is at 92.4% accuracy, set by OmniVec in 2023 [17].

There are four levels of comparison performed in this thesis. First of all, methods are going to be compared qualitatively to each other on a variety of aspects that are relevant to ML interpretability. Secondly, we developed a user study that was administered to a representative sample of subjects that measured various aspects of user explanation satisfaction via developed metrics among the three interpretability methods [18]. Thirdly, we performed a runtime analysis of the methods by varying various hyperparameters of the methods to develop an estimate and comparison of the time complexity amongst the algorithms under investigation. Lastly, we reviewed the literature where these methods were developed and analyzed further to outline possible future research directions.

Chapter 2

Defining interpretabilty

It is understood that there are machine learning models that are intrinsically interpretable, that is, they are complex enough to accurately model certain target relationships and yet simple enough so that we can understand the way they work simply by inspecting their learned parameters in the context of the model's working architecture. Let's briefly cover some examples. The following equations should be considered as having a probability distribution around them as a mean, which for clarity we will not explicitly write. The terms response and activating variable are to be understood as dependent and independent variables respectively. Consider the case of the simplest linear regression model - a linear function with one dependent variable fitted to a certain dataset, say by error square minimization.

$$y = \alpha_0 + \alpha_1 \cdot x \tag{2.1}$$

This is trivial to interpret. There may be some α_0 response to an absence of x, and then for every increment of x our response changes uniformly by α_1. In other words, whatever we are modeling has a baseline response of α_0 and a sensitivity to x of α_1. The interpretative perspective stays the same in the general case.

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}, \qquad i = 1, \ldots, n \tag{2.2}$$

For some response to an i-th p-dimensional data point from a dataset of n instances. The interpretative stance is unchanged. We have a vector $\boldsymbol{\beta}$ of response sensitivities to respective activating variables and a baseline response for no activation. If we wish to explain a certain prediction, we could for example just rank the top five elements of $\boldsymbol{\beta}$ by absolute value. Notice that by doing that, we not only have a local explanation for a given instance, but since the model's parameters $\boldsymbol{\beta}$ are invariant concerning i, we immediately obtain a global description of how in general the response variable behaves concerning the model's features (activating variables).

The linear model, though exceedingly simple to interpret is severely limited since it is a priori bound to make sense only for inherently simple linear phenomena. In other words, the interpretability of the linear model is a consequence of the simplicity of the underlying phenomena being modeled. However, their expressive power can be greatly improved in various cases by introducing non-linearity through the features themselves. For example, we could fit a polynomial instead of a hyperplane. The model is still considered linear even though the features themselves might not be, that is the model stays linear relative to its parameters $\boldsymbol{\beta}$. In other words, the following model is linear and indeed is a hyperplane in a space with the appropriate polynomial base.

$$y_i = \beta_0 + \beta_1 \cdot x_{i1} + \beta_2 \cdot x_{i2}^2 + \cdots + \beta_p \cdot x_{ip}^p, \qquad i = 1, \ldots, n \tag{2.3}$$

We now have obtained a more powerful model but have not lost much if anything in terms of interpretability and we have a rich mathematical apparatus that is very well suited to study properties of polynomials. To see that this model is indeed more powerful, consider what would a simple linear model return for data generated by $\sin x$.

Even though technically speaking linear models are intrinsically interpretable, that is only the case for situations where the vector $\boldsymbol{\beta}$ is of relatively low dimensionality or at least where there is a relatively low-dimensional subset of the vector components that are by absolute value significantly larger than the rest. If we have a linear model of say one thousand variables where all of them have similar weights - we do not gain much understanding from such models.

Imagine now a model from (1.3). This model can exactly fit a dataset consisting of $p+1$ points. That might sound good however, such precision comes at the cost of ever-greater variability of the model between the points. This is the phenomenon of overfitting. Just because we have fitted the underlying data precisely doesn't mean that we have extracted the true relationship behind the data since if all we are doing is fitting to data exactly - we are completely ignoring the fact that there is always some noise in the data itself and that we cannot take the datapoints to be the ground truth themselves - if we are to have a nontrivial model, we need to model the noise in the dataset too. Figure 1.3 visually explains the problem of overfitting.

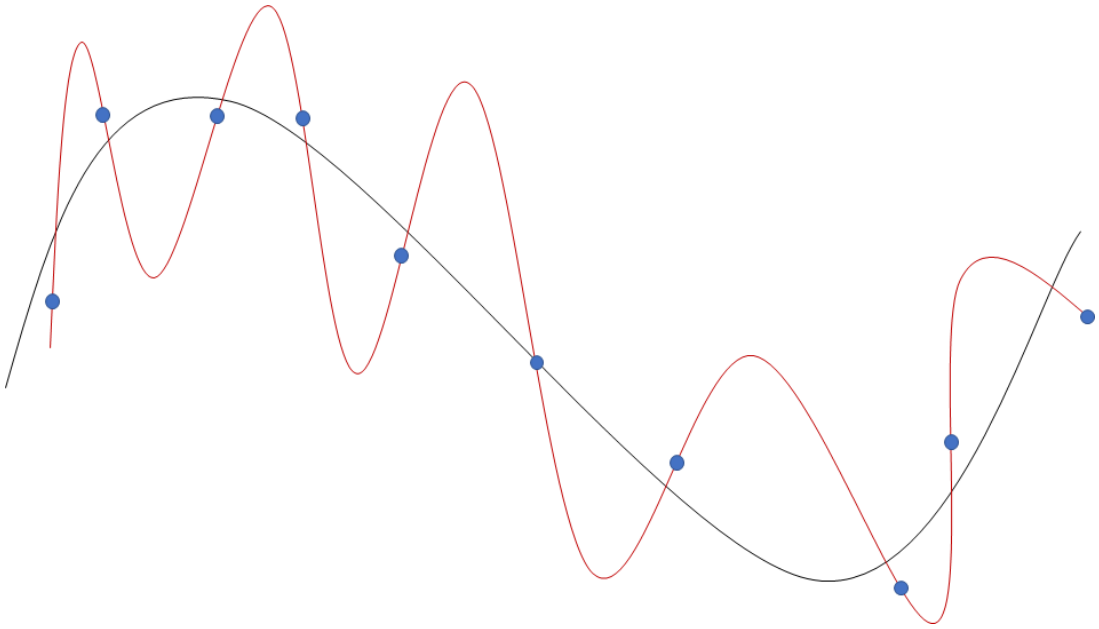

Figure 2.1: Overfitting arises from ignoring noise in data.

Ground truth is the black line and we have a sample dataset with noise. The red polynomial fits exactly every single point but is way off the mark about the true underlying relation.

One should also note the curious relationship between model precision, overfitting, and interpretability. In Figure 1.3 we have an overfitted model to a set of 10 points. We therefore know that the model from (1.3) itself has to be of degree 9. That means we have 9 features to which our model responds and maps them to the final value. We notice that by simply minimizing the training set error, we increase model complexity (the degree of the polynomial), thus increasing the probability of overfitting while also reducing interpretability.

One consequence of this is that while training a model on some data we usually employ cost functions of the following form.

$$\Omega = E + R \tag{2.4}$$

Where E is some chosen measure of training set error while R can be thought of as a complexity measure of the underlying model that is minimized alongside the error, to prevent overfitting and increase interpretability. The specific choice of R is an implementation detail. For example in our simple case of fitting a polynomial, it could be the degree of the polynomial used (akin to L_0 regularization). For decision trees, it could be the depth of the tree. In ensemble models, it could be the number of sub-models the ensemble relies on, etc. Figure 1.4 is a visual demonstration of how regularization improves both model accuracy (by preventing overfitting) and model interpretability (by reducing the number of relevant features).

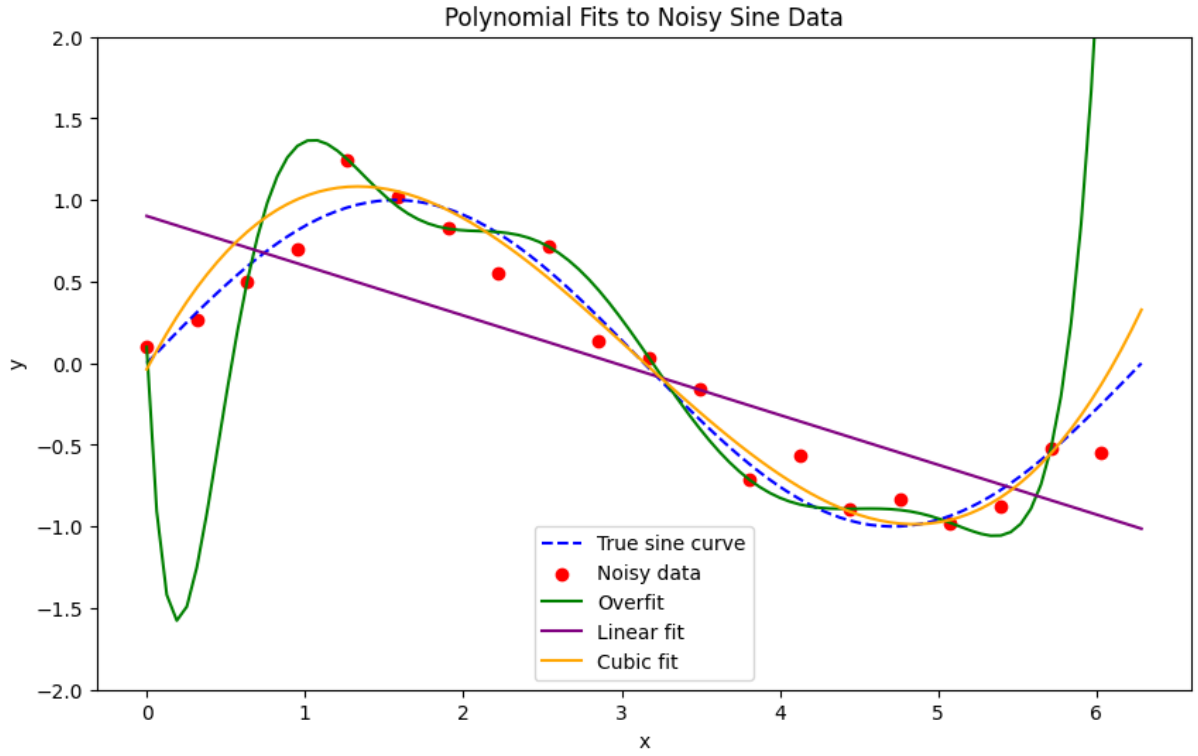

Figure 2.2: A cubic polynomial is the best fit for noisy sine data

The dashed blue line represents the true sine curve, while the red dots indicate the actual noisy data points. Three polynomial fits are demonstrated: an overfit model shown by the green curve, a basic linear fit depicted by the purple line, and an effectively approximating cubic fit represented by the yellow line. The cubic fit line closely follows the underlying sine wave, striking a balance between simplicity and precision. In contrast, the green curve tracks the random noise too closely thus missing the true underlying relation whereas the purple line is too simple to model the data.

We have shown the general form of a univariate polynomial regression model, but to draw some important comparisons, we will write out the general form for a second-order multivariate polynomial regression, with the general model becoming obvious.

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_{11} \cdot x_1^2 + \beta_{22} \cdot x^2 + \beta_{12} \cdot x_1 x_2 \tag{2.5}$$

We can further rewrite this as

$$y = \beta_0 + f_1(x_1) + f_2(x_2) + f_{12}(x_1, x_2) \tag{2.6}$$

such that

$$\begin{aligned} f_1(x_1) &= \beta_1 \cdot x_1 + \beta_{11} \cdot x_1^2 \\ f_2(x_2) &= \beta_2 \cdot x_2 + \beta_{22} \cdot x_2^2 \\ f_{12}(x_1, x_2) &= \beta_{12} \cdot x_1 x_2 \end{aligned} \tag{2.7}$$

It is important here to note that while both f_1 and f_2 represent the expected second-order polynomials that make up the multivariate polynomial regression at hand, we have an emergence of a properly second-order interaction term f_{12} which is novel relative to the first-order case.

Models of general linear and additive nature such as linear and polynomial regression can be generalized and made even more powerful while maintaining quite good interpretability. We have seen that the move from linear to polynomial regression can capture more complex relations behind the data while keeping the method statistically linear in terms of its parameters. We have seen that polynomial regression can be reduced to a multivariate linear regression, just by taking $x_i = x^i$ or even $x_i = f_i$ for all pure and interaction terms in a multivariate polynomial regression. But why stop at polynomials? Surely even the polynomials have their limitations and there exist data that cannot be properly modeled either linearly or polynomialy.

If we observe the general form of (1.6), one might feel compelled to broaden the scope further from polynomials to allow any kind of functions one might compute to fit the data. This is exactly the intuition behind two quite powerful models that despite their good performance retain a significant amount of intrinsic interpretability. The first of these models is obtained by dropping all interaction terms from (1.6) and relaxing the condition that f_i is a polynomial, besides allowing for an arbitrary number of features. Namely one obtains a model described by

$$y = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n) \tag{2.8}$$

Where in the case of classification it is more useful to fit a proxy function of the expected value of y (say its log odds) instead of y itself, which leads to the general form of

$$g(E[y]) = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n) \tag{2.9}$$

where regularization is expressed by the condition that $E[f_i] = 0$ for all i.

Models described by (1.9) are called generalized additive models or GAMs for short since they allow complex data to be fitted by many non-linear functions capturing non-trivial relationships between features and the target response variable. This increment of performance however is not gained at the expense of interpretability. First, notice that every single f_i can be plotted on a graph explicitly and visually describing how changing a single feature corresponds to a change in the target variable. By fully knowing the underlying functions, we can properly understand the model's decisions given the values of its features.

Furthermore, we can use the determined plots of the functions to scan for any odd or unexpected relations, jumps, or strange biases - by doing this we can either detect true underlying relationships that were previously unknown or perhaps more probably detect a certain bias that

was present in the original dataset, which is relevant for example if we intend to train a more advanced model e.g. a deep neural network on the dataset.

It is also worth noting here that once we have a GAM for a certain dataset, concerning interpretability, the model is not just locally interpretable but also globally interpretable as well since we have seen that we have full knowledge of all the functions that make up the GAM, and those functional relations do not change in a case-by-case fashion but are rather global themselves.

One might wonder why we drop interaction terms in (1.6) when advancing to GAMs. The answer is for simplicity, but we can include them. By including the interaction terms in our model, we arrive at an even more powerful model than GAMs which is abbreviated to GA2M to highlight the fact of inclusion of second-order interaction terms. Therefore, our model now has the form of

$$g(E[y]) = \beta_0 + \sum_i f_i(x_i) + \sum_{j \neq i} f_{ij}(x_i, x_j) \tag{2.10}$$

Everything we have hitherto said about GAMs also applies to GA2Ms. GA2Ms are even more powerful since they do not assume that all features attribute to the response without intercorrelation. However, note again that we do not lose interpretability. Just because we have second-order terms doesn't mean we lose out on interpretability. We can plot 2D functions as heatmaps and retain all the transparency of GAMs while obtaining a more powerful model. Everything that was said of GAMs with regards to it being a globally intrinsically interpretable model holds for GA2Ms too.

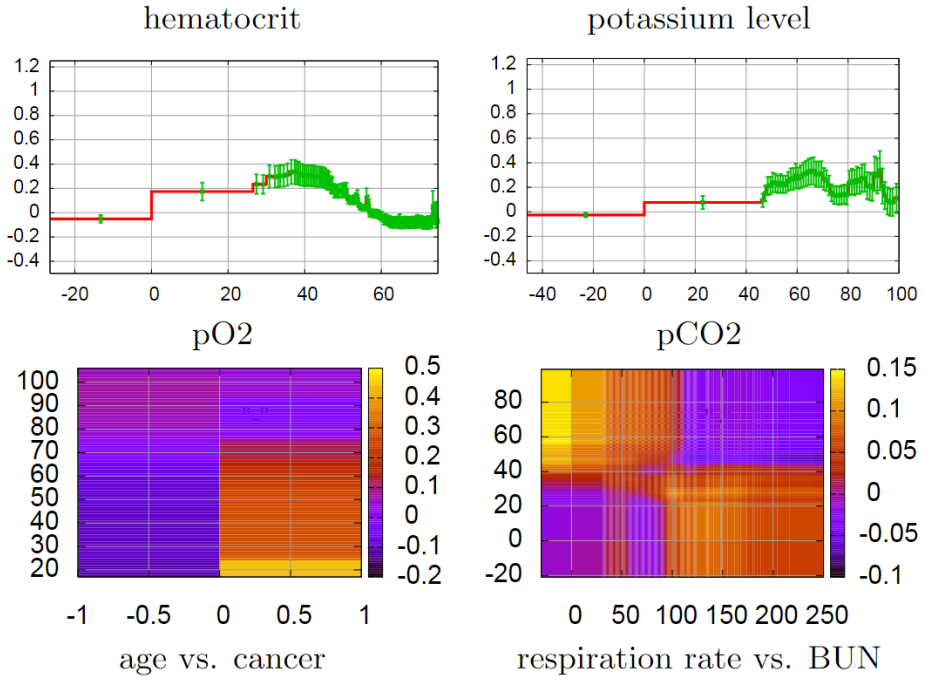

Figure 2.3: A representation of how features can be visualized in GA2Ms [1]

Now technically speaking, but with considerable complication, we can go even further. We can conceive of GA3Ms and consider third-order interaction terms as well. However we will have to incur a loss of interpretability since visualizing 3D functions, while possible by 3D heatmaps, is not nearly as elegant as it is in lower dimensions. Going beyond third-order interaction terms quickly makes further increments of performance (if we are not overfitting that

is) come at a very significant price in interpretability. It is simply not possible to visualize 4D functions without many 3D projections, and even 3D projections themselves are quite awkward for visualization unless we also employ some kind of graphics environment for improved visualization.

GA2Ms have been successfully employed in a healthcare context for two classification tasks - estimating pneumonia risk and estimating the probability of 30-day hospital readmission [1]. Authors achieve superior task performance however, the real selling point is the intrinsic interpretability that accompanies precision, which is very important in a sensitive domain such as healthcare.

Chapter 3

Literature review

The pursuit of transparency and understandability in artificial intelligence systems, particularly in machine and deep learning, marks a growing trend in modern AI research. This necessity stems from the increased deployment of AI in critical decision-making processes, where understanding the automated reasoning behind algorithmic decisions is essential. This domain broadly categorized under explainable AI (XAI), has gained substantial traction in recent years, addressing the opaque nature of some machine learning models and especially deep neural networks [19].

While performance can be quantitatively measured, criteria such as safety and fairness are less quantifiable, making interpretability—an essential feature where a system can explain its reasoning—a necessary fallback. However, there is no consensus on the definition and evaluation of interpretability, typically assessed either through application-based utility or quantifiable proxies like model sparsity. The urgency to define and rigorously evaluate interpretability is underlined especially by regulations requiring algorithms to provide explanations for decisions that significantly affect users.

The ongoing surge in XAI research interest corresponds to significant and dynamic advances and innovation. We will provide a review of some of the most notable recent developments. One notable breakthrough was the introduction of techniques such as Layer-wise Relevance Propagation (LRP) and DeepLIFT, which have provided deeper insights into the contribution of individual neurons to the final decision of a neural network. These techniques can improve the transparency of the automated decision-making process happening inside neural networks [20, 21].

Layer-wise Relevance Propagation (LRP) and DeepLIFT (Deep Learning Important FeaTures) are advanced techniques developed to enhance the interpretability of neural networks by attributing the network's predictions to individual input features. LRP works by assigning relevance scores to neurons, propagating these scores backward from the output layer to the input layer using specific propagation rules, resulting in a heatmap that indicates the contribution of each input feature to the decision. DeepLIFT, on the other hand, compares the activation of neurons to a reference input, calculating the difference and propagating these differences back through the network to determine feature contributions. This proportional attribution helps identify important features relative to a baseline, providing stable and interpretable insights.

Furthermore, the development of model-agnostic interpretability methods — such as Local Interpretable Model-agnostic Explanations (LIME) and SHAP (SHapley Additive exPlanations)—has empowered practitioners to approximate the outputs of complex models with simpler, interpretable explanations. These methods have been instrumental in validating the trustworthiness and fairness of AI systems by highlighting how particular features influence

predictions, thus providing a window into the model's operation that is both accessible and actionable [11, 12, 13]. We will be discussing these two methods at length, along with a slightly different interpretability method called Anchors as well, therefore we postpone giving detailed descriptions of these methods for later in the text.

The integration of visual explanation tools, such as saliency maps and activation atlases, has also been a significant development. These tools visually depict which parts of input data (like pixels in image recognition tasks) most influence the output, providing intuitive and direct insight into the model's perceptual and decision-making processes. This visual approach not only aids developers and researchers in debugging and improving models but also makes the explanations more tangible and understandable to non-experts, enabling broader acceptance and ethical use of AI technologies. All of the methods analyzed in this thesis take advantage and rely on such visual representations of input attribution [22, 23].

Saliency maps identify the most critical regions of an input image by calculating the gradient of the model's output concerning the input pixels. This gradient-based approach produces a heatmap that indicates the areas of the image with the highest influence on the model's predictions. In contrast, activation atlases offer a more extensive visualization by mapping the patterns and features that activate neurons across various layers of the neural network. These atlases provide an aggregate and organized representation of the activations, providing a detailed view of the model's internal feature extraction and decision-making processes.

Saliency maps have been effectively used for visualizing the reasoning behind decisions made by Reinforcement Learning (RL) agents. These maps aim to be human-intelligible by highlighting which pixels the model focuses on when making decisions. In a similar fashion to other modalities, features, and characteristics of the environment are analyzed to gain insight into the learned reasoning of action choice performed by the agent. Pixel-level representations, however, are too granular and difficult to interpret. Instead, the environment is segmented into objects, and then object saliency maps rank those objects in a state based on their impact on the reward. This is achieved by masking each object and calculating the difference in reward, revealing the influence of each object. Positive differences indicate beneficial objects, while negative differences indicate detrimental ones. Object saliency maps provide clearer insights into which objects the model prioritizes in its reasoning and their relative importance.

Moreover, the development of tools such as Google's What-If Tool and IBM's AI Explainability 360 toolkit are significant advances toward embedding interpretability into the lifecycle of AI development and MLOps. The What-If Tool allows users to explore datasets, assess model performance, conduct counterfactual analysis, compare models, and evaluate fairness metrics through interactive visualizations. IBM's AI Explainability 360 offers a suite of explanation methods, providing both global and local explanations, visual and quantitative insights, and tools for detecting and mitigating biases. By bridging the gap between complex data science techniques and practical applications, these innovations are enhancing transparency and developing trust in AI systems [24, 25].

On a more legal and regulatory note, the concept of a "right to explanation" under the GDPR (General Data Protection Regulation) - a European Union regulation concerning information privacy - has faced debate and challenges: it is not explicitly legally binding, applies only to significant automated decisions, is technically difficult to implement, and risks revealing trade secrets. Despite these obstacles, providing explanations remains ethically important. Effective explanations should help individuals understand decisions, contest adverse outcomes, and identify changes needed for favorable results. Counterfactual explanations, for example, which show how different inputs could change the outcome, offer an effective compromise by providing actionable insights without exposing the full internal logic of algorithms [11, 26].

Lastly, the integration of interpretability frameworks into real-world applications has begun to reshape how stakeholders interact with AI systems across various sectors. Financial services, healthcare, and legal fields, where decisions have profound impacts, are increasingly relying on explainable AI to justify and review automated decisions. This push for transparency is crucial for compliance with international regulations such as GDPR, which mandates explanations for automated decisions affecting European Union citizens. As these technologies mature, the focus shifts towards not only understanding AI processes but also refining them to ensure they align with ethical standards and societal values. This evolution towards more accountable AI underscores the critical role of interpretability in building systems that are not only effective but also equitable, just, and aligned with the interests of humanity [27, 28].

Chapter 4

Methods

4.1 LIME

LIME stands for Local Interpretable Model-Agnostic Explanations and is an interpretability method introduced in 2016 by Ribeiro et al. [12]. As the name suggests and as previously mentioned this is an interpretability method that treats the underlying model as a black box and extracts explanations for predictions by analyzing how the model's output changes by perturbing features of the input.

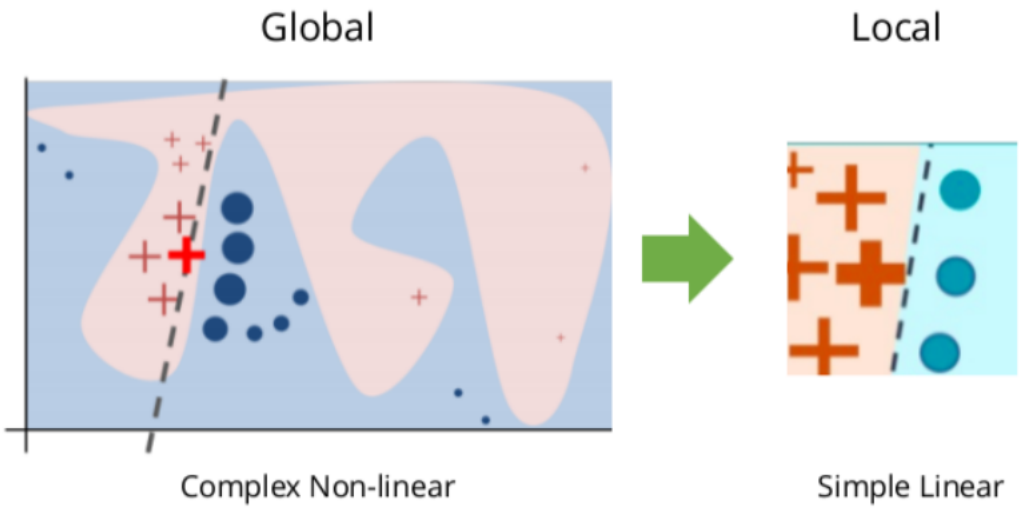

Figure 4.1: LIME approximates the global model linearly in the vicinity of a given instance

LIME is a method that is characterized by its locality in that it provides an explanation for a given input instance and does not provide global model explanations out of the box. It can be, however, extended by utilizing the aforementioned submodular pick algorithm to combine representative local explanations for a global overview of model behavior. The submodular pick approach for attaining global explanations by a local method is also described and introduced in the same paper as LIME itself, and the approach is not necessarily limited to LIME only and can be extended to use other similar local feature attribution interpretability methods as well.

The fundamental idea behind LIME is to first represent the input space of the model in an interpretable fashion (if it is not organically interpretable a priori). The second step then is to gather several perturbed samples around the target instance we wish to explain, weighted by their distance from the original sample defined by some distance metric (displayed on the left

side of Figure 2.1 as the set of pluses and circles, their size reflecting their weight or inverse distance from the original instance which is represented by the bold plus sign). The third step is to generate a new dataset of input-output pairs, based on the previously obtained set of perturbed instances, by calling the underlying model on each element of the set of perturbed instances.

The last step, where we obtain our explanations is to learn a completely new inherently interpretable model (typically a zero-offset linear classifier) on the generated input-output pairs dataset obtained in the previous step. Then, since the learned model is inherently intepretable and as we will see shortly inherently regularized as well, we can simply pick the appropriate number of top features ranked by their respective weights in the resultant model. These extracted features then serve as local interpretable explanations of the original instance's classification.

(a) Husky classified as wolf (b) Explanation

Figure 4.2: An example of LIME explaining an image classification

Notice that, from the provided explanation, it seems that the network makes its decision completely based on the presence of snow in the lower part of the image. Can this really be considered a satisfying prediction?

For the data modality of images in particular, the interpretable representation is taken to be in the form of superpixels (sets of pixels) achieved by segmentation algorithms [29]. Then the perturbations of the original image are represented as boolean vectors, zeros indicating the absence of a given superpixel. The original image is of course represented as a vector of all ones in this representation.

More formally, LIME can be expressed mathematically in the following way.

$$\zeta(x) = \underset{g \in G}{\arg\min}\, L(f,\ g,\ \pi_x) + \Omega(g) \tag{4.1}$$

Where x is the instance whose explanation we seek. Notice the form that is reminiscent of (1.4). In (2.1) L represents the error related to the learned model fidelity concerning the original model we seek to explain. We must minimize that error to have a sensible explanation. On the other hand, we must also minimize Ω which represents the measure of complexity of the learned model. Again, we don't want just any sort of learned model, we need an inherently interpretable learned model, and to achieve that it is important to have a regularizing parameter that will prevent the learned model from becoming opaque itself, thus breaking the concept.

The sum in (2.1) is minimized over the set of all possible interpretable models, but usually for practical purposes, LIME opts for linear models even though other interpretable models such as decision trees could be used as well. The fidelity function L tells us how accurately the interpretable model g approximates f, the target classifier of the explanation, in the π_x space around x, the observed instance.

The exact choice for π_x is an implementation detail, but one example could be a simple Gaussian distance of the form.

$$\pi_x(z) = \exp(-D(x,z)^2/\sigma^2) \tag{4.2}$$

Where D is a distance metric appropriately chosen for the modality of data in question, e.g. a cosine distance between vectors of text embeddings or the Euclidean distance between the pixels of two images. The σ hyperparameter lets us control the density or the narrowness of our proximity function. Furthermore, the full form of the loss function could be given as

$$L(f,\ g,\ \pi_x) = \sum_{z,\ z' \in Z} \pi_x(z)((f(z) - g(z'))^2 \tag{4.3}$$

Where z represents the interpretable map of the original (potentially a priori uninterpretable) instance x and z' represents the interpretable representation of the perturbed datapoint obtained from z. The perturbation itself could be performed simply by randomly setting some of the representation vector's elements to zero which is the standard implementation, however more involved and specifically modified implementations are also possible.

As regards the complexity measure Ω, a perfectly workable choice, since classic implementations of LIME usually rely on learning linear models, would be a function of the form.

$$\Omega(g) = \infty \mathbb{1}[\|w_g\|_0 > K] \tag{4.4}$$

This treats any learned linear model with more than K non-zero features (since we take the L_0 norm of the learned model's weight vector) as having infinite complexity thus rendering them unfeasible, and those with K or less non-zero features as having zero complexity. The user can set an upper limit on what is the acceptable number of features, that is, the acceptable complexity of the learned model itself.

Let us cite the full algorithmic sketch of the LIME method [12]. The prime symbol is used throughout the algorithm to denote interpretable mappings of data points from the original data space. The K-Lasso function called in line 6 is simply a pragmatic approach to practically implement the regularization since the theoretic one given in (2.4) is unsuitable for actual optimization. Therefore, K-Lasso is used, which is just a two-step process. First simple L1 regularization is used on Z to perform feature extraction where the target number of features is K. Then the final step of K-Lasso and LIME itself is to learn a linear model by utilizing these prior computed features. More precisely, we just learn the appropriate weights and in other words, feature importance or prediction sensitivity relative to input features. In conclusion, one could in a sense think of LIME with properly chosen hyperparameters as a sort of gradient of the classifier itself, where the combination of extracted features along with their weights indicate the the direction of greatest prediction change relative to input features.

4.2 SHAP

SHAP is shorthand for Shapley Additive Explanations and is an interpretability method that shares many similarities with LIME. It was introduced in 2017 by Lundberg and Lee [11]. First

Algorithm 1 LIME

Require: Classifier f, Number of samples N
Require: Instance x, and its interpretable version x'
Require: Similarity kernel π_x, Length of explanation K
$Z \leftarrow \emptyset$
for $i = 1$ to N **do**
  $z'_i \leftarrow$ sample_around(x')
  $Z \leftarrow Z \cup \{(z'_i, f(z_i), \pi_x(z_i))\}$
end for
$w \leftarrow$ K-Lasso(Z, K) with z_i as features, $f(z)$ as target
return w

of all, just like LIME, it is a model agnostic and local interpretability method. Another similarity is that SHAP is also a feature attribution method, which means that the provided explanations tend to extract features of input that were relevant for making a given prediction. Lastly, just as LIME can be extended to generate global explanations, SHAP also has methods developed around it to develop global explanations by combining many local ones.

The main difference therefore between SHAP and LIME is in the way explanations themselves are generated. Whereas LIME entails learning a surrogate linear model that is locally faithful to the model under investigation, SHAP utilizes concepts and equations developed in cooperative game theory to calculate feature importance values. Crucially, the concept of Shapley values is of fundamental importance for SHAP.

Shapley values were introduced in 1953 by Lloyd Shapley in the context of cooperative game theory [30]. Cooperative game theory allows for the theoretical treatment of coalitional dynamics emergent from strategic competitive games. The inspiration for developing Shapley values came from analyzing the question of how the rewards and costs associated with engaging in a cooperative game through a coalition of players can be optimally distributed among the members of the coalition itself.

The idea behind Shapley's values is that the reward assigned to any individual player must be at least equal to and preferably greater than the expected reward obtained by that particular player in the situation where the player acts completely alone or joins a competitor coalition, since otherwise there would be no incentive for the player to engage in cooperative game-playing with the observed coalition at all. Then the intuition behind the Shapley value of a player is to compute the average marginal increase of the reward relative to adding that player to any possible coalition. Formally, the Shapley value for a player j is calculated as

$$\phi_j(\text{val}) = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \frac{|S|!(p-|S|-1)!}{p!} \left(\text{val}(S \cup \{j\}) - \text{val}(S)\right) \tag{4.5}$$

Where the summation happens over all possible subsets that is, coalitions of given players, weighted by a combinatorial factor to account for all possible orderings for any particular subset. The Shapley value therefore calculates the fair expected contribution of a particular player for achieving a given game reward.

Translating all of this back into the domain of machine learning interpretability, the idea is to treat the classification as a cooperative game where the reward is the inverse of the difference between the actual model prediction of a perturbed instance and the original instance prediction. In this analogy, features are treated as players of this cooperative game. In the case of images, SHAP also uses an intermediary step of translating raw pixel data into sets of superpixels, just

like LIME. Therefore it is not the pixels themselves that take on the role of players, rather it is the superpixels.

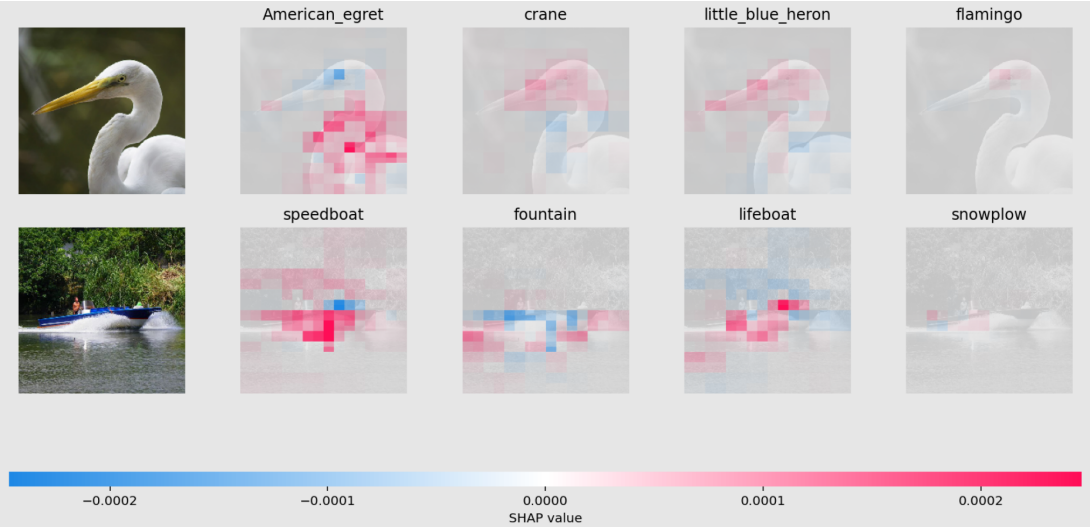

Figure 4.3: An example of SHAP explanations of three top ResNet50 classifications for two images.

Notice that, unlike LIME, the features or superpixels in SHAP are usually chosen to be simple regular squares. Attributions can be positive or negative, just like in LIME.

The specific implementation of SHAP cannot however be based on the full calculation of Shapley values per (2.5) since that would require unreasonable computing power because of the combinatorial explosion of the number of all possible coalitions. However, it is not necessary to do this. It is perfectly possible to approximate Shapley values with a much more reasonable procedure that is practical from a computing standpoint [31]. The specific model-agnostic implementation of SHAP that was proposed is called KernelSHAP. It is noteworthy that there are also model-dependent versions of SHAP that were developed such as TreeSHAP which is specifically suited for tree-based models.

KernelSHAP is strikingly similar to the algorithm of LIME and the similarity is not coincidental. Namely, since Shapley values calculate the expected fair allotment of the prediction class certainty concerning input features, the resultant model is going to be a linear additive one, since simply adding up all the Shapley values by features will amount to the prediction value. Namely, SHAP will result in a model of the form.

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{4.6}$$

where ϕ_j are the approximated Shapley values, and z' is a vector of the interpretable superpixel representation.

The procedure is therefore very similar to LIME. The original instance is translated into an interpretable superpixel representation. After that, various coalitions are sampled, and the underlying model is called upon to generate a labeled dataset of perturbed input-output pairs. For every perturbed instance (a coalition), a weight is assigned in the same manner as in LIME. Then a linear model of the form in (2.6) is learned upon the generated dataset.

The crucial difference however is in the way the instances are weighted before learning the model. Since Shapley values are not derived heuristically, but rather in a rigorous mathemat-

ical fashion for them to possess certain mathematical properties that make them suitable for interpretability applications, and since we want the weights of our learned model (2.6) to adopt exactly these Shapley values (or at least approximate them well enough), it follows that the whole scheme crucially depends on how we define the specifics of the optimization equation in (2.1) since that is the starting point of KernelSHAP itself, and at this point the influence of LIME on it becomes apparent. Now whereas LIME has its philosophy of choosing its kernel and regularization functions, the approach of SHAP is guided by the goal that the linear weights of the resultant model approximate the true Shapley values. Guided by this purpose, the authors of KernelSHAP show that the goal is accomplished by the following choices.

$$L(\hat{f}, g, \pi_x) = \sum_{z' \in Z} \left[\hat{f}\left(h_x(z')\right) - g(z')\right]^2 \pi_x(z') \tag{4.7}$$

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)} \tag{4.8}$$

$$\Omega(g) = 0 \tag{4.9}$$

The loss function in (2.7) is the classic error square that was also used in LIME and is generally used with linear regression models. The function h_x is simply a function that translates the interpretable representation z' back into its original dataspace for the underlying itself then to classify. The crucial choice, however, is of the proximity kernel, which, unlike LIME, doesn't depend on any sense of distance from the original instance. M is the upper limit on coalition size and $|z'|$ is the L_0 norm of the interpretable representation, in other words, the number of present features in the sampled instance. It is shown that this choice of π_x and Ω yields a linear model with approximated Shapley values as weights of the output linear model [11, 31].

Notice however the lack of a regularization term in (2.9). This is a necessary choice that has to be made for this scheme to result in a linear model with Shapley values as weights. However this choice, unlike LIME, can result in non-sparse explanations whereas in LIME it was a point of particular attention that the resultant explanations have to be sparse i.e. interpretable. This fact, along with the possibility that the approach doesn't yield Shapley values of features that vary significantly concerning their attributed importance can be a potential drawback of the method in certain cases. However, one can usually at least up to a point remedy this situation by indirectly controlling the complexity of the provided explanation by setting the M hyperparameter which determines the upper limit on sampled coalition size.

4.3 Anchors

Anchors is an interpretability method introduced in 2018 by the same authors of LIME [13]. Just like LIME and SHAP, Anchors is a model-agnostic method used for generating local explanations. Whereas the idea behind LIME was to locally learn an inherently interpretable surrogate model and use it to generate explanations, and whereas SHAP modified this approach to utilize the attribution idea behind Shapley values, Anchors opted for a different approach to generating explanations.

Namely, instead of learning any surrogate model, the Anchors method tries to express the explanation of a prediction in the form of a simple if-then rule - that is as an implication where the prediction is taken to be the consequent and the target explanation as the antecedent. More specifically, the idea behind the method is to extract the part of the original input that could

be treated as the single reason for triggering the observed prediction, in such a manner that changes in the original input feature that are not however in the extracted explanation subset, do not influence the resultant prediction. Consequently, after finding an explanation by Anchors, by simply inserting the explanation in other unrelated and differently classified instances, one should be able to reliably shift the unrelated classification toward the original prediction class, as exemplified in Fig. 2.4.

Figure 4.4: If the extracted portion is present in the picture - it's a beagle, even if it used to be a stream.

For the data modality of images, as already shown, the rule is expressed as a portion of the image. In other data modalities such as table data, the rule could be and often is expressed in terms of predicates upon values of a subset of features or columns. The Anchors method seeks to improve upon a limitation of local surrogate methods like LIME and SHAP. These two aforementioned methods share a common limitation in that one cannot be sure of the degree of confidence one should assign to the obtained explanation in the vicinity of the original observed instance. Sure, in smooth and well-behaved underlying models all should be well and the local surrogate model should have a relatively high degree of confidence. However if the model exhibits strong non-linearity in the near vicinity of our instance, we are out of luck since it will be impossible to obtain an explanation that holds for neighboring instances as well. This limitation is also related to the choice of kernel width problem which occurs in LIME and which will be touched upon in the next chapter.

Think back to the gradient analogy for LIME which perhaps holds to a lesser extent for SHAP as well. The question of explanation reliability can be then stated as being equivalent to the question of whether the gradient of a function varies significantly in the vicinity of a point in space, or stated in yet another way, how large the vicinity around the datapoint for which we can claim explanation reliability? The Anchors method approaches this problem by introducing the notion of coverage and precision. Precision is the probability that the model outputs the expected prediction class on other instances that satisfy the anchor rule and it is a hyperparameter that can be set before anchor computation. Coverage is defined simply as the proportion of the sample space that satisfies the anchor i.e. contain it. If precision is set too high then the coverage will necessarily be relatively low, so a balance must be struck. Anchors therefore attempt to find a rule, or in our case, a portion of an image, that maximizes its coverage i.e. applies to the largest number of neighboring instances while satisfying the provided precision.

Let us provide a mathematical definition of an anchor. Let us first define D as a distribution that will represent the way the original instance is perturbed to obtain samples. In LIME we used a simple random boolean vector over an interpretable representation of the original instance, so

a similar approach might be applied here as well although it is most certainly not limited to this choice only. Let us define A as a predicate (possibly composite i.e. a set of predicates) that is defined upon the interpretable representation. We will define

$$A(x) = 1 \tag{4.10}$$

to hold true when all predicates of A evaluate true for x. Let us now define $D(\cdot \mid A)$ to represent the conditional distribution of perturbed samples that evaluate A as true in the sense of (2.10). We can now define the anchor we are seeking, mathematically as

$$\mathbb{E}_{D(z|A)}[\mathbb{1}_{f(x)=f(z)}] \geq \tau,\ A(x) = 1. \tag{4.11}$$

The idea behind this definition is that for A to be considered an anchor, we first draw samples x from the conditional perturbation distribution centered around our original instance z, which we denote as $D(z \mid A)$. In other words, we observe all sampled points around z which satisfy the proposed anchor A. Then we insist upon such a set that the expected value of the share of classifications made over it that are equal to the original classification $f(z)$, is greater than or equal to τ which is our precision hyperparameter mentioned before. Of course for every element x of a subset drawn from $D(\cdot \mid A)$ the anchor has to, by definition hold.

The question now remains - how to compute these anchors? Since, as we have seen in other methods as well, the problem is computationally intractable in its original formulation, we need to relax the formulation a bit and leave room for approximation. Instead of computing the precision from (2.11) directly which is the complicated part, we choose to calculate not precision itself but rather the probability that a given anchor possesses the specified precision in the following manner.

$$P(\text{prec}(A) \geq \tau) \geq 1 - \delta \tag{4.12}$$

Since multiple different anchors may satisfy (2.12) we introduce the mentioned concept of coverage and choose the anchor with the maximum coverage. Formally, we define coverage as

$$\text{cov}(A) = \mathbb{E}_{D(z)}[A(z)] \tag{4.13}$$

This is self-explanatory, after which we can state the problem of finding the anchor in optimization terms as

$$\max_{A \text{ s.t. } P(\text{prec}(A) \geq \tau) \geq 1-\delta} \text{cov}(A) \tag{4.14}$$

The computational aspect of practically solving this optimization problem is not trivial and we cannot here provide the full outline. We will however sketch out the general idea behind the approach. There are two versions of the algorithm, the greedy version and an improved beam search version which is the one that is implemented. However since the fundamental idea is outlined in the greedy version, we will briefly describe it here.

The process begins with an empty anchor that applies to every instance. This anchor is then iteratively refined by adding feature predicates to the original anchor thereby generating a series of candidate rules. In each iteration, if any of the refined anchors have a higher estimated precision than the previously held one, the refined anchor replaces the previous one as the current anchor.

This process continues until the anchor meets set precision and coverage criteria, or until it cannot be improved further. This adds an element of computational convergence to the process, unlike LIME and SHAP, which we will see have runtime implications. The non-trivial part

however is the estimation of precision for candidate anchors. The crucial insight is that the problem of precision estimation is modeled as a pure-exploration multi-armed bandit problem.

The multi-armed bandit problem is a decision-making dilemma where a choice must be made between multiple options (or "arms") repeatedly, with each choice providing a different, initially unknown reward, and the goal is to maximize the total reward over time by learning the best options through trial and error. In our case, each candidate anchor acts as an individual "arm", with the task of determining which anchor has the highest precision akin to identifying the most rewarding arm based on repeated trials and evaluations. This is where the Anchor method relies on the domain of reinforcement learning (RL) since the bandit problem itself originates from the domain of RL. The specific algorithm used for estimating the precision of a given candidate anchor by modeling it as the multi-arm bandit problem is called KL-LUBC and was introduced in 2013 by Kaufmann and Kalyanakrishnan [32, 33].

Chapter 5

Comparative analysis

5.1 Qualitative comparison

There are expected commonalities among the observed methods. For one, all of them are model-agnostic which makes them highly versatile and usable, especially for intrinsically opaque models such as deep neural networks. They are all also local methods, providing however possible extensions to generate global explanations by relying on the idea that a representative sample of local explanations can communicate global model behavior to the end user. In addition, SHAP also has model-specific implementations.

The methods do however differ in their details. Whereas LIME learns a sparse surrogate linear model around the instance, SHAP opts for a more theoretically driven approach and develops a linear model, but one that was specifically designed to calculate Shapley values of determined features. The resulting model of SHAP might not necessarily be sparse and relative to LIME might in some circumstances be less clear or ambiguous. At the same time, since the intuition behind Shapley values is not common knowledge for lay users, they might find LIME explanations less confusing and more intuitive especially because LIME enforces explanation sparseness.

On the other hand, compared to Anchors, LIME can provide more nuanced insight into the local behavior of the model and develop a differential perspective or ranking of features by their relevance in driving a change in the predicted class. On the other hand, Anchors is by far the superior method in terms of explanation simplicity, stability, and precision understood from the perspective of a layman end-user. It's not surprising since Anchors were specifically developed to possess exactly those qualities. Therefore one can hypothesize that if there is a premium on the simplicity of our explanations one should pick the Anchors method. This might be the case for example in situations where the end user doesn't require much detail from the explanation but is rather looking for a more broad and general overview.

If however, one is interested in developing more nuanced insight into a prediction, LIME sounds more appropriate since it allows for multiple features while remaining sparse. It is within the purview of the user to control how granularized LIME's explanations should be. In the limiting case, one could simply ask LIME to model the prediction by using a single-feature linear model which is reminiscent of the Anchor approach. However one must note that such usage of LIME will still necessarily suffer from possible limitations of applicability coverage, unlike Anchors where we know that the obtained explanation is certainly the one with its applicability coverage maximized.

There are two most commonly mentioned disadvantages to using LIME. By far, the greatest hurdle is the aforementioned choice for the kernel width. The explanation can vary significantly

as a function of kernel width.

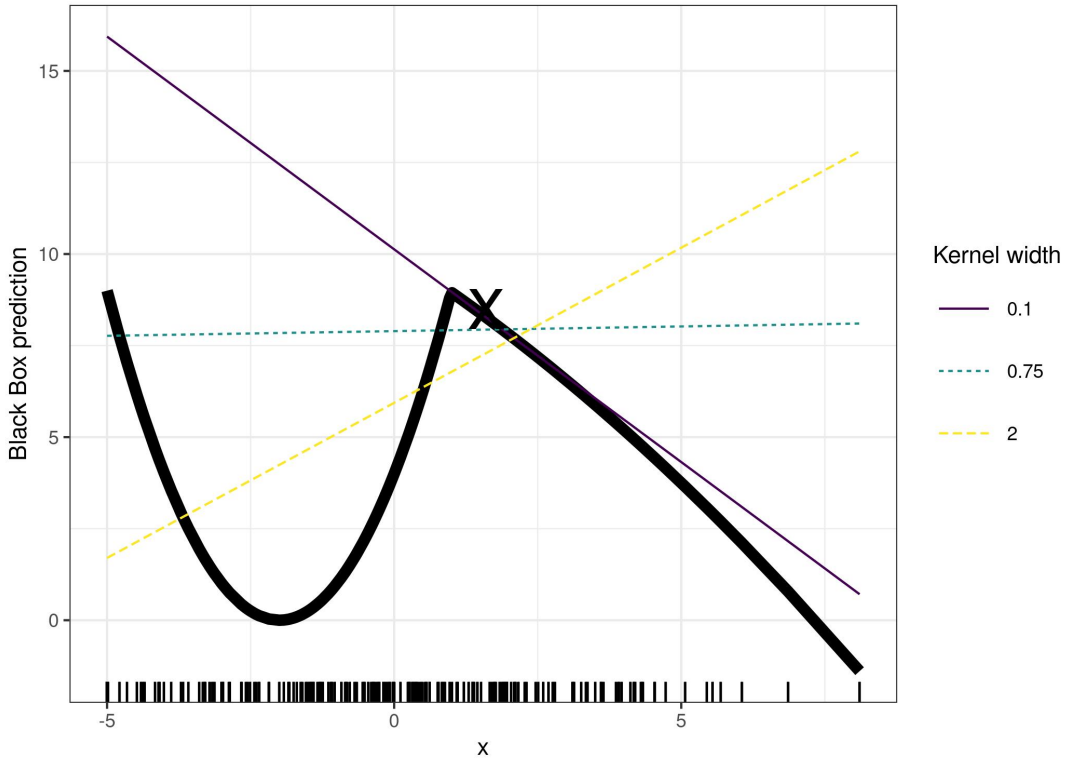

Figure 5.1: LIME might be underwhelming when used near points of great model variability.

The black curve shows the black-box model's predictions, while the purple (kernel width 0.1), cyan (0.75), and yellow (2) lines represent LIME approximations. A narrow kernel (0.1) captures too little context, missing broader trends, while a wide kernel (2) over-generalizes, overlooking important local variations. The intermediate kernel (0.75) provides a more balanced approximation. This highlights the importance of choosing an appropriate kernel width to balance local and broader trends for accurate explanations.

Figure 3.1 demonstrates this phenomenon. Depending on the choice of kernel width, LIME will pick up on and average out different model behaviors. In these situations, it might be preferable to opt for the Anchors method, possibly by explicitly keeping track of the coverage region for any one explanation, and gathering a sort of mosaic of explanations by seeking to cover the relevant region of the function with multiple explanations, similar to the idea of generating global explanations from local ones. The other somewhat smaller issue with using LIME is that since the default kernel is a Gaussian, that can in some circumstances fail to capture the possible inter-correlations of the underlying data, therefore reducing the relevance of the perturbed instance set and decreasing the quality of the resultant explanation.

5.2 Runtime analysis comparison

We performed comparisons of runtime for calculating explanations by relying on methods studied in this thesis. All of them have open-source implementations with accompanying documentation freely accessible on the web. LIME and SHAP have more comparable configurations,

where the computation will, stated roughly, run for as long as the user prefers. The most important function for generating explanations in LIME is the `explain_instance` function which we call from the previously instantiated `LimeImageExplainer` object. The main parameter of the function is the number of samples that are generated before learning a linear surrogate model.

SHAP is similar in its use. Firstly a SHAP explainer is instantiated as a functional object, and then it is called upon the instance we wish to explain along with boilerplate parameters except the `max_evals` parameter which serves the same function to SHAP as the number of samples for LIME. It provides an upper limit on the number of model calls, thereby indirectly on the number of sampled instances around the original datapoint and lastly on the granularity of the explanation itself. Therefore LIME and SHAP are comparable in runtime simply by varying the number of sample parameters.

Anchors is a bit different since it is convergence-based. Therefore the runtime is not necessarily a simple linear function of its parameters and it can vary depending on input alone which is not the case with SHAP and LIME. Similar to other methods, in Anchors, an explainer object is instantiated, and then a `explain` function is called with the appropriate parameters. The convergence nature of the method stems from searching for an anchor that satisfies the target precision with the maximum coverage. Under some circumstances, a nontrivial anchor will not exist. In those cases, the method will run for a very long time until it simply recovers the original instance as the anchor, which is a trivial solution.

We provide data tables that detail and compare the runtime of LIME SHAP and Anchors separately. We display Anchors separately since we consider that the methods are not directly comparable for the reasons outlined above. The models were analyzed in the Google Collab environment utilizing GPUs freely available and served over the cloud. SHAP and LIME were evaluated, of course, on the same image. The results are displayed in seconds, broken down by the method, and the number of samples/evaluations.

	500	1 000	5 000	10 000	15 000
LIME	20	31	155	304	453
SHAP	11	20	104	210	314

Table 5.1: SHAP and LIME runtime compared by runtime in seconds.

We follow up with runtime data for Anchors. We have tested Anchors on three different images, for each running Anchors with three different hyperparameter settings which we have noted as the triple (t, δ, τ). The Anchor precision parameters are denoted as t whereas δ is the desired significance level of the returned anchor having the target precision t and τ is the allowed error tolerance when picking the best anchor while solving the underlying bandit problem.

The data in tabular form follows on the next page.

$(t,\ \delta,\ \tau)$	Image 1	Image 2	Image 3
(0.90, 0.05, 0.15)	35	97	35
(0.95, 0.10, 0.10)	115	255	118
(0.85, 0.01, 0.05)	1	299	1

Table 5.2: Anchors runtime comparison by seconds until convergence.

The short runtimes for the third configuration are an example where the method returns an empty anchor which is also a possibility and a failure mode of the method, along with the method returning a portion of the image comparable in size to the original, thereby showing low discriminatory power in such edge cases.

5.3 User study comparison

Techniques such as LIME, SHAP, and Anchors provide frameworks to explain the decisions of complex models. This research hypothesizes that each interpretability method offers unique advantages based on data characteristics, runtime constraints, and preferred explanation features. By comparing these methods, we aim to identify which best elucidates model decisions, particularly for deep neural networks, to enhance the transparency and accountability of AI systems. The study seeks to determine the optimal conditions for each method's effectiveness, considering end-user characteristics, computational resources, model type, and interpretability needs.

We performed a user study analyzing user satisfaction with explanations generated by our studied explanation methods. We randomly assigned each respondent with a method and then administered the appropriate version of the questionnaire we developed. Users were provided with basic and short explanations of how to interpret the provided explanations. We also gathered some demographic data about the users themselves to test whether there exist any correlations.

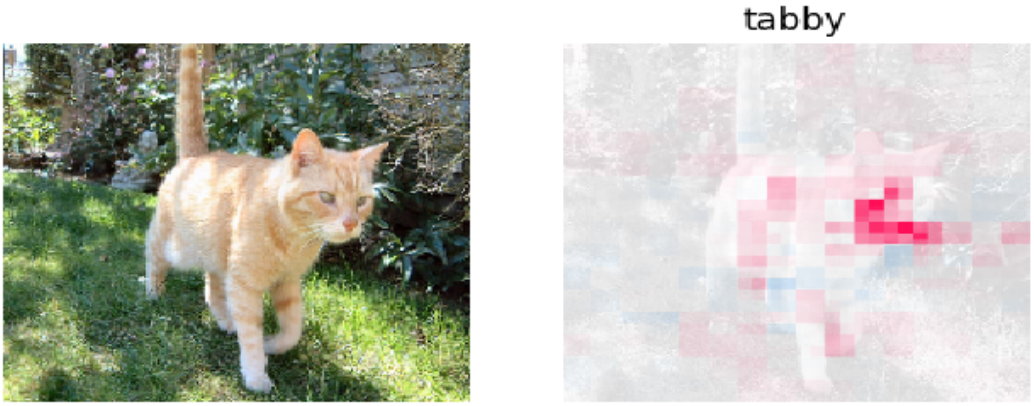

Figure 5.2: Example of SHAP presentation.

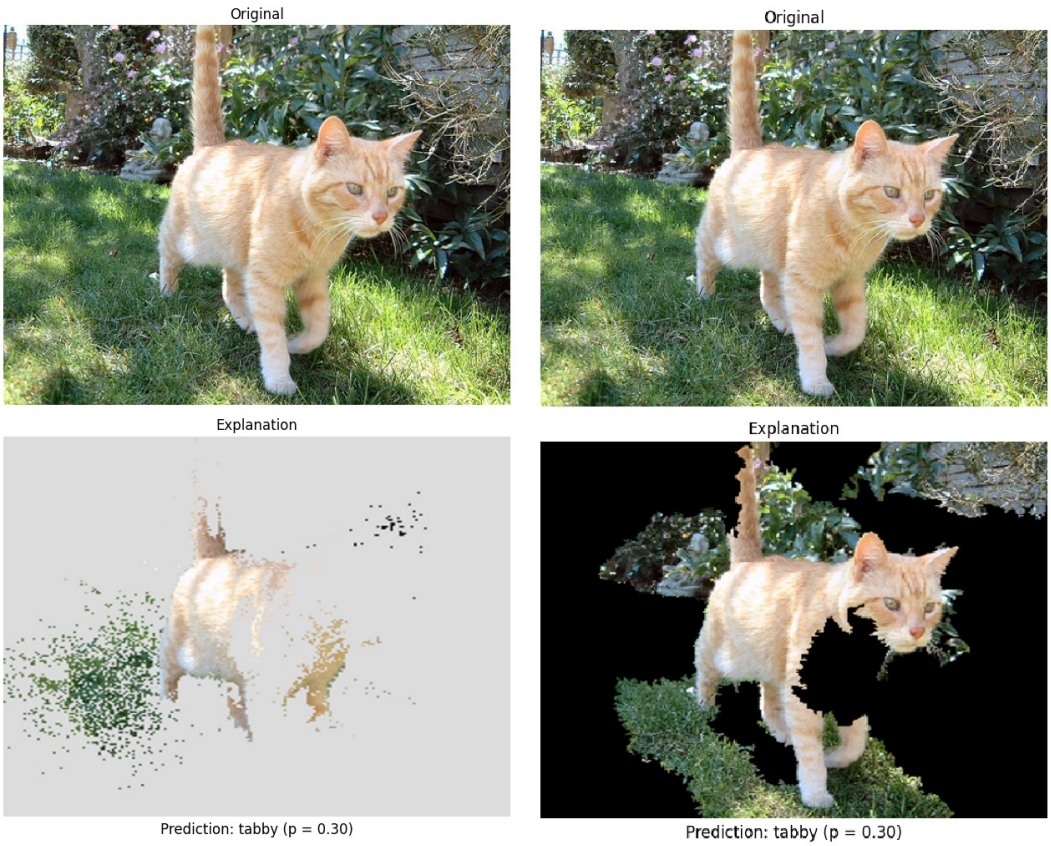

Figure 5.3: Example of the way explanations were presented. LIME is left and Anchors is right.

For SHAP and LIME, we used a sampling rate of 15,000 samples or model calls, utilizing a surrogate linear model with 15 features for LIME. We used Anchors with default parameters with a precision threshold of 95%. User explanation satisfaction itself was measured by administering the Hoffman Explanation Satisfaction Scale [18]. We gathered a total of 92 responses, 32 for LIME and 30 for SHAP and Anchors each.

With regards to respondent demographics, respondent age ranged from 19 to 28 years. Respondents were most often 23-year-olds, with those under 20 and over 25 accounting for a smaller portion of the sample. The gender split was fifty-fifty and with regards to education level, 80% of respondents were undergraduates or held a bachelor's degree whereas the rest were masters students. Among the respondents, 35% reported that they had previous experience with explainable artificial intelligence. None reported color blindness that might have interfered with interpreting explanations. No correlations have been found between measured demographic data and explanation satisfaction levels.

We will show the survey response data for each method and item of the questionnaire in a table form by assigning a total response score as a simple weighted average of all responses where responses from the Likert scale were directly coded as values from $1-5$, with 1 indicating strong disagreement and 5 strong agreement. We will then analyze the results in more detail by each method. We will now provide the question legend.

Q1: From the explanations, I understand how the neural network performed its classifications.

Q2: The provided explanations of the network's classification are satisfying.

Q3: The provided explanations of the network's classification have sufficient detail.

Q4: The provided explanations of the network's classification seem complete.

Q5: The provided explanations of the network's classification tell me how to use it.

Q6: The provided explanations of the network's classification are useful to my goals.

Q7: The provided explanations show me how accurate the network's classification is.

Q8: The provided explanations let me judge when I should trust and not trust the network's classifications.

The results are summarized in the following table.

Question	LIME	SHAP	Anchors
Q1	3.6	3.6	3.3
Q2	3.0	3.4	2.9
Q3	2.6	3.1	2.9
Q4	2.5	2.9	2.8
Q5	2.6	2.6	2.6
Q6	2.7	2.8	2.9
Q7	3.7	3.6	3.5
Q8	3.4	3.4	3.4

Table 5.3: Averages of the responses obtained in the survey.

We can quickly glean that the layman user prefers SHAP explanations in the majority of situations. LIME and Anchors are comparable in their performance, however both are outperformed by SHAP, even if by a small margin. Breaking the results down by questions, both LIME and SHAP received the same score on Q1, indicating that participants found these methods equally effective in helping them understand how the neural network performed its classifications, whereas Anchors scored slightly lower.

Performance in Q2 and Q3 indicates that SHAP's explanations were more satisfying for users compared to LIME and Anchors as well as perceived as more detailed, LIME is comparable to Anchors on satisfaction but behind on detail. Scores for Q4 show that with regards to explanation completeness, SHAP again takes the lead, with Anchors outperforming LIME as well. On Q5, the question of model usage, there are no differences between the models. Truly, explanations as generated by the observed methods do not tell the user anything about the usage of the underlying network itself.

Answers to Q6 indicate that Anchors scored slightly ahead of SHAP and LIME, suggesting that Anchors' explanations were perceived as slightly more useful for the users' goals. This might stem from the fact that Anchors is perceived as the most straightforward method for layman users. With regards to accuracy, the question Q7, LIME scored highest, closely followed by SHAP and Anchors. We believe this indicates that participants felt LIME and SHAP were more effective in showing how accurate the network's classifications were due to the possibility

of Anchors generating incomplete (too narrow) explanations for some instances. LIME has a slight lead relative to SHAP on accuracy probably because LIME has better feature selection compared to SHAP.

Lastly, all methods received equal scores on Q8, indicating that participants found all methods equally effective in helping them judge when to trust or not trust the network's classifications. We believe this is because all of our methods were analyzed for local explanations only, whereas to get a more nuanced view of the underlying model's failure modes, global explanations are probably necessary and preferred.

In conclusion, SHAP generally received higher ratings for satisfaction, detail, and completeness of explanations, suggesting a slight preference for this method among participants. LIME and SHAP were equally effective in helping users understand the classifications and judge the accuracy and trustworthiness of the network's outputs. Anchors, while slightly lagging in some areas, were perceived as most useful for participants' goals. Overall, these results indicate that while SHAP is preferred for its detailed and satisfying explanations, LIME and Anchors also have strong points in accuracy and goal alignment, respectively.

Chapter 6

Conclusion

In this thesis, we conducted a comparative analysis of three prominent interpretability methods—LIME, SHAP, and Anchors—to determine which method provides the most effective, intuitive, and detailed explanations for model decisions, particularly for deep neural networks. Our analysis was designed to evaluate these methods under varying conditions, emphasizing their application in settings where explanations are served to layman end users without prior domain or AI expertise.

During this research, we thoroughly explored each interpretability method's particularities, advantages, and limitations. For instance, we analyzed LIME's variability in explanations due to kernel width adjustments and examined Anchors' tendency to yield overly simplistic explanations, emphasizing the necessity for careful parameter tuning and an understanding of each method's theoretical foundations and how their theoretical underpinnings entail their optimal use cases. Similarly, our work with SHAP involved delving into the nuances of Shapley values, which is essential for understanding the method's approach in providing detailed and satisfactory explanations. The GitHub repository that contains code used to generate explanations and test the methods can be found at `https://github.com/arascic1/ML-Interpretability`.

Our experimental setups, encompassing qualitative comparisons, runtime analyses, and user studies, were designed to help us discover the strengths and weaknesses of each method. The user study, in particular, was crucial in gauging the practical utility of the explanations as perceived by end-users, providing valuable insights into user preferences and the effectiveness of each method in real-world scenarios.

The findings from this study suggest that while no single method universally outperforms the others, SHAP generally offers a better balance of detail and user satisfaction, making it slightly more preferred among the participants. However, LIME and Anchors still demonstrate significant value, particularly in scenarios where specific characteristics of explanations are prioritized.

In conclusion, this thesis not only advances our understanding of model interpretability methods but also sets the stage for future innovations in making AI systems more transparent and accountable. The insights gained here contribute to ongoing developments in the field of XAI and the development of more user-friendly explainable AI technologies.

6.1 Future work

One significant issue of LIME that can arise in settings where explanation stability is very important, such as healthcare, is that due to the inherent stochasticity of the sampling step,

LIME can generate different explanations for the same instance. An attempt to address this has been made by trying to ensure that LIME behaves deterministically in an extension titled DLIME [34]. DLIME relies on two key techniques: Agglomerative Hierarchical Clustering (AHC) and K-nearest neighbor (KNN). AHC clusters the training data, initially by treating each data point as its cluster and then gradually merging the nearest clusters, creating a tree-like structure. When DLIME needs to explain a new instance, it uses KNN to find the cluster from the AHC that is most relevant to this new instance. Once the relevant cluster is identified, LIME proceeds as usual by training a linear model. This method ensures that the explanations behave in a more deterministic fashion. There is ongoing research and development on the issue of LIME explanation stability, whether arising from its inherent sampling stochasticity or kernel width.

A use case for SHAP, and possibly other interpretability methods as well, that is worth mentioning is the application of interpretability algorithms for filtering for outlier data and phenomena. There are neural network architectures called autoencoders that can be used to learn an efficient, dimensionally reduced representation of a dataset. When new data is introduced, the autoencoder attempts to reconstruct it. If the data is similar to what the autoencoder has seen during training (normal data), the reconstruction error will be low. However, if the data is anomalous and deviates significantly from the training data, the reconstruction error will be high. By setting a threshold for this reconstruction error, instances with errors above the threshold are flagged as anomalies.

Autoencoders identify the top-k outliers by scoring each instance, however, manually validating these results can be difficult without explanations. The original model agnostic formulation of SHAP, Kernel SHAP was successfully used in such a setting to generate explanations for manual validation of candidate anomalies detected by autoencoders [35]. It is worthwhile mentioning however that the end users for this approach were domain experts and not necessarily laymen.

There are more serious outstanding issues with post hoc model agnostic methods, however. It can be demonstrated that post hoc explanation methods such as those discussed in this thesis can be unreliable [36]. Namely, it has been shown that it is possible to construct highly biased models, but that under inspection by black-box interpretability methods can in effect conceal their biases and appear inconspicuous when examined through generated explanations. These are open issues with the security and robustness of explanation methods. It is unclear though whether this has hitherto raised significant security issues of relevance in actual use cases.

While all of the studied methods in this thesis are truly model agnostic, they are not necessarily data modality agnostic as well, meaning that depending on what type of data they are used on, their performance may differ. It is perfectly possible that for images for example LIME and Anchors prove superior by their relative performance, while for i.e. tabular data SHAP and Anchors find themselves more suitable. The particular choice of which method to use will necessarily depend on the preferred characteristics of the explanation, who the consumers of explanations are, the data modality, local stability, and the behavior of the underlying model, etc.

Bibliography

[1] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., Elhadad, N., "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission", Proceedings of the 21st ACM SIGKDD international conference on knowledge discovery and data mining, 2015, str. 1721–1730.

[2] Böhm, C., Jacopini, G., "Flow diagrams, turing machines, and languages with only two formation rules", Communications of the ACM, Vol. 9, No. 5, 1966, str. 366–371.

[3] Mahesh, B., "Machine learning algorithms-a review", International Journal of Science and Research (IJSR).[Internet], Vol. 9, No. 1, 2020, str. 381–386.

[4] Rosenblatt, F., "The perceptron: a probabilistic model for information storage and organization in the brain.", Psychological review, Vol. 65, No. 6, 1958, str. 386.

[5] Cybenko, G., "Approximation by superpositions of a sigmoidal function", Mathematics of control, signals and systems, Vol. 2, No. 4, 1989, str. 303–314.

[6] Siegelmann, H. T., Sontag, E. D., "On the computational power of neural nets", Proceedings of the fifth annual workshop on Computational learning theory, 1992, str. 440–449.

[7] Krizhevsky, A., Sutskever, I., Hinton, G. E., "Imagenet classification with deep convolutional neural networks", Advances in neural information processing systems, Vol. 25, 2012.

[8] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S. *et al.*, "Gpt-4 technical report", arXiv preprint arXiv:2303.08774, 2023.

[9] Liesenfeld, A., Lopez, A., Dingemanse, M., "Opening up chatgpt: Tracking openness, transparency, and accountability in instruction-tuned text generators", Proceedings of the 5th international conference on conversational user interfaces, 2023, str. 1–6.

[10] Bostrom, N., "How long before superintelligence", International Journal of Futures Studies, Vol. 2, No. 1, 1998, str. 1–9.

[11] Lundberg, S. M., Lee, S.-I., "A unified approach to interpreting model predictions", Advances in neural information processing systems, Vol. 30, 2017.

[12] Ribeiro, M. T., Singh, S., Guestrin, C., "" why should i trust you?" explaining the predictions of any classifier", Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, str. 1135–1144.

[13] Ribeiro, M. T., Singh, S., Guestrin, C., "Anchors: High-precision model-agnostic explanations", Proceedings of the AAAI conference on artificial intelligence, Vol. 32, No. 1, 2018.

[14] He, K., Zhang, X., Ren, S., Sun, J., "Deep residual learning for image recognition", Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, str. 770–778.

[15] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., "Imagenet: A large-scale hierarchical image database", 2009, str. 248–255.

[16] Pham, H., Dai, Z., Xie, Q., Le, Q. V., "Meta pseudo labels", Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, str. 11 557–11 568.

[17] Srivastava, S., Sharma, G., "Omnivec: Learning robust representations with cross modal sharing", Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2024, str. 1236–1248.

[18] Hoffman, R. R., Mueller, S. T., Klein, G., Litman, J., "Metrics for explainable ai: Challenges and prospects", arXiv preprint arXiv:1812.04608, 2018.

[19] Doshi-Velez, F., Kim, B., "Towards a rigorous science of interpretable machine learning", arXiv preprint arXiv:1702.08608, 2017.

[20] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., Samek, W., "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation", PloS one, Vol. 10, No. 7, 2015, str. e0130140.

[21] Shrikumar, A., Greenside, P., Kundaje, A., "Learning important features through propagating activation differences", International conference on machine learning, 2017, str. 3145–3153.

[22] Carter, S., Armstrong, Z., Schubert, L., Johnson, I., Olah, C., "Activation atlas", Distill, Vol. 4, No. 3, 2019, str. e15.

[23] Simonyan, K., Vedaldi, A., Zisserman, A., "Deep inside convolutional networks: Visualising image classification models and saliency maps", arXiv preprint arXiv:1312.6034, 2013.

[24] Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F., Wilson, J., "The what-if tool: Interactive probing of machine learning models", IEEE transactions on visualization and computer graphics, Vol. 26, No. 1, 2019, str. 56–65.

[25] Bellamy, R. K., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilović, A. *et al.*, "Ai fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias", IBM Journal of Research and Development, Vol. 63, No. 4/5, 2019, str. 4–1.

[26] Wachter, S., Mittelstadt, B., Russell, C., "Counterfactual explanations without opening the black box: Automated decisions and the gdpr", Harv. JL & Tech., Vol. 31, 2017, str. 841.

[27] Jobin, A., Ienca, M., Vayena, E., "The global landscape of ai ethics guidelines", Nature machine intelligence, Vol. 1, No. 9, 2019, str. 389–399.

[28] Goodman, B., Flaxman, S., "European union regulations on algorithmic decision-making and a "right to explanation"", AI magazine, Vol. 38, No. 3, 2017, str. 50–57.

[29] Stutz, D., Hermans, A., Leibe, B., "Superpixels: An evaluation of the state-of-the-art", Computer Vision and Image Understanding, Vol. 166, 2018, str. 1–27.

[30] Shapley, L. S. *et al.*, "A value for n-person games", 1953.

[31] Molnar, C., Interpretable machine learning. Lulu. com, 2020.

[32] Kaufmann, E., Kalyanakrishnan, S., "Information complexity in bandit subset selection", Conference on Learning Theory, 2013, str. 228–251.

[33] Cover, T. M., Thomas, J. A. *et al.*, "Entropy, relative entropy and mutual information", Elements of information theory, Vol. 2, No. 1, 1991, str. 12–13.

[34] Zafar, M. R., Khan, N., "Deterministic local interpretable model-agnostic explanations for stable explainability", Machine Learning and Knowledge Extraction, Vol. 3, No. 3, 2021, str. 525–541.

[35] Antwarg, L., Miller, R. M., Shapira, B., Rokach, L., "Explaining anomalies detected by autoencoders using shap", arXiv preprint arXiv:1903.02407, 2019.

[36] Slack, D., Hilgard, S., Jia, E., Singh, S., Lakkaraju, H., "Fooling lime and shap: Adversarial attacks on post hoc explanation methods", Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, 2020, str. 180–186.