Enhancing And Evaluating Interpretability In Machine Learning Through Theory And Practice

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

am Fachbereich Informatik und Mathematik

der Freie Universität Berlin

vorgelegt von

Leon Sixt

Berlin, 2023

Erstgutachter: Prof. Dr. Tim Landgraf
Zweitgutachter: Prof. Dr. Oisin Mac Aodha

Tag der Disputation: 16.02.2024

Contents

Abstract viii

Publication Of The Thesis ix

Declaration Of Authorship xi

1 Introduction 1
- 1.1 Why Should We Explain Deep Neural Networks? 1
- 1.2 Challenges of Explainable AI 3
- 1.3 Contributions 4
- 1.4 Structure of the Thesis 6
- 1.5 Contributions To Individual Publications 7

2 History Of The Field 9

3 Restricting the Flow: Information Bottlenecks for Attribution 13

4 When Explanations Lie: Why Many Modified BP Attributions Fail 33

5 A Rigorous Study Of The Deep Taylor Decomposition 55

6 Do Users Benefit From Interpretable Vision? A User Study, Baseline, And Dataset 79

7 DNNR: Differential Nearest Neighbor Regression 105

8 Conclusion 129
- 8.1 Discussion 129
- 8.2 Future Prospects 130

Bibliography 131

This solution of the problem of induction gives rise to a new theory of the method of science, to an analysis of the critical method, the method of trial and error: the method of proposing bold hypotheses, and exposing them to the severest criticism, in order to detect where we have erred.

— Karl Popper

Acknowledgements

First, I want to thank my supervisor, Prof. Dr. Tim Landgraf, for his encouragement and dedicated support throughout my Ph.D. studies.

I had the pleasure of collaborating with great people during my studies. In particular, I would like to thank Karl Schulz, Maximilian Granz, Martin Schüßler, Oana-Iuliana Popescu, and Youssef Nader.

During my Ph.D., I had the opportunity to apply my research during two internships at Google Research and Angsa Robotics. I thank my supervisors Dr. Been Kim, Dr. Martin Maas, and Lukas Wießmeier.

I would like to thank the Elsa Neumann Scholarship issued by the state of Berlin for their financial support and the Freie Universität Berlin for providing me with free access to their HPC cluster.

Finally, I am grateful to my partner, family, and friends for their support and encouragement.

Abstract

The field of Explainable AI (XAI) aims to explain the decisions made by machine learning models. Recently, there have been calls for more rigorous and theoretically grounded approaches to explainability. In my thesis, I respond to this call by investigating the properties of explainability methods theoretically and empirically. As an introduction, I provide a brief overview of the history of XAI. The first contribution is a novel theoretically motivated attribution method that estimates the importance of each input feature in bits per pixel, an absolute frame of reference. The method is evaluated against 11 baselines in several benchmarks. In my next publication, the limitations of modified backward propagation methods are examined. It is found that many methods fail the weight-randomization sanity check, and the reasons for these failures are analyzed in detail. In a follow-up publication, the limitations of one particular method, Deep Taylor Decomposition (DTD), are further analyzed. DTD has been cited as the theoretical basis for many other attribution methods. However, it is found to be either under-constrained or reduced to the simpler gradient $\times$ input method. In the next contribution, a user study design is presented to evaluate the helpfulness of XAI to users in practice. In the user study, only a partial improvement is observed when users are working with explanation methods compared to a baseline method. As a final contribution, a simple and explainable model of the k-nearest neighbors regression is proposed. We integrate higher-order information into this classical model and show that it outperforms the classical k-nearest neighbors, while still maintaining much of the simplicity and explainability of the original model. In conclusion, this thesis contributes to the field of XAI by exploring explainability methods, identifying their limitations, and suggesting novel, theoretically motivated approaches. My work seeks to improve the performance and interpretability of explainable models toward more transparent, reliable, and comprehensible machine learning systems.

My dissertation is based on the following publications:

K. Schulz, L. Sixt, F. Tombari, and T. Landgraf (2020). "Restricting the Flow: Information Bottlenecks for Attribution". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=HkxJzAEtvS

L. Sixt, M. Granz, and T. Landgraf (2020). "When Explanations Lie: Why Many Modified BP Attributions Fail". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 9046–9057. URL: https://proceedings.mlr.press/v119/sixt20a.html

L. Sixt and T. Landgraf (2022a). "A Rigorous Study Of The Deep Taylor Decomposition". In: *Transactions on Machine Learning Research*. URL: https://openreview.net/forum?id=Y4mgmw9OgV

L. Sixt, M. Schuessler, O.-I. Popescu, P. Weiß, and T. Landgraf (2022b). "Do Users Benefit From Interpretable Vision? A User Study, Baseline, And Dataset". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=v6s3HVjPerv

Y. Nader, L. Sixt, and T. Landgraf (2022). "DNNR: Differential Nearest Neighbors Regression". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 16296–16317. URL: https://proceedings.mlr.press/v162/nader22a.html

Declaration Of Authorship

Last Name: Sixt
First Name: Leon

I declare to the Freie Universität Berlin that I have completed the submitted dissertation independently and without the use of sources and aids other than those indicated. The present thesis is free of plagiarism. I have marked as such all statements that are taken literally or in content from other writings. This dissertation has not been submitted in the same or similar form in any previous doctoral procedure.

I agree to have my thesis examined by a plagiarism examination software.

_____ _____

Date Signature

1 Introduction

1.1 Why Should We Explain Deep Neural Networks?

The topic of my dissertation is explaining the decisions made by Machine Learning models. Different types of explanations exist for Machine Learning models. For example, a model can be explained by presenting the most important features that led to a specific decision. Or, the model can be explained by presenting a counterfactual example, which is a similar input that would have led to a different decision. Or even, a simple textual summary of the model's behavior could be considered an explanation. In the explainability literature, no common definition of "an explanation" exists. In my dissertation, I will consider anything that is created to help a human understand the model's behavior as an explanation.

Explanations are relevant for various applications: One such application is explaining decisions to end-users, such as providing detailed reports to physicians about automatic medical diagnoses. In the case, that an AI model identifies a tumor as malignant, specific features like irregular shape or increased density that led to the classification could be presented, allowing the physician to better understand and trust the model's output (Hauser et al. 2022).

Another application is using explanations to debug models. Machine Learning Engineers can detect bugs or biases in the model by examining explanations. For example, if a loan approval model disproportionately rejects applicants from a particular minority group, an explanation could reveal that the model places too much weight on an irrelevant feature like the applicant's postal code. This insight enables the engineer to identify and address the issue.

When applying Machine Learning to scientific questions, Explainable AI plays an important role in unveiling the underlying relationships and mechanisms governing complex phenomena (Roscher et al. 2020). Researchers in fields such as physics, chemistry, biology, and climate science often leverage machine learning models to analyze large-scale data and identify patterns or make predictions. However, the inherently opaque nature of these models can hinder the scientific community's ability to build upon these findings or develop

a deeper understanding of the principles at play. Explainable AI bridges this gap by demystifying the decision-making processes of these models, enabling scientists to gain valuable insights into the interdependencies and interactions between variables, and validate the models against established scientific theories.

Lastly, explanations are crucial in complying with legislation that requires transparency and fairness such as the AI act (European Commission 2021). Detailed explanations of the factors contributing to each decision enable institutions, such as financial organizations using AI models for credit scoring, to demonstrate compliance with anti-discrimination laws and regulations. By justifying credit risk assessments and ensuring the model does not discriminate against protected groups, the institution can operate within the frameworks set by relevant legislation and regulations.

Given the various ways in which explanations are applied, there are numerous approaches to create explanations. For vision tasks, the most common approach is to use saliency maps (Simonyan et al. 2013). They highlight the input features most relevant to the network's decision. For images, this can be visualized using a heatmap, where the color intensity indicates the importance of the corresponding pixel. Other approaches include prototypical explanations (Chen et al. 2019a), textual explanations (Kim et al. 2018b), distilling simpler models (Zhang et al. 2021a) and concept-based explanations (Kim et al. 2018a; Ghorbani et al. 2019).

The importance of explanations can also be seen when looking at a possible alternative: summary statistics such as accuracy, AUC, or F1 score. When computed on a held-out test set, they provide a summary of the model's capabilities, e.g. how well it generalizes to unseen data. These metrics can also be seen as an explanation of the model's capabilities. Especially, when curating custom test sets, also other properties than generalization accuracy can be measured. For example, by creating two test sets, one containing only samples of men and the other only women, it can be determined whether the model is equally accurate for both. While these summary statistics provide crucial information about the model's capabilities, they may be inadequate when it is impossible to collect a representative test set, such as when the model is used in a production environment with changing data distribution. Furthermore, metrics computed over test sets do not provide any information regarding individual data points. For example, regulations might require to demonstrate that for each prediction, certain prohibited features did not influence the decision.

1.2 Challenges of Explainable AI

At first, it may seem that the decisions of deep neural networks could be explained easily; after all, neural networks are fully observable systems whose inputs and internal parameters are all known, and they operate according to a well-defined algorithm. However, modern neural networks consist of millions or even billions of parameters and are capable of learning highly complex functions. Thus, providing an algorithm and the network's parameters might enable a computer to reproduce the prediction. Even, for a human, it would not be a helpful explanation but rather a disordered mess.

Explanation methods must therefore break down these complex models to deliver concise explanations. Necessarily, this means that explanations must simplify the model's decision process; otherwise, the explanation would be as complex as the model itself and consequently not understandable to the end-user. One example of such a simplification is focusing on the local neighborhood surrounding an input sample. In addition, approximations such as linearization are often used, e.g., gradient-based methods (Simonyan et al. 2013; Sundararajan et al. 2017), or assumptions such as the independence of the input features, are made (see (Kumar et al. 2020) for a discussion on the approximations of Shapley values). Unfortunately, these simplifications and assumptions are often introduced without transparency, and the differences between the original model and the simplified explanation are seldom quantified.

A negative example is the widely used integrated gradients method (Sundararajan et al. 2017). The integrated gradient method was built on results for path methods (Friedman 2004) from the cost-sharing literature. The question of cost-sharing arises in cooperative game theory, which deals with the fair distribution of costs among multiple agents. In the context of integrated gradients, the authors (Sundararajan et al. 2017) argue that the individual input dimensions (e.g. pixels) can be considered agents, and the model's prediction is the cost that is shared among them. The path method from (Friedman 2004) then provides a mechanism to assign attribution of to the individual dimensions. However, (Sundararajan et al. 2017) did not discuss the assumptions made in (Friedman 2004) and whether they are valid for neural networks too. In fact, the recent work (Lundstrom et al. 2022) shows that neural networks are not contained in the function class discussed in (Friedman 2004). Thus, the claimed theoretical justification of integrated gradients does not hold for neural networks in general.

Contrary to the integrated gradients method, the publication (Chen et al. 2019b) provides a theoretical guarantee on the approximation error of the Shap-

ley value (Shapley 1951). Shapley values also have their origins in cooperative game theory (Shapley 1951) and have been adapted for explainable AI to attribute the contribution of input features to a model's prediction. Despite their desirable properties (e.g., efficiency, symmetry, linearity, null player), computing exact Shapley values can be computationally expensive, as it requires computing the model's prediction for an enormous number of permutations of the input features. In the work (Chen et al. 2019b), the authors provide a new approximation method for Shapley values and also provide a theoretical guarantee on the approximation error.

The issue of unclear assumptions and approximations becomes even more complex due to the challenges in evaluating explanation methods. If we do not understand the model, how can we evaluate the correctness of the explanations? Moreover, no ground truth about a "correct" explanation exists. First, an erratic-looking explanation might correctly explain a faulty model. Furthermore, there might not be a single "correct" explanation, as different applications might justify different assumptions.

Different proxy tasks have been proposed to address this problem. For example, (Samek et al. 2016) introduced the so-called MoRF / LeRF task, which masks the input based on the relative importance of the features. Another task is the bounding box task, which measures how many of the highest-scored input features are contained in ground-truth bounding boxes (Schulz et al. 2020).

Another approach is the sanity checks proposed in (Adebayo et al. 2018). For example, the authors propose to randomize the network's weights and measure the effect on the resulting explanations – surprisingly, the explanations of one method, GuidedBP (Springenberg et al. 2014), did not change.

In summary, explainable AI methods must make some simplifications and assumptions to create understandable explanations. However, these simplifications and assumptions are often not discussed transparently. Furthermore, the lack of hard ground truth makes it difficult to evaluate the correctness of the explanations. While a set of proxy tasks and sanity checks can be used to evaluate the explanations, each evaluation also has its own blind spots.

1.3 Contributions

My contributions address the previously outlined challenges in explainable AI, aiming to foster a more rigorous and transparent field.

Throughout my work, I have focused on several subtopics of explainability, including the development and technical evaluation of a theoretically grounded

saliency method (Schulz et al. 2020), the empirical and theoretical evaluation of modified backpropagation methods (Sixt et al. 2020; Sixt et al. 2022a), a human subject study (Sixt et al. 2022b), and the development of a simple and interpretable model (Nader et al. 2022).

(Schulz et al. 2020) introduced a new explanation method, which is based on the information bottleneck theory (Tishby et al. 2000). The main advantage over existing methods is that the amount of information used can be quantified in bits per pixel. In this publication, we conducted a comprehensive evaluation of saliency maps involving 10 methods, demonstrating that our new method consistently outperforms the others.

In (Sixt et al. 2020), we applied the weight-randomization sanity check (Adebayo et al. 2018) to so-called modified backpropagation methods and found that many of them do not pass it. The sanity check tests whether the explanations change when the network's parameters are randomized. If this is not the case, the explanation method can not explain the model's prediction. In our publication, we also investigate why the methods fail the sanity check to better understand which assumptions or simplifications are responsible for the failure.

In a follow-up publication (Sixt et al. 2022a), we analyzed the Deep Taylor Decomposition (DTD) (Montavon et al. 2017), which is cited as the theoretical basis of several modified backpropagation methods (Kindermans et al. 2018a; Holzinger et al. 2022). We found that the so-called train-free approximation leads to a violation of the assumptions of the Taylor theorem. Furthermore, the theory of DTD either degrades to the gradient $\times$ input method or is underconstrained.

Besides the theoretical analyses, a design for a human-subject study is presented (Sixt et al. 2022b). Given the absence of ground truth for correct explanations, human-subject studies serve as the gold standard for evaluating explainability methods.

In our study design, we proposed a bias-detection task, i.e. users have to use explanation to detect whether the model is biased towards a certain property. In the study, we found that a simple baseline based on the model's logit scores performed surprisingly well, and even a strong generative explanation method did not significantly outperform the baseline.

In the manuscript (Nader et al. 2022), we extended k-NN regression by including higher-order derivatives. For the evaluation, we conducted a thorough comparison against existing methods on a large number of datasets.

The contributions of this thesis aim to positively impact the field of explainable AI by providing more rigorous and transparent methods for evaluating and

understanding model explanations. I contribute to the ongoing development of reliable, accurate, and user-friendly AI systems.

1.4 Structure of the Thesis

The thesis is structured as follows: In the next chapter 2, I provide an overview of the history of explainable AI and the field's current state. Subsequently, each publication is presented with a brief introduction in chapters 3,4, 5, 6, and 7. Finally, I conclude with a discussion of my work and future directions in Chapter 8.

1.5 Contributions To Individual Publications

In this section, I summarize the contributions to the publications according to § 7 Abs. 3 of the doctoral regulations from 1st December 2021:

1.5.1 Restricting the Flow: Information Bottlenecks for Attribution

L.S. conceived the general idea. K.S. had the idea of the per-sample bottleneck. K.S. implemented the algorithm. L.S. extended the evaluation and added more baselines. L.S., K.S., F.T., and T.L wrote the paper together. F.T. and T.L. provided supervision. The open-source library was implemented by K.S. and L.S.

1.5.2 When Explanations Lie: Why Many Modified BP Attributions Fail

L.S. had the idea after performing the sanity checks in (Schulz et al. 2020). L.S. conducted the empirical experiments. M.G. proved the theoretical result. L.S., M.G., and T.L. wrote the paper. T.L. provided supervision.

1.5.3 Do Users Benefit From Interpretable Vision? A User Study, Baseline, And Dataset

The paper's conception was a collaborative process between L.S., M.S., and T.L. P.W. wrote the original Two4Two data generator, which L.S. significantly extended. M.S. designed the user study. O.-I.P. worked on a counterfactual method, designed the questionnaire to measure the user's understanding together with M.S. and ran pre-study interviews. L.S trained the models and created the explanations used in the study. M.S. and L.S performed the statistical analysis. L.S., M.S., O.-I.P., and T.L wrote the manuscript. T.L. provided supervision.

1.5.4 DNNR: Differential Nearest Neighbors Regression

Y.N. conceived the idea. Y.N. implemented the algorithm and experiments. L.S. derived the theoretical results. Y.N., L.S., and T.L. wrote the paper. L.S. and T.L. provided supervision.

1.5.5 A Rigorous Study Of The Deep Taylor Decomposition

L.S. had the idea, implemented the empirical experiments, and derived the theoretical results. L.S. wrote the paper together with T.L.. T.L. provided supervision.

2 History Of The Field

The long history of explanation methods is often overlooked. For example, (Bauer et al. 2021) state that "[...] interpretable machine learning as a field is still in its infancy and requires more scrutiny and rigorous scientific research". While I agree with the call for more scrutiny and rigor, I would like to point out that the field actually has a rich history of more than 50 years. In this section, I aim to recapitulate each period of research by outlining the major motivations, questions, and ideas from the late 1960s until the arrival of deep learning at the beginning of 2010.

The Beginning: The first AI system that was in some sense able to explain its decisions was Winograd's SHRDLU system (Winograd 1972). The system was a dialogue system that could understand English commands about a simulated 3D world. Users gave natural language commands to the system to move objects such as boxes, blocks, and pyramids around in a the simulated 3D world. A typical command would be *Pick up a green block*. The user could also ask *why* questions such as: *Why did you do that? TO CLEAN OFF THE RED CUBE.* The explanation was derived from the internal stack of recent actions. However, (Winograd 1972) did not discuss why one should include explanations in a dialogue system. Therefore, I assume that the main motivation was to cover as much English grammar as possible.

I found the first call for explanations in (Gorry 1973) with in the context of medical diagnosis:

> "The other side of the coin is explanation. If experts are to use and improve the program directly, then it must be able to explain the reasons for its actions. Furthermore, this explanation must be in terms the physicians can understand. The steps in a deduction and the facts employed must be identified for the expert so that he or she can correct one or more of them if necessary. As a corollary, the user must be able to easily find out what the program knows about a particular subject." — (Gorry 1973)

Although Gorry raised the need to explain the decisions first, his own system (Gorry 1967), which was based on the famous ELIZA program (Weizenbaum 1966), was not able to explain its decisions.

Influenced by Gorry's remark, (Shortliffe 1976) developed the first system (MYCIN) that provided explanations for its decision. A doctor could ask MYCIN to explain its rules and the steps it took to reach a conclusion.

(Teach et al. 1981) surveyed physicians about computer-based consultation systems and found that a system's explanations were the most important feature for physicians. The need for explanations in the medical field was recognized early on.

Explanation-Based Learning: A planning system for robots (Fikes et al. 1972) inspired the development of explanation-based learning systems in the 1980s (Silver 1983; Mitchell 1982; DeJong 1981).

The main idea behind explanation-based learning is to generalize from examples by identifying and removing irrelevant information, thus resulting in more general examples. For instance, suppose an agent is asked to arrange boxes of different colors and sizes. As the boxes are stacked above each other, the agent has to solve a series of sub-problems. The solution to one specific problem (e.g., "Stack the blue box on top of the red box") can be reused later if the agent identifies irrelevant information (e.g., the specific color). While in the simplest case, single attributes (e.g. color = blue) are replaced by intervals (e.g., "color = any RGB value"), more elaborated generalizations can be made by describing partial intervals or interactions between attributes.

However, a major limitation of explanation-based learning systems is the need for a domain theory to specify irrelevant properties. The publications (Mitchell et al. 1986; Dejong et al. 1986) and the survey (Ellman 1989) provide a good introduction to the field.

Rules And Neural Networks In the late 1980s, connectionism experienced a resurgence in popularity with the popularization of the backpropagation algorithm in (Rumelhart et al. 1986). With this renewed interest came the question of how to extract rules from neural networks. An early approach to this question was proposed by (Berzuini 1988) who extracted rules from a simple regression model. Soon after, (Towell et al. 1991) proposed the KBANN algorithm, which combined an artificial neural network with an expert system. The authors concluded that *"interpretation of ANNs (artificial neural networks) after learning can be helpful in understanding why the ANN behaves as it does"*.

The construction of rules from neural networks became an active line of research in the early 1990s, with the goal of increasing the comprehensibility of neural networks. Studies in this area include those by (Towell et al. 1991; Mahoney et al. 1992; Setiono et al. 1996; Tresp et al. 1992; Craven et al. 1993; Giles et al. 1993). Relevant Ph.D. theses in this area include (Towell 1993) and (Craven et al. 1993).

Relevance Measures Pruning methods for neural networks were the first to analyze the relevance of neurons. One of the first works in this area was proposed by (Mozer et al. 1988), who suggested using the gradient with respect to the neurons to prune them.

(LeCun et al. 1989) additionally incorporated second-order terms and noted that deleting parameters by the order of saliency causes a significantly smaller increase of the objective function than deleting them according to their magnitude. This established early on that a neuron's magnitude and saliency differ.

(Sanger 1989) was one of the first works to analyze neurons' contributions with the goal to understand the networks better. They analyzed the NETtalk network (Sejnowski et al. 1987), which learned to transcript English words to phonemes.

(Garson 1991) was motivated by the black box character of neural networks and aimed at deriving relative importance for the inputs. (Garson 1991) calculated the inputs' importance of a 2-layered MLP by multiplying the layer's weights and normalizing them per hidden and input neuron (see also Goh 1995). This work (Garson 1991) is remarkable since it is the first work that raises the question of how to interpret the weights of a neural network. Additionally, they simulated the data-generating process in order to have a ground truth to compare their results – something later work omitted.

However, (Garson 1991) ignored the non-linearity of the activation function and also took the absolute value of the weights. Consequently, it is not surprising that (Sarle 2000) found the importance to be not calculated correctly on a simple example (additive model with three inputs). Extensions of this approach (Milne 1995; Gedeon 1997) suffer from the same problem.

In an application to trout growth, (Lek et al. 1996) quantifies the importance of input features by changing each dimension separately and measuring the effect on the model output. Another study (Scardi et al. 1999) uses randomization of inputs to measure importance in the context of phytoplankton primary production.

A review of different methods for measuring input sensitivity was conducted by (Gevrey et al. 2003). The methods used in this review include partial input

differences (Dimopoulos et al. 1995; Fu et al. 1993), input perturbation (Scardi et al. 1999), Garson's method (Garson 1991), the profile method (Lek et al. 1996), and the stepwise method (Sung 1998). The stability of each method was calculated by training the network ten times with different random seeds, and the PaD method was found to give the most stable results. However, the study design was questioned in (Olden 2004) due to its reliance on an empirical dataset with an unknown "true correlation structure" and the influence of different random initializations on the stability measure. To address these concerns, they proposed using a synthetic dataset and measuring how well the methods ranked the input features. According to (Olden 2004), the best-performing method was the Connection Weight Approach (Olden et al. 2002).

Many publications on input sensitivity were motivated by ecological applications and were published in the journal *Ecological Modelling* (Lek et al. 1996; Dimopoulos et al. 1999; Scardi et al. 1999; Olden et al. 2002; Gevrey et al. 2003; Pastor-Bárcenas et al. 2005; Gevrey et al. 2006). Probably, ecological modeling strikes a good balance between being too complex for linear models but still simple enough that shallow MLPs were sufficient. Furthermore, quantifying the effect of input variables might steer and prioritize real-world changes such as the renaturation of rivers or regulation of fertilizer use.

In the 1990s and early 2000s, the problem of feature importance was solely studied for the importance of the whole dataset and not for the importance of individual examples. The main reason was that the models were shallow (e.g., one hidden layer) and that the same feature dimension was used for all examples.

The first work that analyzed the importance of features for individual examples was (Baehrens et al. 2010). The work (Simonyan et al. 2013) introduced saliency maps for deep neural networks. For the related work on saliency maps since (Simonyan et al. 2013), see the related work section of (Schulz et al. 2020) in chapter 3.

Summary of the History First, it is surprising that (Gorry 1973) already identified the black-box problem as problematic. The field of explanation-based learning (EBL), popular in the 80s was a completely different way of generalization than is used today. EBL puts explanations at the center of learning. For neural networks, rule extraction and pruning were early applications of explainability. The research on feature importance in the 1990s and 2000s has similarities with the current field of saliency maps. For example, the points made in the controversy between (Gevrey et al. 2003) and (Olden 2004) on how to evaluate methods is still valid today.

3 Restricting the Flow: Information Bottlenecks for Attribution

Saliency methods highlight the regions of an input image based on their relevance for the model's decision. In recent years, various saliency methods have been proposed (Simonyan et al. 2013; Shrikumar et al. 2017a; Bach et al. 2015). A limitation of these methods is that they only provide relative importance scores for each input feature, i.e. the importance can only be compared within the same image. For instance, input gradients (the change in the output w.r.t. the input dimension) might have the unit of logit/pixel, but they are not comparable across models and datasets, as the output and input scale might differ. Furthermore, as saliency maps are post-processed by outlier clipping and normalized to the range $[0, 1]$, they can also be difficult to compare across different samples.

To introduce an absolute frame of reference, we developed a method that provides absolute importance scores for each input feature in bits per pixel. This absolute frame of reference allows us to compare the importance across different models and datasets. The work builds on the master thesis of Karl Schulz (Schulz 2019), which I co-supervised.

Building on Karl's thesis, we published our work "Restricting the Flow: Information Bottlenecks for Attribution" at ICLR 2020 (Schulz et al. 2020). The main idea of our work is to add noise to reduce the information flow from the input to the output. By minimizing the amount of information while retaining the original network's output, we can estimate the importance of each image region. Our method is inspired by the information bottleneck method (Tishby et al. 2000) and the variational autoencoder (Kingma et al. 2013).

Furthermore, we provided a thorough technical evaluation of Information Bottleneck Attribution (IBA) using the MoRF / LeRF metrics, the weight-randomization sanity checks, and the sensitivity-n (Ancona et al. 2017) metric. From all eleven evaluated methods, our method consistently outperformed the other methods.

Other researchers extended our work to the transformer architecture (Jiang et al. 2020) and it was used as a baseline in (Cao et al. 2020). In (Cao et al. 2020), IBA was criticized for having a so-called hindsight bias. As IBA relies on gradient information obtained from the model output, it would remove information too aggressively. For example, at a current layer, it might not have been determined whether a certain feature is important or not. However, using information from later layers through the backpropagation of the gradient, this feature might still be masked. While this criticism is valid in principle, it could be easily fixed by using only the local layer's activation as input to the Readout network.

The follow-up work InputIBA (Zhang et al. 2021b) estimated the input features' relevance scores. Their approach allows using IBA also for NLP models where saliency maps cannot be upscaled as for convolutional networks. Compared to various baselines and a wide range of tasks (weight-randomization sanity checks, Sensitivity-N, Insertion/Deletion, ROAR, Effective Heat Ratios), they report InputIBA to be consistently performing as the best method.

Besides the work building on our method, we received over 120 citations within the past few years. Also, our work was accepted for a long talk at ICLR 2020, which placed it in the top 7% of all accepted papers.

The work on "Restricting the Flow: Information Bottlenecks for Attribution" is an important part of my thesis. We have developed a method that quantifies the importance of input features in deep learning models, with the theoretical properties of quantifying the amount of used information. This provides an absolute frame of reference to enable comparisons of importance across different models and datasets. This work shows that theoretical foundations can be used on the one hand to develop new methods and on the other hand that theoretical foundations also provide an improvement compared to heuristics.

RESTRICTING THE FLOW: INFORMATION BOTTLENECKS FOR ATTRIBUTION

Karl Schulz[1*†], **Leon Sixt**[2*], **Federico Tombari**[1], **Tim Landgraf**[2]
* contributed equally † work done at the Freie Universität Berlin
Technische Universität München[1] Freie Universität Berlin[2]
Corresponding authors: karl.schulz@tum.de, leon.sixt@fu-berlin.de

ABSTRACT

Attribution methods provide insights into the decision-making of machine learning models like artificial neural networks. For a given input sample, they assign a relevance score to each individual input variable, such as the pixels of an image. In this work, we adopt the information bottleneck concept for attribution. By adding noise to intermediate feature maps, we restrict the flow of information and can quantify (in bits) how much information image regions provide. We compare our method against ten baselines using three different metrics on VGG-16 and ResNet-50, and find that our methods outperform all baselines in five out of six settings. The method's information-theoretic foundation provides an absolute frame of reference for attribution values (bits) and a guarantee that regions scored close to zero are not necessary for the network's decision.

1 INTRODUCTION

Deep neural networks have become state of the art in many real-world applications. However, their increasing complexity makes it difficult to explain the model's output. For some applications such as in medical decision making or autonomous driving, model interpretability is an important requirement with legal implications. Attribution methods (Selvaraju et al., 2017; Zeiler & Fergus, 2014; Smilkov et al., 2017) aim to explain the model behavior by assigning a relevance score to each input variable. When applied to images, the relevance scores can be visualized as heatmaps over the input pixel space, thus highlighting salient areas relevant for the network's decision.

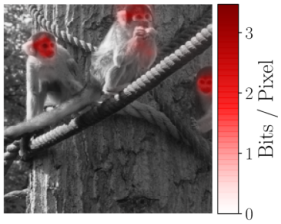


Figure 1: Exemplary heatmap of the Per-Sample Bottleneck for the VGG-16.

For attribution, no ground truth exists. If an attribution heatmap highlights subjectively irrelevant areas, this might correctly reflect the network's unexpected way of processing the data, or the heatmap might be inaccurate (Nie et al., 2018; Viering et al., 2019; Sixt et al., 2019). Given an image of a railway locomotive, the attribution map might highlight the train tracks instead of the train itself. Current attribution methods cannot guarantee that the network is ignroing the low-scored locomotive for the prediction.

We propose a novel attribution method that estimates the amount of information an image region provides to the network's prediction. We use a variational approximation to upper-bound this estimate and therefore, can guarantee that areas with zero bits of information are not necessary for the prediction. Figure 1 shows an exemplary heatmap of our method. Up to 3 bits per pixel are available for regions corresponding to the monkeys' faces, whereas the tree is scored with close to zero bits per pixel. We can thus guarantee that the tree is not necessary for predicting the correct class.

To estimate the amount of information, we adapt the information bottleneck concept (Tishby et al., 2000; Alemi et al., 2017). The bottleneck is inserted into an existing neural network and restricts the information flow by adding noise to the activation maps. Unimportant activations are replaced almost entirely by noise, removing all information for subsequent network layers. We developed two approaches to learn the parameters of the bottleneck – either using a single sample (Per-Sample Bottleneck), or the entire dataset (Readout Bottleneck).

We evaluate against ten different baselines. First, we calculated the Sensitivity-n metric proposed by Ancona et al. (2018). Secondly, we quantified how well the object of interest was localized using bounding boxes and extend the degradation task proposed by Ancona et al. (2017). In all these metrics, our method outperforms the baselines consistently. Additionally, we test the impact of cascading layer-wise weight randomizations on the attribution heatmaps (Adebayo et al., 2018). For reproducibility, we share our source code and provide an easy-to-use* implementation.

We name our method *IBA* which stands for Information Bottleneck Attribution. It provides a theoretic upper-bound on the used information while demonstrating strong empirical performance. Our work improves model interpretablility and increases trust in attribution results.

To summarize our contributions:

- We adapt the information bottleneck concept for attribution to estimate the information used by the network. Information theory provides a guarantee that areas scored irrelevant are indeed not necessary for the network's prediction.
- We propose two ways – *Per-Sample* and *Readout* Bottleneck – to learn the parameters of the information bottleneck.
- We contribute a novel evaluation method for attribution based on bounding boxes and we also extend the metric proposed by Ancona et al. (2017) to provide a single scalar value and improve the metric's comparability between different network architectures.

2 Related Work

Attribution is an active research topic. *Gradient Maps* (Baehrens et al., 2010) and *Saliency Maps* (Simonyan & Zisserman, 2014) are based on calculating the gradient of the target output neuron w.r.t. to the input features. *Integrated Gradient* (Sundararajan et al., 2017) and *SmoothGrad* (Smilkov et al., 2017) improve over gradient-based attribution maps by averaging the gradient of multiple inputs, either over brightness level interpolations or in a local neighborhood. Other methods, such as *Layer-wise Relevance Propagation (LRP)* (Bach et al., 2015), *Deep Taylor Decomposition (DTD)* (Montavon et al., 2017), *Guided Backpropagation (GuidedBP)* (Springenberg et al., 2014) or *DeepLIFT* (Shrikumar et al., 2017) modify the propagation rule. *PatternAttribution* (Kindermans et al., 2018) builds upon DTD by estimating the signal's direction for the backward propagation. Perturbation-based methods are not based on backpropagation and treat the model as a black-box. *Occlusion* (Zeiler & Fergus, 2014) measures the importance as the drop in classification accuracy after replacing individual image patches with zeros. Similarly, blurring image patches has been used to quantify the importance of high-frequency features (Greydanus et al., 2018) .

Grad-Cam (Selvaraju et al., 2017) take the activations of the final convolutional layer to compute relevance scores. They also combine their method with GuidedBP: *GuidedGrad-CAM*. Ribeiro et al. (2016) uses image superpixels to explain deep neural networks. High-level concepts rather input pixels are scored by *TCAV* (Kim et al., 2018). Khakzar et al. (2019) prune irrelevant neurons. Similar to our work, Macdonald et al. (2019) uses a rate-distortion perspective, but minimize the norm of the mask instead of the shared information. To the best of our knowledge, we are the first to estimate the amount of used information for attribution purposes.

Although many attribution methods exist, no standard evaluation benchmark is established. Thus, determining the state of the art is difficult. The performance of attribution methods is highly dependent on the used model and dataset. Often only a purely visual comparison is performed (Smilkov et al., 2017; Springenberg et al., 2014; Montavon et al., 2017; Sundararajan et al., 2017; Bach et al., 2015). The most commonly used benchmark is to degrade input images according to the attribution heatmap and measure the impact on the model output. (Kindermans et al., 2018; Samek et al., 2016; Ancona et al., 2017). The Sensitivity-n score (Ancona et al., 2018) is obtained by randomly masking the input image and then measuring the correlation between removed attribution mass and model performance. For the ROAR score (Hooker et al., 2018), the network is trained from scratch on the degraded images. It is computationally expensive but avoids out-of-domain inputs – an inherent problem of masking. Adebayo et al. (2018) proposes a sanity check: if the network's parameters are randomized, the attribution output should change too.

*https://github.com/BioroboticsLab/IBA

Adding noise to a signal reduces the amount of information (Shannon, 1948). It is a popular way to regularize neural networks (Srivastava et al., 2014; Kingma et al., 2015; Gal et al., 2017). For regularization, the noise is applied independently from the input and no attribution maps can be obtained. In Variational Autoencoders (VAEs) (Kingma & Welling, 2013), noise restricts the information capacity of the latent representation. In our work, we construct a similar information bottleneck that can be inserted into an existing network. Deep convolutional neural networks have been augmented with information bottlenecks before to improve the generalization and robustness against adversarial examples (Achille & Soatto, 2018; Alemi et al., 2017). Zhmoginov et al. (2019) and Taghanaki et al. (2019) extract salient regions using information bottlenecks. In contrast to our work, they add the information bottleneck already when training the network and do not focus on post-hoc explanations.

3 Information Bottleneck for Attribution

Instead of a backpropagation approach, we quantify the flow of information through the network in the *forward* pass. Given a pre-trained model, we inject noise into a feature map, which restrains the flow of information through it. We optimize the intensity of the noise to minimize the information flow, while simultaneously maximizing the original model objective, the classification score. The parameters of the original model are not changed.

3.1 Information Bottleneck for Attribution

Generally, the information bottleneck concept (Tishby et al., 2000) describes a limitation of available information. Usually, the labels Y are predicted using all information from the input X. The information bottleneck limits the information to predict Y by introducing a new random variable Z. The information the new variable Z shares with the labels Y is maximized while minimizing the information Z and X share :

$$\max I[Y;Z] - \beta I[X,Z] \,, \tag{1}$$

where $I[X,Z]$ denotes the mutual information and β controls the trade-off between predicting the labels well and using little information of X. A common way to reduce the amount of information is to add noise (Alemi et al., 2017; Kingma & Welling, 2013).

For attribution, we inject an information bottleneck into a pretrained network. The bottleneck is inserted into an layer which still contains local information, e.g. for the ResNet the bottleneck is added after conv3_* (after the last conv3 block). Let R denote the intermediate representations at this specific depth of the network, i.e. $R = f_l(X)$ where f_l is the l-th layer output. We want to reduce information in R by adding noise. As the neural network is trained already, adding noise should preserve the variance of the input to the following layers. Therefore, we also damp the signal R when increasing the noise, effectively replacing the signal partly with noise. In the extreme case, when no signal is transmitted, we replace R completely with noise of the same mean and variance as R. For this purpose, we estimate the mean μ_R and variance σ_R^2 of each feature of R empirically. As information bottleneck, we then apply a linear interpolation between signal and noise:

$$Z = \lambda(X)R + (1 - \lambda(X))\,\epsilon \,, \tag{2}$$

where $\epsilon \sim \mathcal{N}(\mu_R, \sigma_R^2)$ and $\lambda(X)$ controls the damping of the signal and the addition of the noise. The value of λ is a tensor with the same dimensions as R and $\lambda_i \in [0,1]$. Given $\lambda_i(X) = 1$ at the feature map location i, the bottleneck transmits all information as $Z_i = R_i$. Whereas if $\lambda_i = 0$, then $Z_i = \epsilon$ and all information of R_i is lost and replaced with noise. It could be tempting to think that Z from equation 2 has the same mean and variance as R. This is not the case in general as $\lambda(X)$ and R both depend on X (for more detail see Appendix E).

In our method, we consider an area relevant if it contains useful information for classification. We need to estimate how much information Z still contains about R. This quantity is the mutual information $\mathbf{I}[R,Z]$ that can be written as:

$$\mathbf{I}[R,Z] = \mathbb{E}_R[D_{\mathrm{KL}}[P(Z|R)||P(Z)]] \,, \tag{3}$$

where $P(Z|R)$ and $P(Z)$ denote the respective probability distributions. We have no analytic expression for $P(Z)$ since it would be necessary to integrate over the feature map $p(z) = \int_R p(z|r)p(r)\mathrm{d}r$

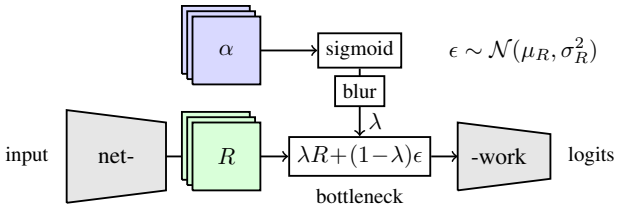


Figure 2: *Per-Sample Bottleneck*: The mask (blue) contains an α_i for each r_i in the intermediate feature maps R (green). The parameter α controls how much information is passed to the next layer. The mask α is optimized for each sample individually according to equation 6.

– an intractable integral. It is a common problem that the mutual information can not be computed exactly but is instead approximated (Poole et al., 2019; Suzuki et al., 2008). We resort to a variational approximation $Q(Z) = \mathcal{N}(\mu_R, \sigma_R)$ which assumes that all dimensions of Z are distributed normally and independent – a reasonable assumption, as activations after linear or convolutional layers tend to have a Gaussian distribution (Klambauer et al., 2017). The independence assumption will generally not hold. However, this only overestimates the mutual information as shown below. Substituting $Q(Z)$ into the previous equation 3, we obtain:

$$\mathrm{I}[R, Z] = \mathbb{E}_R[D_{\mathrm{KL}}[P(Z|R)||Q(Z)]] - D_{\mathrm{KL}}[P(Z)||Q(Z)]\,. \tag{4}$$

The derivation is shown in appendix D and follows Alemi et al. (2017). The first term contains the KL-divergence between two normal distributions, which is easy to evaluated and we use it to approximate the mutual information. The information loss function $\mathcal{L}_I$ is therefore:

$$\mathcal{L}_I = \mathbb{E}_R[D_{\mathrm{KL}}[P(Z|R)||Q(Z)]]\,. \tag{5}$$

We know that $\mathcal{L}_I$ overestimates the mutual information, i.e. $\mathcal{L}_I \geq \mathrm{I}[R, Z]$ as the second KL-divergence term $D_{\mathrm{KL}}[P(Z)||Q(Z)]$ has to be positive. If $\mathcal{L}_I$ is zero for an area, we can guarantee that information from this area is not necessary for the network's prediction. Information from this area might still be used when no noise is added.

We aim to keep only the information necessary for correct classification. Thus, the mutual information should be minimal while the classification score should remain high. Let $\mathcal{L}_{CE}$ be the cross-entropy of the classification. Then, we obtain the following optimization problem:

$$\mathcal{L} = \mathcal{L}_{CE} + \beta\mathcal{L}_I\,, \tag{6}$$

where the parameter β controls the relative importance of both objectives. For a small β, more bits of information are flowing and less for a higher β. We propose two ways of finding the parameters λ – the Per-Sample Bottleneck and the Readout Bottleneck. For the Per-Sample Bottleneck, we optimize λ for each image individually, whereas in the readout bottleneck, we train a distinct neural network to predict λ.

3.2 Per-Sample Bottleneck

For the *Per-Sample Bottleneck*, we use the bottleneck formulation described above and optimize $\mathcal{L}$ for individual samples – not for the complete dataset at once. Given a sample x, λ is fitted to the sample to reflect important and unimportant regions in the feature space. A diagram of the Per-Sample Bottleneck is shown in Figure 2.

Parameterization The bottleneck parameters λ have to be in $[0, 1]$. To simplify optimization, we parametrize $\lambda = \mathrm{sigmoid}(\alpha)$. This parametrization allows $\alpha \in \mathbb{R}^d$ and avoids any clipping of λ to $[0, 1]$ during optimization.

Initialization As when training neural networks, the initialization of parameters matters. In the beginning, we want to retain all the information. For all dimensions i, we initialize $\alpha_i = 5$ and thus $\lambda_i \approx 0.993 \Rightarrow Z \approx R$. At first, the bottleneck has practically no impact on the model performance. It then deviates from this starting point to suppress unimportant regions.

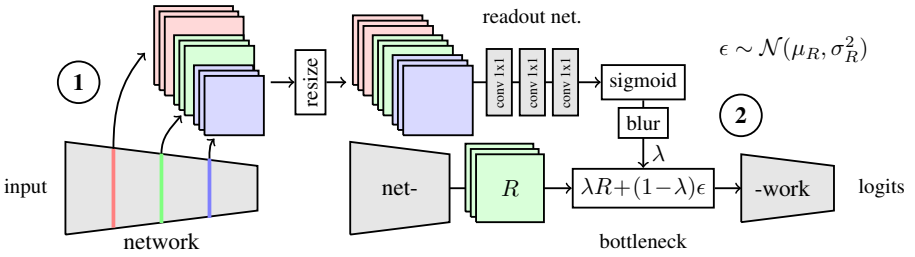


Figure 3: *Readout Bottleneck*: In the first forward pass (1), feature maps are collected at different depths. The readout network uses a resized version of the feature maps to predict the parameters for the bottleneck layer. In the second forward pass (2), the bottleneck is inserted and noise added. All parameters of the analyzed network are kept fixed.

Optimization We do 10 iterations using the Adam optimizer (Kingma & Ba, 2014) with learning rate 1 to fit the mask α. To stabilize the training, we copy the single sample 10 times and apply different noise to each. In total, we execute the model on 100 samples to create a heatmap, comparable to other methods such as SmoothGrad. After the optimization, the model usually predicts the target with probability close to 1, indicating that only negative evidence was removed.

Measure of information To measure the importance of each feature in Z, we evaluate $D_{\mathrm{KL}}(P(Z|R)||Q(Z))$ per dimension. It shows where the information flows. We obtain a two-dimensional heatmap m by summing over the channel axis: $m_{[h,w]} = \sum_{i=0}^{c} D_{\mathrm{KL}}(P(Z_{[i,h,w]}|R_{[i,h,w]})||Q(Z_{[i,h,w]}))$. As convolutional neural networks preserve the locality in their channel maps, we use bilinear interpolation to resize the map to the image size. For the ResNet-50, we insert the bottleneck after layer conv3_*. Choosing a later layer with a lower spatial resolution would increase the blurriness of the attribution maps, due to the required interpolation. The effect of different depth choices for the bottleneck is visualized in figure 4.

Enforcing local smoothness Pooling operations and convolutional layers with stride greater than 1 are ignoring parts of the input. The ignored parts cause the Per-Sample Bottleneck to overfit to a grid structure (shown in Appendix B). To obtain a robust and smooth attribution map, we convolve the sigmoid output with a fixed Gaussian kernel with standard deviation σ_s. Smoothing the mask during training is *not* equivalent to smoothing the resulting attribution map, as during training also the gradient is averaged locally. Combining everything, the parametrization for the Per-Sample Bottleneck is:

$$\lambda = \mathrm{blur}(\sigma_s, \mathrm{sigmoid}(\alpha)) \; . \tag{7}$$

3.3 Readout Bottleneck

For the *Readout Bottleneck*, we train a second neural network to predict the mask α. In contrast to the Per-Sample Bottleneck, this model is trained on the entire training set. In Figure 3, the Readout Bottleneck is depicted. Kümmerer et al. (2014) introduced the readout concept for gaze prediction. The general idea is to collect feature maps from different depths and then combine them using 1x1 convolutions.

In a first forward pass, no noise is added and we collect the different feature maps. As the spatial resolution of the feature maps differs, we interpolate them bilinearly to match the spatial dimensions of the bottleneck layer. The readout network then predicts the information mask based on the collected feature maps. In a second forward pass, we insert the bottleneck layer into the network and restrict the flow of information.

Except for the learned importance values, the Readout Bottleneck is identical to the mechanism of the Per-Sample Bottleneck. The measure of information works in the same way as for the Per-Sample Bottleneck and we also use the same smoothing. Given a new sample, we can obtain a heatmap by merely collecting the feature maps and executing the readout network.

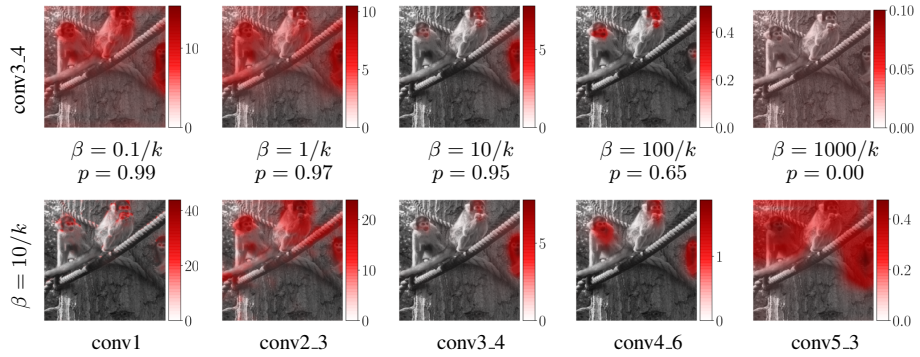


Figure 4: Effect of varying layer depth and β on the Per-Sample Bottleneck for the ResNet-50. The color bars measure the bits per input pixel. The resulting output probability of the correct class $p = p(y|x, \beta)$ is decreasing for higher β.

The readout network consists of three 1x1 convolutional layers. ReLU activations are applied between convolutional layers and a final sigmoid activation yields $\lambda \in [0, 1]$. As the input consists of upscaled feature maps, the field-of-view is large, although the network itself only uses 1x1 conv. kernels.

4 Evaluation

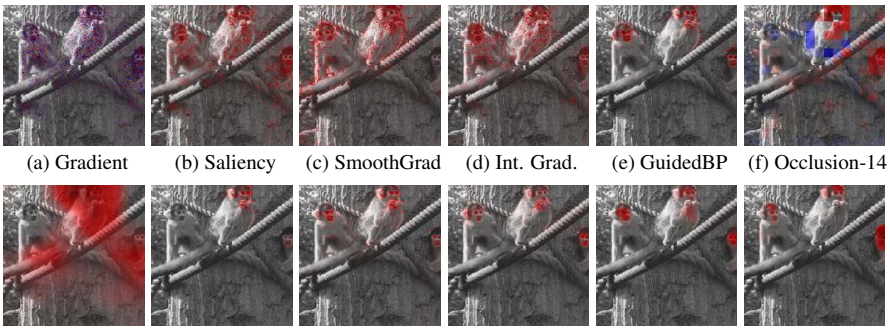
4.1 Experimental Setup

As neural network architectures, we selected the ResNet-50 (He et al., 2016) and the VGG-16 Simonyan & Zisserman (2014), using pretrained weights from the torchvision package (Paszke et al., 2017). These two models cover a range of concepts: dimensionality by stride or max-pooling, residual connections, batch normalization, dropout, low depth (16-weight-layer VGG), and high depth (50-weight-layer ResNet). This variety makes the evaluation less likely to overfit on a specific model type. They are commonly used in the literature concerning attribution methods. For PatternAttribution on the VGG-16, we obtained weights for the signal estimators from Kindermans et al. (2018).

As naive baselines, we selected random attribution, Occlusion with patch sizes 8x8 and 14x14, and Gradient Maps. SmoothGrad and Integrated Gradients cover methods that accumulate gradients. We include three methods with modified backpropagation rules: PatternAttribution, GuidedBP, and LRP. As our implementation of PatternAttribution and LRP does not support skip connections, we report no results for them on the ResNet-50. We also include Grad-CAM and its combination with GuidedBP, GuidedGrad-CAM.

For the compared methods, we use the hyperparameters suggested by the original authors. For LRP (Bach et al., 2015), different rules exist. We include the most commonly used α=1, β=0, ϵ=0 rule which is also displayed in the figures. We also include α=0, β=-1, ϵ=5 as it gives better results on the bounding-box task. For sensitivity-n and sanity checks, only the later LRP variante is evaluted. For our methods, the hyperparameters are obtained using grid search with the degradation metric as objective (Appendix F). The readout network is trained on the training set of the ILSVRC12 dataset (Russakovsky et al., 2015) for $E = 20$ epochs.

The optimization objective of the bottleneck is $\mathcal{L}_{CE} + \beta\mathcal{L}_I$ as given in equation 6. Generally, the information loss $\mathcal{L}_I$ is larger than the classifier loss by several orders of magnitude as it sums over height h, width w and channels c. We therefore use $k = hwc$ as a reference point to select β. In figure 4, we display the effect of different orders of β and the effect of layer depth for ResNet-50 (see Appendix C for VGG-16). A uniform uninformative heatmap is obtained for $\beta \geq 1000/k$ – all information gets discarded, resulting in an output probability of 0 for the correct class. The heatmap for $\beta = 0.1/k$ is similar to $\beta = 1/k$ but more information is passed, i.e. less noises is added. In appendix F, we also compare pre- to post-training accuracy and estimate the mutual information for



(a) Gradient (b) Saliency (c) SmoothGrad (d) Int. Grad. (e) GuidedBP (f) Occlusion-14

(g) Grad-CAM (h) G.Grad-CAM (i) PatternAttr. (j) LRP (k) *Per-Sample* (l) *Readout*

Figure 5: Heatmaps of all implemented methods for the VGG-16 (see Appendix A for more).

both Bottleneck types. For the Per-Sample Bottleneck, we investigate $\beta = 1/k, 10/k, 100/k$. The Readout network is trained with the best performing value $\beta = 10/k$.

4.2 Qualitative Assessment

In Figure 5, the heatmaps of all evaluated samples are shown (for more samples, see Appendix A). Subjectively, both the Per-Sample and Readout Bottleneck identify areas relevant to the classification well. While Guided Backpropagation and PatternAttribution tend to highlight edges, the Per-Sample Bottleneck focuses on image regions. For the Readout Bottleneck, the attribution is concentrated a little more on the object edges. Compared to Grad-CAM, both our methods are more specific, i.e. fewer pixels are scored high.

4.3 Sanity Check: Randomization of Model Parameters

Adebayo et al. (2018) investigates the effect of parameter randomization on attribution masks. A sound attribution method should depend on the entire network's parameter set. Starting from the last layer, an increasing proportion of the network parameters is re-initialized until all parameters are random. We excluded PatternAttribution as the randomization would require to re-estimate the signal directions. The difference between the original heatmap and the heatmap obtained from the randomized model is quantified using SSIM (Wang et al., 2004). We discuss implementation details in the appendix G.1.

In figure 6, we display the results of the sanity check. For our methods, we observe that randomizing the final dense layer drops the mean SSIM by around 0.4. The values for the Readout Bottleneck are of limited expressiveness as we did not re-train it after randomization. For SmoothGrad and Int. Gradients, the SSIM drops by more than 0.4. The heatmaps of GuidedBP and LRP remain similar even if large portion of the network's parameters are randomized – they do not explain the network prediction faithfully. Nie et al. (2018) provides theoretical insights about why GuidedBP fails. Sixt et al. (2019) analyzes why LRP and other modified BP methods fail.

4.4 Sensitivity-n

Ancona et al. (2018) proposed Sensitivity-n as an evaluation metric for attribution methods. Sensitivity-n masks the network's input randomly and then measures how strongly the amount of attribution in the mask correlates with the drop in classifier score.Given a set T_n containing n randomly selected pixel indices, Sensitivity-n measures the Pearson correlation coefficient:

$$\text{corr}\left(\sum_{i \in T_n} R_i(x),\ S_c(x) - S_c(x_{[x_{T_n}=0]})\right), \tag{8}$$

where $S_c(x)$ is the classifier logit output for class c, R_i is the relevance at pixel i and $x_{[x_{T_n}=0]}$ denotes the input with all pixels in T_n set to zero. As in the original paper, we pick the number of masked

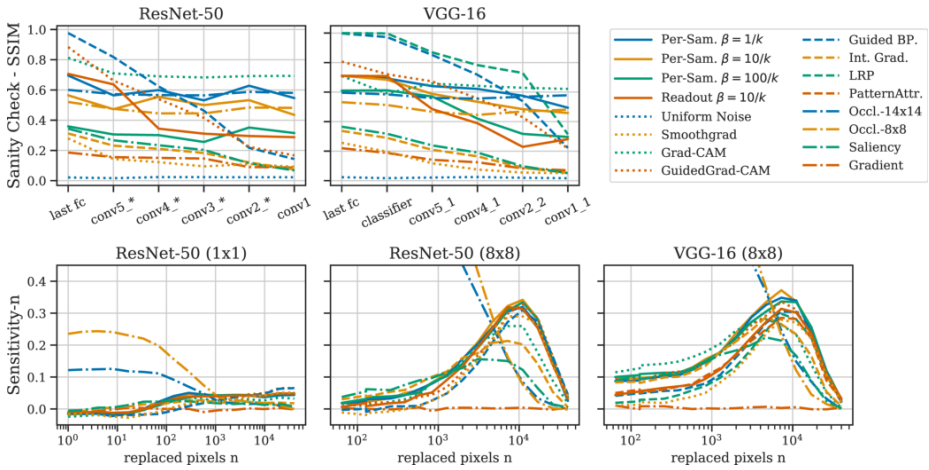


Figure 6: *First row* drop in SSIM score when network layers are randomized. Best viewed in color. *Second-row* Sensitivity-n scores for ResNet-50 and VGG-16. For the ResNet-50, also tile size 1x1 is shown. We clipped the y-axis at 0.4 to improve discriminability.

pixels n in a log-scale between 1 and 80% of all pixels. For each n, we generate 100 different index sets T and test each on 1000 randomly selected images from the validation set. The correlation is calculated over the different index sets and then averaged over all images.

In Figure 6, the Sensitivity-n scores are shown for ResNet-50 and VGG-16. When masking inputs pixel-wise as done in Ancona et al. (2018), the Sensitivity-n score across methods are not well discriminable, all scores range within the lower 10% of the scale. We ran the metric with a tile size of 8x8 pixels. This resulted in an increase of the Sensitivity-n score to 30% of the scale. Although not shown in the figure due to the zooming, Occlusion-8x8 performs perfectly for small n as its relevance scores correspond directly to the drop in logits per 8x8 tile. We find that the Per-Sample Bottlenecks $\beta = 10/k$ perform best for both models above $n = 2 \cdot 10^3$ pixels, i.e. when more then 2% of all pixels are masked.

4.5 Bounding Box

To quantify how well attribution methods identify and localize the object of interest, we rely on bounding boxes available for the ImageNet dataset. Bounding boxes may contain irrelevant areas, in particular for non-rectangular objects. We restrict our evaluation to images with bounding boxes covering less than 33% of the input image. In total, we run the bounding box evaluation on 11,849 images from the ImageNet validation set.

If the bounding box contains n pixels, we measure how many of the n-th highest scored pixels are contained in the bounding box. By dividing by n, we obtain a ratio between 0 and 1. The results are shown in Table 1. The Per-Sample Bottleneck outperforms all baselines on both VGG-16 and ResNet-50 each by a margin of 0.152.

An alternative metric would be to take the sum of attribution in the bounding box and compare it to the total attribution in the image. We found this metric is not robust against extreme values. For the ResNet-50, we found basic Gradient Maps to be the best method as a few pixels receiving extreme scores are enough to dominate the sum.

4.6 Image Degradation

As a further quantitative evaluation, we rely on the degradation task as used by Ancona et al. (2017); Kindermans et al. (2018); Hooker et al. (2018); Samek et al. (2016). Given an attribution heatmap, the input is split in tiles, which are ranked by the sum of attribution values within each corresponding

Model & Evaluation	ResNet-50 deg.		VGG-16 deg.		ResNet	VGG
	8x8	14x14	8x8	14x14	bbox	bbox
Random	0.000	0.000	0.000	0.000	0.167	0.167
Occlusion-8x8	0.162	0.130	0.267	0.258	0.296	0.312
Occlusion-14x14	0.228	0.231	0.402	0.404	0.341	0.358
Gradient	0.002	0.005	0.001	0.005	0.259	0.276
Saliency	0.287	0.305	0.326	0.362	0.363	0.393
GuidedBP	0.491	0.515	0.460	0.493	0.388	0.373
PatternAttribution	–	–	0.440	0.457	–	0.404
LRP $\alpha=1, \beta=0$	–	–	0.471	0.486	–	0.397
LRP $\alpha=0, \beta=1, \epsilon=5$	–	–	0.462	0.467	–	0.441
Int. Grad.	0.401	0.424	0.420	0.453	0.372	0.396
SmoothGrad	0.485	0.502	0.438	0.455	0.439	0.399
Grad-CAM	0.536	0.541	0.510	0.517	0.465	0.399
GuidedGrad-CAM	0.565	**0.577**	0.555	0.576	0.468	0.419
IBA Per-Sample $\beta=1/k$	**0.573**	0.573	0.581	0.583	0.606	0.566
IBA Per-Sample $\beta=10/k$	0.572	0.571	**0.582**	**0.585**	**0.620**	**0.593**
IBA Per-Sample $\beta=100/k$	0.534	0.535	0.542	0.545	0.574	0.568
IBA Readout $\beta=10/k$	0.536	0.536	0.490	0.536	0.484	0.437

Table 1: *Degradation (deg.)*: Integral between LeRF and MoRF in the degradation benchmark for different models and window sizes over the ImageNet test set. *Bounding Box (bbox)*: the ratio of the highest scored pixels within the bounding box. For ResNet-50, we show no results for PatternAttribution and LRP as no PyTorch implementation supports skip-connections.

tile of the attribution. At each iteration, the highest-ranked tile is replaced with a constant value, the modified input is fed through the network, and the resulting drop in target class score is measured. A steep descent of the accuracy curve indicates a meaningful attribution map.

When implemented in the described way, the most relevant tiles are removed first (MoRF). However, Ancona et al. (2017) argues that using only the MoRF curve for evaluation is not sufficient. For the MoRF score, it is beneficial to find tiles disrupting the output of the neural network as quickly as possible. Neural networks are sensitive to subtle changes in the input (Szegedy et al., 2013). The tiles do not necessarily have to contain meaningful information to disrupt the network. Ancona et al. (2017) proposes to invert the degradation direction, removing tiles ranked as least relevant by the attribution method first (LeRF). The LeRF task favors methods that identify areas sufficient for classification. We scale the network's output probabilities to be in $[0, 1]$:

$$s(x) = \frac{p(y|x) - b}{t_1 - b}\,, \tag{9}$$

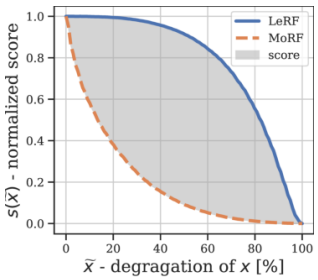


Figure 7: Mean MoRF and LeRF for the Per-Sample Bottleneck. The area is the final degradation score.

where t_1 is the model's average top-1 probability on the original samples and b is the mean model output on the fully degraded images. Both averages are taken over the validation set. A score of 1 corresponds to the original model performance.

Both LeRF and MoRF degradation yield curves as visualized in Figure 7, measuring different qualities of the attribution method. To obtain a scalar measure of attribution performance and to combine both metrics, we propose to calculate the integral between the MoRF and LeRF curves.

The results for all implemented attribution methods on the degradation task are given in Table 1. We evaluated both models on the full validation set using 8x8 and 14x14 tiles. In Appendix H, we show the mean LeRF and MoRF curves for 14x14 tiles. The Per-Sample Bottleneck outperforms all other methods in the degradation benchmark except for GuidedGrad-CAM on ResNet-50 where it scores comparably (score difference of 0.004). The Readout Bottleneck generally achieves a lower degradation scores but still perform competitively.

5 CONCLUSION

We propose two novel attribution methods that return an upper bound on the amount of information each input region provides for the network's decision. Our models' core functionality is a bottleneck layer used to inject noise into a given feature layer and a mechanism to learn the parametrized amount of noise per feature. The Per-Sample Bottleneck is optimized per single data point, whereas the Readout Bottleneck is trained on the entire dataset.

Our method does not constrain the internal network structure. In contrast to several modified backpropagation methods, it supports any activation function and network architecture. To evaluate our method, we extended the degradation task to quantify model performance deterioration when removing both relevant and irrelevant image tiles first. We also show results on how well ground-truth bounding boxes are scored. Our Per-Sample and Readout Bottleneck both show competitive results on all metrics used, outperforming state of the art with a significant margin for some of the tasks.

Generally, we would advise using the Per-Sample Bottleneck over the Readout Bottleneck. It performs better and is more flexible as it only requires to estimate the mean and variance of the feature map. The Readout Bottleneck has the advantage of producing attribution maps with a single forward pass once trained. Images with multiple object instances provide the network with redundant class information. The Per-Sample Bottleneck may discard some of the class evidence. Even for single object instances, the heatmaps of the Per-Sample Bottleneck may vary slightly due to the randomness of the optimization process.

The method's information-theoretic foundation provides a guarantee that the network does not require regions of zero-valued attribution for correct classification. To our knowledge, our attribution method is the only one to provide scores with units (bits). This absolute frame of reference allows a quantitative comparison between models, inputs, and input regions. We hope this contributes to a deeper understanding of neural networks and creates trust to use modern models in sensitive areas of application.

ACKNOWLEDGEMENT

We want to thank our anonymous reviewers for their valuable suggestions. In particular, we thank reviewer 1 for encouraging us to include the sanity checks. We are indebted to B. Wild, A. Elimelech, M. Granz, and D. Dormagen for helpful discussions and feedback. For LRP, we used the open source implementation by M. Böhle and F. Eitel (Böhle et al., 2019) and we thank A-K. Dombrowski for sharing an implementation and patterns for PatternAttribution with us. We thank Chanakya Ajit Ekbote for pointing out a mistake in equation 4. LS was supported by the Elsa-Neumann-Scholarship by the state of Berlin. We are also grateful to Nvidia for providing us with a Titan Xp and to ZEDAT for granting us access to their HPC system.

REFERENCES

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40 (12):2897–2905, 2018.

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

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *5th International Conference on Learning Representations (ICLR 2017)*, 2017.

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

Marco Ancona, Enea Ceolini, Cengiz Oztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations (ICLR 2018)*, 2018.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7), 2015.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

Moritz Böhle, Fabian Eitel, Martin Weygandt, and Kerstin Ritter. Layer-wise relevance propagation for explaining deep neural network decisions in mri-based alzheimer's disease classification. *Frontiers in Aging Neuroscience*, 11:194, 2019. ISSN 1663-4365. doi: 10.3389/fnagi.2019.00194.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, pp. 3581–3590, 2017.

Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pp. 1787–1796, 2018.

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

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating Feature Importance Estimates. *arXiv e-prints*, 2018.

Ashkan Khakzar, Soroosh Baselizadeh, Saurabh Khanduja, Seong Tae Kim, and Nassir Navab. Explaining neural networks via perturbing important learned features. *arXiv preprint arXiv:1911.11081*, 2019.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pp. 2668–2677, 2018.

Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. In *International Conference on Learning Representations*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pp. 971–980, 2017.

Matthias Kümmerer, Lucas Theis, and Matthias Bethge. Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet. *arXiv:1411.1045 [cs, q-bio, stat]*, 2014.

Jan Macdonald, Stephan Wäldchen, Sascha Hauch, and Gitta Kutyniok. A rate-distortion framework for explaining neural network decisions. *arXiv preprint arXiv:1905.11092*, 2019.

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

Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, pp. 3809–3818, 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5171–5180. PMLR, 2019.

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

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

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

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Leon Sixt, Maximilian Granz, and Tim Landgraf. When Explanations Lie: Why Many Modified BP Attributions Fail. *arXiv preprint arXiv:1912.09818*, 2019.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv:1706.03825 [cs, stat]*, 2017.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *arXiv e-prints*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328. JMLR. org, 2017.

Taiji Suzuki, Masashi Sugiyama, Jun Sese, and Takafumi Kanamori. Approximating mutual information by maximum likelihood density ratio estimation. In *New challenges for feature selection in data mining and knowledge discovery*, pp. 5–20, 2008.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Saeid Asgari Taghanaki, Mohammad Havaei, Tess Berthier, Francis Dutil, Lisa Di Jorio, Ghassan Hamarneh, and Yoshua Bengio. Infomask: Masked variational latent representation to localize chest disease. In Dinggang Shen, Tianming Liu, Terry M. Peters, Lawrence H. Staib, Caroline Essert, Sean Zhou, Pew-Thian Yap, and Ali Khan (eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, pp. 739–747, Cham, 2019. Springer International Publishing. ISBN 978-3-030-32226-7.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Tom Viering, Ziqi Wang, Marco Loog, and Elmar Eisemann. How to Manipulate CNNs to Make Them Lie: the GradCAM Case. *arXiv:1907.10901 [cs]*, August 2019. URL `http://arxiv.org/abs/1907.10901`.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: From error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612, 2004.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Andrey Zhmoginov, Ian Fischer, and Mark Sandler. Information-bottleneck approach to self-attention. In *ICML 2019 Workshop on Self-Supervised learning*, 2019.

A VISUAL COMPARISON OF ATTRIBUTION METHODS

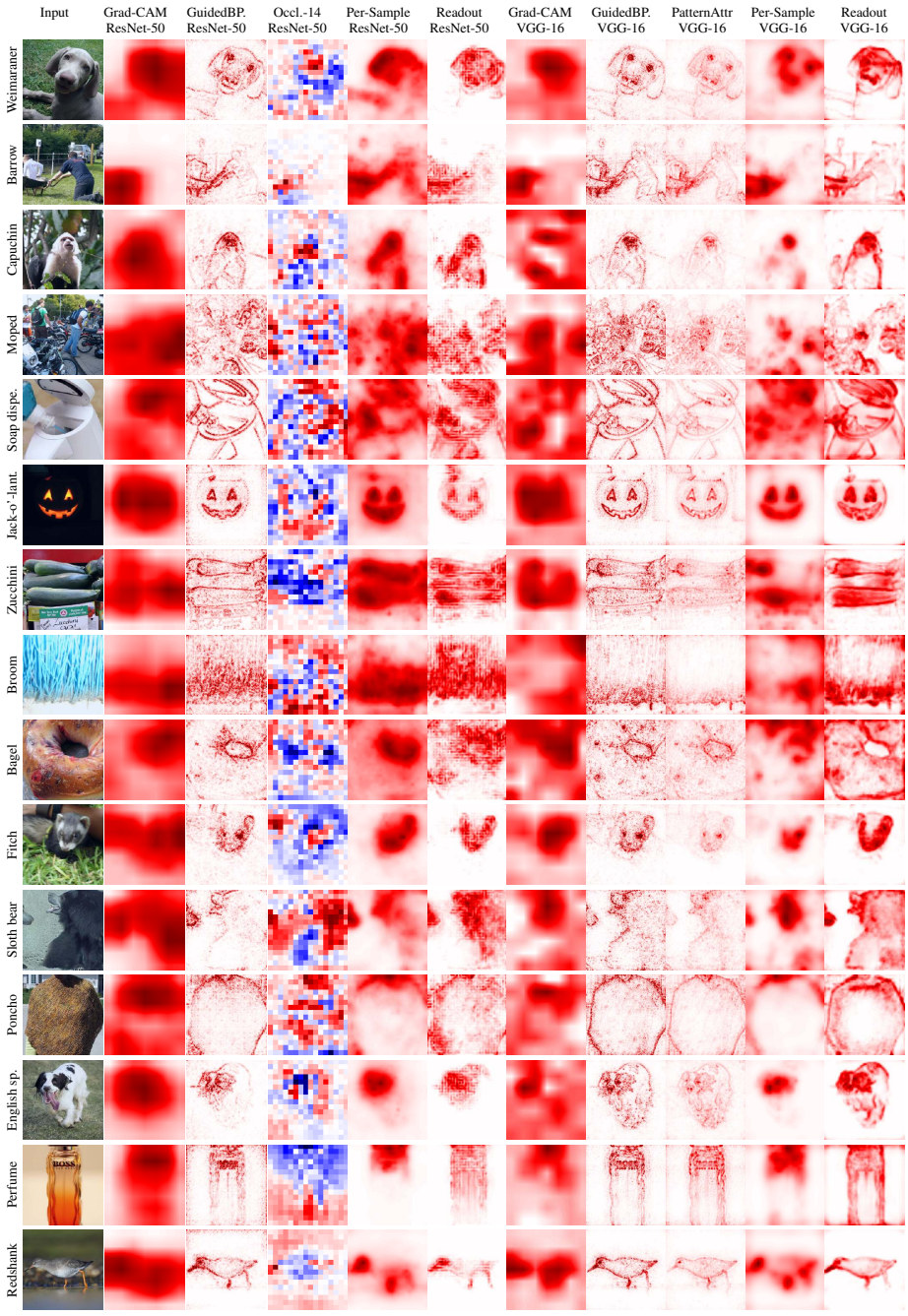


Figure 8: Blue indicates negative relevance and red positive. The authors promise that the samples were picked truly randomly, no cherry-picking, no lets-sample-again-does-not-look-nice-enough.

B GRID ARTIFACTS WHEN NOT USING SMOOTHING

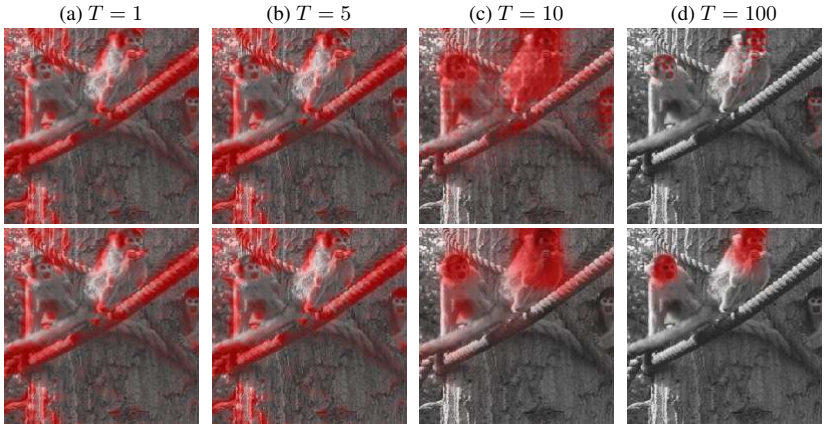


Figure 9: Development of $D_{\mathrm{KL}}(Q(Z|X)||Q(Z))$ of the Per-Sample Bottleneck for layer `conv1_3` of the ResNet-50. Red indicate areas with maximal information flow and semi-transparent green for zero information flow. Top row: without smoothing the mask exhibits a grid structure. Bottom row: smoothing with $\sigma_s = 2$. The smoothing both prevents the artifacts and reduces overfitting to small areas.

C EFFECTS OF DIFFERENT β AND LAYER DEPTH FOR THE VGG-16

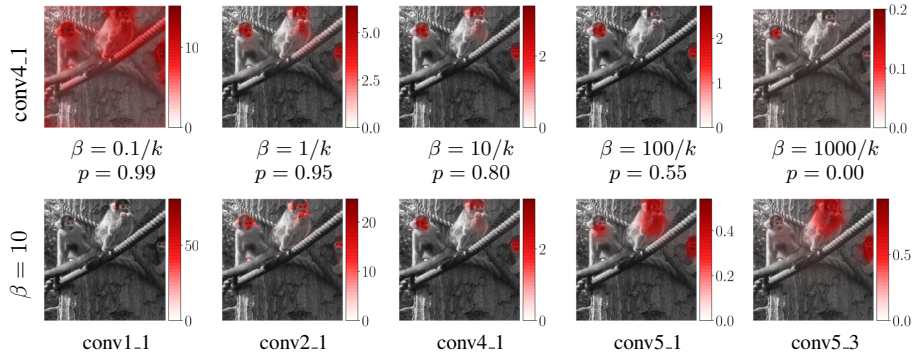


Figure 10: Effect of varying layer depth and β on the Per-Sample Bottleneck for the VGG-16. The resulting ouput probability for the correct class is given as p.

D Derivation of the Upper-bound of Mutual Information

For the mutual information $\mathrm{I}[R, Z]$, we have:

$$\begin{aligned}
\mathrm{I}[R, Z] &= \mathbb{E}_R[D_{\mathrm{KL}}[P(Z|R)||P(Z)]] && (10)\\
&= \int_R p(r) \left(\int_Z p(z|r) \log \frac{p(z|r)}{p(z)} dz \right) dr && (11)\\
&= \int_R \int_Z p(r, z) \log \frac{p(z|r)}{p(z)} \frac{q(z)}{q(z)} dz dr && (12)\\
&= \int_R \int_Z p(r, z) \log \frac{p(z|r)}{q(z)} dz dr + \int_R \int_Z p(r, z) \log \frac{q(z)}{p(z)} dz dr && (13)\\
&= \int_R \int_Z p(r, z) \log \frac{p(z|r)}{q(z)} dz dr + \int_Z p(z) \left(\int_R p(r|z) dr \right) \log \frac{q(z)}{p(z)} dz && (14)\\
&= \mathbb{E}_R \left[D_{\mathrm{KL}}[P(Z|R)||Q(Z)] \right] - D_{\mathrm{KL}}[P(Z)||Q(Z)] && (15)\\
&\leq \mathbb{E}_R \left[D_{\mathrm{KL}}[P(Z|R)||Q(Z)] \right] && (16)
\end{aligned}$$

E Mean and variance of Z

The $\lambda(X)$ linearly interpolate between the feature map R and the noise $\epsilon \sim \mathcal{N}(\mu_R, \sigma_R^2)$, where μ_R and σ_R are the estimated mean and standard derivation of R. Both $R = f_l(X)$ and $\lambda(X)$ depend on the input random variable X.

$$Z = \lambda(X) R + (1 - \lambda(X)) \epsilon \tag{17}$$

For the mean of Z, we have:

$$\begin{aligned}
\mathbb{E}[Z] &= \mathbb{E}[\lambda(X)R] + \mathbb{E}[(1 - \lambda(X))\epsilon] && \triangleright \text{ substituting in definition of } Z\\
&= E[\lambda(X)R] + \mathbb{E}[1 - \lambda(X)]\mathbb{E}[\epsilon] && \triangleright \text{ independence of } \lambda \text{ and } \epsilon\\
&= \mathrm{cov}(\lambda(X), R) + \mathbb{E}[\lambda(X)]\mathbb{E}[R] + \mathbb{E}[1 - \lambda(X)]\mathbb{E}[\epsilon] && \triangleright \ \mathrm{cov}(A, B) = \mathbb{E}[AB] - \mathbb{E}[A]\mathbb{E}[B]\\
&\approx \mathrm{cov}(\lambda(X), R) + \mathbb{E}[\epsilon] && \triangleright \ \mathbb{E}[\epsilon] = \mu_R \approx \mathbb{E}[R]
\end{aligned}$$

As $\lambda(X)$ and R are multiplied together, they form a complex product distribution. If they do not correlate, $\mathbb{E}[Z] \approx \mathbb{E}[\epsilon] \approx \mathbb{E}[R]$.

A similar problem araises for the variance:

$$\begin{aligned}
\mathrm{Var}[Z] &= \mathbb{E}[Z^2] - \mathbb{E}[Z]^2\\
&= \mathbb{E}[(\lambda(X)R + (1 - \lambda(X)\epsilon)^2] - (\mathrm{cov}(\lambda(X), R) + \mathbb{E}[\epsilon])^2
\end{aligned}$$

The multiplication of $\lambda(X)$ and R causes in general the variance of Z and R to not match: $\mathrm{Var}[Z] \neq \mathrm{Var}[R]$.

F Hyperparameters

Parameter	ResNet-50	VGG-16	Search space
Target layer	`conv3_4`	`conv4_1`	
Optimizer	Adam (Kingma & Ba (2014))		
Learning Rate	$\eta = 1$		$\{0.03, 0.1, 0.3, 1, 3, 10\}$
Balance Factor	$\beta = 10/k$		$\{0.001, 0.01, 0.1, 1, 10, 100, 300\}$
Iterations	$T = 10$		$\{1, 3, 5, 10, 30, 100\}$
Batch Size	$B = 10$		$\{1, 5, 10, 30\}$
Smoothing	$\sigma_s = 1$		$\{0.5, 1, 2\}$

Table 2: Hyperparameters for Per-Sample Bottleneck. The layer notations for the ResNet-50 are taken from the original publication (He et al., 2016). The first index denotes the block and the second the layer within the block. For the VGG-16, `conv_n` denotes the n-th convolutional layer.

		Orig.	Initial	$\beta{=}0.01/k$	$0.1/k$	$1/k$	$10/k$	$100/k$
Per-Sample	$\mathcal{L}_I/k$	–	2.500	3.174	0.525	0.248	0.098	0.019
	Top-5 Acc.	0.928	0.928	1.000	0.971	1.000	0.990	0.522
	Top-1 Acc.	0.760	0.760	1.000	0.963	1.000	0.984	0.427
Readout	$\mathcal{L}_I/k$	–	2.500	1.822	0.628	0.222	0.079	0.023
	Top-5 Acc.	0.928	0.928	0.930	0.928	0.917	0.870	0.505
	Top-1 Acc.	0.760	0.760	0.761	0.756	0.735	0.660	0.302

Table 4: Influence of β on the information loss $\mathcal{L}_I$ and the test accuracy on ResNet-50. k is the size of the feature map, i.e. $k = hwc$. *Initial*: Configuration of the untrained bottleneck with $\alpha = 5$. *Original*: Values for the original model without the bottleneck.

Parameter	ResNet-50	VGG-16	Search space
Target layer	`conv2_3`	`conv3_1`	
Reading out	`conv2_3`	`conv3_1`	
	`conv3_4`	`conv4_1`	
	`conv4_6`	`conv5_1`	
	`conv5_3`	`conv5_3`	
	`fc`	`fc`	
Optimizer	Adam (Kingma & Ba (2014))		
Learning Rate	$\eta = 10^{-5}$		{e-4, e-5, e-6}
Balance Factor	$\beta = 10/k$		$\{0.1/k, 1/k, 10/k, 100/k\}$
Epochs	$E = 10$		
Batch Size	$B = 16$		
Smoothing	$\sigma_s = 1$		

Table 3: Hyperparameters for the Readout Bottleneck.

In Table 2, we provide hyperparameters for the Per-Sample Bottleneck. In Table 3, we provide hyperparameters for the Readout Bottleneck.

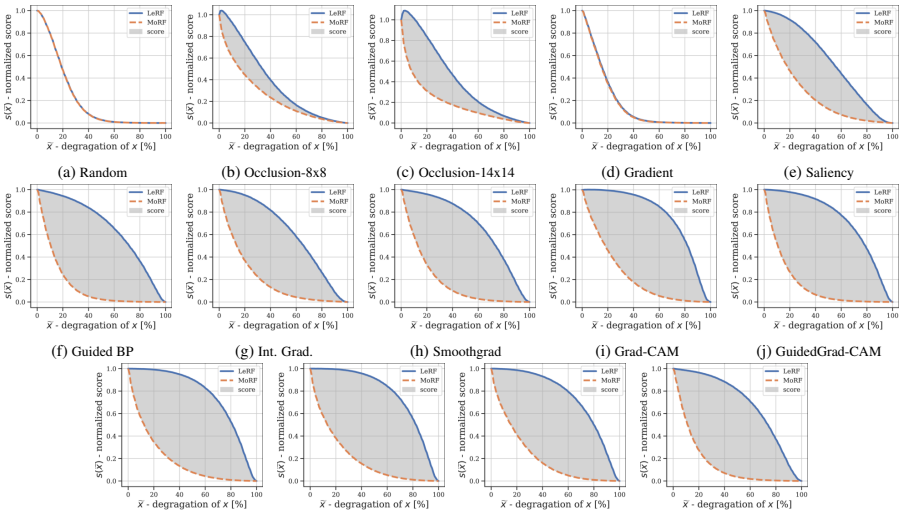
In Table 4, a comparison of pre- to post-training accuracy and the estimated mutual information is shown for both Bottleneck types. The Per-Sample Bottleneck can learn to suppress negative evidence for each sample and the accuracy is close to 1 for $\beta \leq 10/k$. For $\beta = 100/k$, also positive evidence is discarded and the accuracy decreases to 0.43. The Readout Bottleneck learns to suppress negative evidence for small $\beta \leq 0.1/k$ and slightly improves the final accuracy.

G Evaluation Metrics

G.1 Sanity Check: Weight Randomization

We use the cascading parameter randomization sanity check from Adebayo et al. (2018). Following the original paper, we used the skimage SSIM implementation with a window of size 5. For LRP, we found that the weight randomization flips the values of the saliceny heatmap, e.g. $h_r \approx -h_o$ where h_r is the heatmap with random weights and h_o the heatmap on the original model. Therefore for LRP, we used: $\max(\text{ssim}(h_o, \text{normalize}(h_r), \text{ssim}(h_o, \text{normalize}(-h_r)))$. We normalized the heatmaps by first clamping the 1-th and 99-th percentile and then rescaling the heatmap it to $[0, 1]$. We run the sanity check on 200 randomly selected samples from the ImageNet valdiation set.

H MoRF and LeRF Degradation Paths

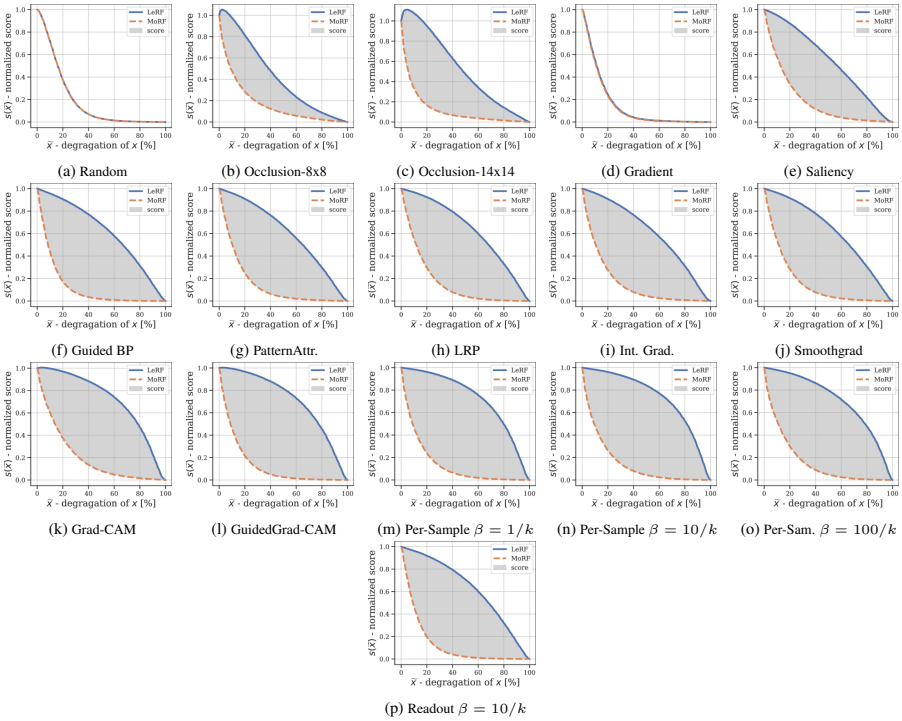


(a) Random (b) Occlusion-8x8 (c) Occlusion-14x14 (d) Gradient (e) Saliency

(f) Guided BP (g) Int. Grad. (h) Smoothgrad (i) Grad-CAM (j) GuidedGrad-CAM

(k) Per-Sample $\beta = 1/k$ (l) Per-Sample $\beta = 10/k$ (m) Per-Sam. $\beta = 100/k$ (n) Readout $\beta = 10/k$

Figure 11: MoRF and LeRF for the ResNet-50 network using 14x14 tiles.



(a) Random (b) Occlusion-8x8 (c) Occlusion-14x14 (d) Gradient (e) Saliency

(f) Guided BP (g) PatternAttr. (h) LRP (i) Int. Grad. (j) Smoothgrad

(k) Grad-CAM (l) GuidedGrad-CAM (m) Per-Sample $\beta = 1/k$ (n) Per-Sample $\beta = 10/k$ (o) Per-Sam. $\beta = 100/k$

(p) Readout $\beta = 10/k$

Figure 12: MoRF and LeRF paths for the VGG-16 network using 14x14 tiles.

4 When Explanations Lie: Why Many Modified BP Attributions Fail

As outlined in the introduction, methods in the field of explainable AI have to make simplifications and assumptions about the model. While this is necessary to create comprehensible explanations, it can also create explanations misrepresenting the actual model. For example, in Figure 1 of the following paper, the saliency maps for the evidence for the cat and dog classes seem identical. It is confusing for observers to see the evidence for the two different classes being identical. As scientists, we should analyze why this perplexing behavior occurs.

The following publication was the direct follow-up work from the sanity check experiment performed in (Schulz et al. 2020, Section 4.3). In the sanity check experiment, we randomized the weights of a neural network partially and tested whether the attribution methods change their explanations. In this experiment, we found that explanations from the layer-wise relevance propagation (LRP) (Bach et al. 2015) were independent of the network's deeper layers. In addition to LRP, we also found that other modified backpropagation show similar perplexing behaviors: Deep Taylor Decomposition (DTD), PatternAttribution, Excitation BP, Deconv, GuidedBP, and RectGrad (Bach et al. 2015; Montavon et al. 2017; Kindermans et al. 2018b; Zhang et al. 2018; Zeiler et al. 2014; Springenberg et al. 2014; Kim et al. 2019). Of all tested propagation-based methods, only DeepLIFT passed the sanity check experiment (Shrikumar et al. 2017b). As we found out through an ablation study, the reason for this is that DeepLIFT considers both positive and negative gradients.

All these methods modify the backpropagation algorithm in different ways. For example, Guided Backpropagation (Springenberg et al. 2014) applied a rectifier function to the gradients after each layer – a simple but heuristic approach. Another method, PatternAttribution (Kindermans et al. 2018b), estimates a signal direction for each layer, which is then used to perform a Taylor expansion per layer.

For $\text{LRP}_{\alpha 1 \beta 0}$ (the particular LRP rule tested), we found that a chain of positive matrices causes the problem, as this chain converges to a rank-1 matrix. The characteristic property of a rank-1 matrix M is that $M\boldsymbol{v} = \lambda \boldsymbol{m}$, i.e., the resulting output is a scaled version of a fixed vector $\boldsymbol{m}$. A rank-1 matrix has only a single degree of freedom, which is even lost for saliency maps as they are usually normalized. We also proved that the chain converges exponentially fast under certain conditions, which are commonly satisfied for fully connected layers.

The following paper reveals that many established methods produce explanations that are not reliable, as they are independent of the explained class and, as a result, fail to provide meaningful explanations. We have not only demonstrated this fact empirically, but we have also provided theoretical reasoning for why it occurs.

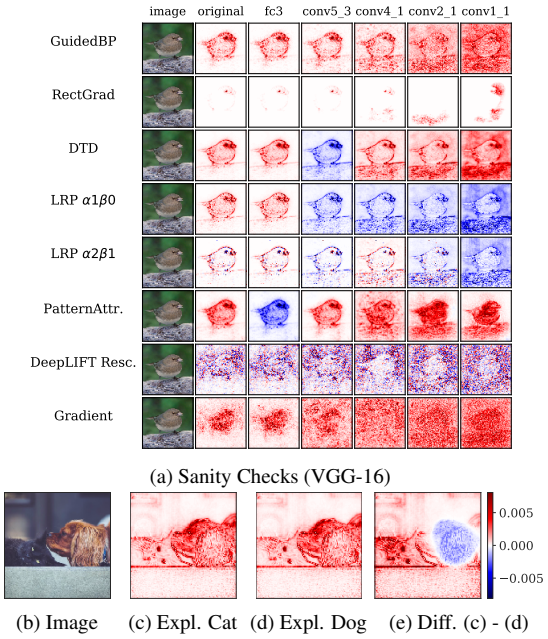
Our work served as a wake-up call for numerous research groups working on attribution methods, as it shows how easily one can be deceived by saliency maps. Our findings highlight the need to be aware of confirmation bias, which can lead researchers to erroneously accept or prioritize information that reinforces their preexisting beliefs, while neglecting opposing evidence. The field of explainable AI must address these concerns in order to develop more reliable and accurate explanation methods.

When Explanations Lie: Why Many Modified BP Attributions Fail

Leon Sixt [1] **Maximilian Granz** [1] **Tim Landgraf** [1]

Abstract

Attribution methods aim to explain a neural network's prediction by highlighting the most relevant image areas. A popular approach is to backpropagate (BP) a custom relevance score using modified rules, rather than the gradient. We analyze an extensive set of modified BP methods: Deep Taylor Decomposition, Layer-wise Relevance Propagation (LRP), Excitation BP, PatternAttribution, DeepLIFT, Deconv, RectGrad, and Guided BP. We find empirically that the explanations of all mentioned methods, except for DeepLIFT, are independent of the parameters of later layers. We provide theoretical insights for this surprising behavior and also analyze why DeepLIFT does not suffer from this limitation. Empirically, we measure how information of later layers is ignored by using our new metric, cosine similarity convergence (CSC). The paper provides a framework to assess the faithfulness of new and existing modified BP methods theoretically and empirically. [2]



(a) Sanity Checks (VGG-16)

(b) Image (c) Expl. Cat (d) Expl. Dog (e) Diff. (c) - (d)

Figure 1: **(a)** Sanity Checks: Saliency maps should change if network parameters are randomized. Parameters are randomized from the last to the first layer. *Red* denotes positive and *blue* negative relevance. **(b-e)** Class insensitivity of $\text{LRP}_{\alpha 1 \beta 0}$ on VGG-16. Explanation for **(c)** *Persian cat (283)* and **(d)** *King Charles Spaniel (156)*. **(e)** Difference (c) - (d), both normalized to $[0, 1]$. L1-norm of (e) = 0.000371.

1. Introduction

Explainable AI (XAI) aims to improve the interpretability of machine learning models. For deep convolutional networks, attribution methods visualize the areas relevant for the prediction with so-called saliency maps. Various attribution methods have been proposed, but do they reflect the model behavior correctly?

(Adebayo et al., 2018) proposed a sanity check: if the parameters of the model are randomized and therefore the network output changes, do the saliency maps change too? Surprisingly, the saliency maps of GuidedBP (Springenberg et al., 2014) stay identical, when the last layer (fc3) is randomized (see Figure 1a). A method ignoring the last layer can *not* explain the network's prediction faithfully.

In addition to (Adebayo et al., 2018), which only reported GuidedBP to fail, we found several modified backpropagation (BP) methods fail too: Layer-wise Relevance Propagation (LRP), Deep Taylor Decomposition (DTD), PatternAttribution, Excitation BP, Deconv, GuidedBP, and RectGrad (Bach et al., 2015; Montavon et al., 2017; Kindermans et al., 2018; Zhang et al., 2018; Zeiler & Fergus, 2014; Springenberg et al., 2014; Kim et al., 2019). The only tested modified BP method passing is DeepLIFT (Shrikumar et al., 2017).

[1]Dahlem Center of Machine Learning and Robotics, Freie Universität Berlin, Germany. Correspondence to: Leon Sixt <leon.sixt@fu-berlin.de>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[2] For code see: github.com/berleon/when-explanations-lie

Modified BP methods estimate relevant areas by backpropagating a custom relevance score instead of the gradient. For example, DTD only backpropagates positive relevance scores. Modified BP methods are popular with practitioners (Yang et al., 2018; Sturm et al., 2016; Eitel et al., 2019). For example, (Schiller et al., 2019) uses saliency maps to improve the classification of whale sounds or (Böhle et al., 2019) use LRP$_{\alpha 1 \beta 0}$ to localize evidence for Alzheimer's disease in brain MRIs.

Deep neural networks are composed of linear layers (dense, conv.) and non-linear activations. For each linear layer, the weight vector reflects the importance of each input directly. (Bach et al., 2015; Kindermans et al., 2018; Montavon et al., 2017) argue that aggregating explanations of each linear model can explain a a deep neural network. Why do these methods then fail the sanity check?

Theoretically, we show that the z^+-rule – used by DTD, LRP$_{\alpha 1 \beta 0}$, and Excitation BP – yields a multiplication chain of non-negative matrices. Each matrix corresponds to a layer. The saliency map is a function of this matrix chain. We show that such a non-negative matrix chain converges to a rank-1 matrix. If $C \in \mathbb{R}^{n \times m}$ is a rank-1 matrix, then it can be written as an outer product $C = \boldsymbol{c}\boldsymbol{\gamma}^T$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{\gamma} \in \mathbb{R}^m$. Multiplying C with any vector $\boldsymbol{v}$ yields always the same the direction: $C\boldsymbol{v} = \boldsymbol{c}\boldsymbol{\gamma}^T\boldsymbol{v} = \lambda \boldsymbol{c}, \lambda \in \mathbb{R}$. The scaling is irrelevant as saliency maps are normalized. If sufficiently converged, the backpropagated vector can merely switch the sign of the saliency map. For example, in Figure 1a, the sign of the PatternAttribution saliency map switches due to the randomization of fc3. Figure 1b-1e show how the saliency maps of LRP$_{\alpha 1 \beta 0}$ become class-insensitive.

Empirically, we quantify the convergence to a rank-1 matrix using our novel cosine similarity convergence (CSC) metric. CSC allows to retrace, layer by layer, how modified BP methods lose information about previous layers. Using CSC, we observe that all analyzed modified BP methods, except for DeepLIFT, converge towards a rank-1 matrix on VGG-16 and ResNet-50. For sufficiently large values of α and β, LRP$_{\alpha\beta}$ does not converge but also produces rather noisy saliency maps.

The paper focuses on modified BP methods, as other attribution methods do not suffer from the converges problem. They either rely on the gradient directly (Smilkov et al., 2017; Sundararajan et al., 2017), which does not converge or consider the model as a black-box (Ribeiro et al., 2016; Lundberg & Lee, 2017).

Our findings show that many modified BP methods are prone to class-insensitive explanations and provide saliency maps that rather highlight low-level features. Negative relevance scores are crucial to avoid the convergence to a rank-1 matrix — a possible future research direction.

2. Theoretical Analysis

Notation For our theoretical analysis, we consider feed-forward neural networks with a ReLU activation function $[\boldsymbol{x}]^+ = \max(0, \boldsymbol{x})$. The neural network $f(\boldsymbol{x})$ contains n layers, each with weight matrices W_l. The output of the l-th layer is denoted by $\boldsymbol{h}_l$. We use $[ij]$ to index the i, j element in W_l as in $W_{l_{[ij]}}$. To simplify notation, we absorb the bias terms into the weight matrix, and we omit the final softmax layer. We refer to the input with $\boldsymbol{h}_0 = \boldsymbol{x}$ and to the output with $\boldsymbol{h}_n = f(\boldsymbol{x})$. The output of the l-th layer is given by:

$$\boldsymbol{h}_l = [W_l \boldsymbol{h}_{l-1}]^+ \tag{1}$$

All the results apply to convolutional neural networks as convolution can be expressed as matrix multiplication.

Gradient The gradient of the k-th output of the neural network w.r.t. the input $\boldsymbol{x}$ is given by:

$$\frac{\partial f_k(\boldsymbol{x})}{\partial \boldsymbol{x}} = W_1^T M_1 \frac{\partial f_k(\boldsymbol{x})}{\partial \boldsymbol{h}_1} = \prod_l^n \left(W_l^T M_l\right) \cdot \boldsymbol{v}_k, \tag{2}$$

where $M_l = \operatorname{diag}(1_{h_l > 0})$ denotes the gradient mask of the ReLU operation. The last equality follows from recursive expansion. The vector $\boldsymbol{v}_k$ is a one-hot vector to select the k-th output.

The gradient of residual blocks is also a product of matrices. The gradient of $\boldsymbol{h}_{l+1} = \boldsymbol{h}_l + g(\boldsymbol{h}_l)$ is:

$$\frac{\partial \boldsymbol{h}_{l+1}}{\partial \boldsymbol{h}_l} = I + G_{\partial g(\boldsymbol{h}_l)/\partial \boldsymbol{h}_l}, \tag{3}$$

where $G_{\partial g(\boldsymbol{h}_l)/\partial \boldsymbol{h}_l}$ denotes the derivation matrix of the residual block, and I is the identity matrix. For the gradient, the final saliency map is usually obtained by summing the absolute channel values of the relevance vector $r_0^{\nabla}(\boldsymbol{x})$ of the input layer.

The following methods modify the gradient definition and to distinguish the rules, we introduce the notation: $r_l^{\nabla}(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{h}_l}$ which denotes the relevance at layer l for an input $\boldsymbol{x}$.

Interpretability of Linear Models The relevance of the input of a linear model can be calculated directly. Let $y = \boldsymbol{w}^T\boldsymbol{x}$ be a linear model with a single output scalar. The relevance of the input $\boldsymbol{x}$ to the i-th output $\boldsymbol{y}_{[i]}$ is :

$$r_{\boldsymbol{x}}^{\text{Linear}}(\boldsymbol{x}) = \boldsymbol{w} \odot \boldsymbol{x}. \tag{4}$$

2.1. z^+-Rule

The z^+-rule is used by DTD (Montavon et al., 2017), Excitation BP (Zhang et al., 2018) and also corresponds to the LRP$_{\alpha 1 \beta 0}$ rule (Bach et al., 2015). The z^+-rule back-propagates positive relevance values, which are supposed to

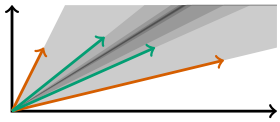


Figure 2: The positive column vectors $\boldsymbol{a}_1, \boldsymbol{a}_2$ of matrix A_1 (orange) form a cone. The resulting columns of $A_1 A_2$ (green) are contained in the cone as they are positive linear combinations of $\boldsymbol{a}_1, \boldsymbol{a}_2$. At each iteration, the cone shrinks.

correspond to the positive evidence for the prediction. Let w_{ij} be an entry in the weight matrix W_l:

$$r_l^{z^+}(\boldsymbol{x}) = Z_l^+ \cdot r_{l+1}^{z^+}(\boldsymbol{x})$$
$$\text{where } {Z_l^+}^T = \left(\frac{[w_{ij}\boldsymbol{h}_{l_{[j]}}]^+}{\sum_k [w_{ik}\boldsymbol{h}_{l_{[k]}}]^+} \right)_{[ij]} \tag{5}$$

Each entry in the derivation matrix Z_l^+ is obtained by measuring the positive contribution of the input neuron i to the output neuron j and normalizing by the total contributions to neuron j. The relevance from the previous layer $r_{l+1}^{z^+}$ is then distributed according to Z_l^+. The relevance function $r_l^{z^+} : \mathbb{R}^n \mapsto \mathbb{R}^m$ maps input $\boldsymbol{x}$ to a relevance vector of layer l. For the final layer the relevance is set to the value of the explained logit value, i.e. $r_n^{z^+}(\boldsymbol{x}) = f_k(\boldsymbol{x})$. In contrast to the vanilla backpropagation, algorithms using the z^+-rule do not apply a mask for the ReLU activation.

The relevance of multiple layers is computed by applying the z^+-rule to each of them. Similar to the gradient, we obtain a product of non-negative matrices: $C_k = \prod_l^k Z_l^+$.

Theorem 1. *Let $A_1, A_2, A_3 \ldots$ be a sequence of non-negative matrices. We require that every column vector $\boldsymbol{a}$ of A_n has a norm $||\boldsymbol{a}|| \geq \epsilon_0$ and that infinite many matrices A_i with $i \in I$ and $|I| = |\mathbb{N}|$ exists for which two column vectors have a dot product of at least $\epsilon_{\langle\cdot,\cdot\rangle}$, i.e. $\langle \boldsymbol{a}, \boldsymbol{b} \rangle \geq \epsilon_{\langle\cdot,\cdot\rangle}$, where both $\epsilon_0, \epsilon_{\langle\cdot,\cdot\rangle} > 0$. Then the product of all terms of the sequence converges to a rank-1 matrix $\bar{C}$:*

$$\bar{C} := \lim_{n\to\infty} \prod_{i=1}^{n} \frac{A_i}{||\prod_{i=1}^{n} A_i||} = \bar{\boldsymbol{c}}\boldsymbol{\gamma}^T \,. \tag{6}$$

(Hajnal, 1976; Friedland, 2006) proved a similar result for squared matrices. In appendix A, we provide a rigorous proof of the theorem using the cosine similarity.

The geometric intuition of the proof is depicted in Figure 2. The column vectors of the first matrix are all non-negative and therefore in the positive quadrant. For the matrix multiplication $A_i A_j$, observe that $A_i \boldsymbol{a}_k$ is a non-negative linear combination of the column vectors of A_i, where $\boldsymbol{a}_k$ is the k-th column vector $A_{j_{[:k]}}$. The result will remain in the convex cone of the column vectors of A_i. The conditions stated in the theorem ensure that the cone shrinks with every iteration and it converges towards a single vector. In the appendix **??**, we simulate different matrix properties and find non-negative matrices to converge exponentially fast.

The column vectors of a rank-1 matrix are linearly dependent $C = \boldsymbol{c}\boldsymbol{\gamma}^T$. A rank-1 matrix C always gives the same direction of $\boldsymbol{c}$: $CZ_{k+1}^+ = \boldsymbol{c}\boldsymbol{\gamma}^T Z_{k+1}^+ = \boldsymbol{c}\boldsymbol{\lambda}^T$ and for any vector $\boldsymbol{v}$: $CZ_{k+1}^+ \boldsymbol{v} = \boldsymbol{c}\boldsymbol{\lambda}^T \boldsymbol{v} = t\boldsymbol{c}$, where $t \in \mathbb{R}$. For a finite number of matrices $C_k = \prod_l^k Z_l^+$, C_k might resemble a rank-1 matrix up to floating-point imprecision or $C_k Z_{k+1}^+$ might still be able to alter the direction. In any case, the influence of later matrices decreases.

The Z^+ matrices of dense layers fulfill the conditions of theorem 1. Convolutions can be written as matrix multiplications. For 1x1 convolutions, the kernels do not overlap and the row vectors corresponding to each location are orthogonal. In this case, the convergence happens only locally per feature map location. For convolutions with overlapping kernels, the global convergence is slower than for dense layers. In a ResNet-50 where the last convolutional stack has a size of (7x7), the overlapping of multiple (3x3) convolutions still induces a considerable global convergence (see LRP_{CMP} on ResNet-50 in section 5).

If an attribution method converges, the contributions of the layers shrink by depth. In the worst-case scenario, when converged up to floating-point imprecision, the last layer can only change the scaling of the saliency map. However, the last layer is responsible for the network's final prediction.

2.2. Modified BP algorithms

LRP_z The LRP_z rule of Layer-wise Relevance Propagation modifies the backpropagation rule as follows:

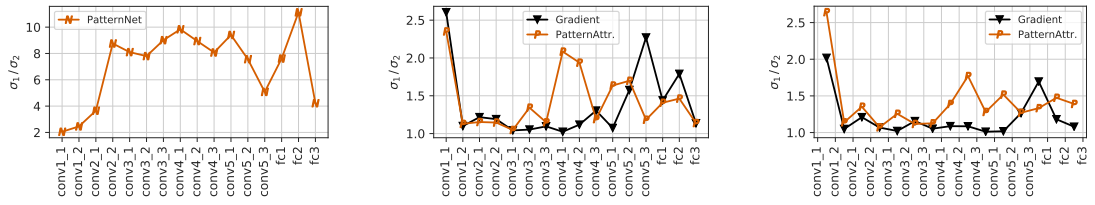
$$r_l^{z-\text{LRP}}(\boldsymbol{x}) = Z_l \cdot r_{l+1}^{z-\text{LRP}}(\boldsymbol{x}),$$
$$\text{where } Z_l = \left(\frac{w_{ij}\boldsymbol{h}_{l_{[j]}}}{\sum_k w_{ik}\boldsymbol{h}_{l_{[k]}}} \right)^T_{[ij]} . \tag{7}$$

If only max-pooling, linear layers, and ReLU activations are used, it was shown that LRP_z corresponds to gradient⊙input, i.e. $r_0^{z-\text{LRP}}(\boldsymbol{x}) = \boldsymbol{x} \odot \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$ (Shrikumar et al., 2016; Kindermans et al., 2016; Ancona et al., 2017). LRP_z can be considered a gradient-based and not a modified BP method. The gradient is not converging to a rank-1 matrix and therefore gradient⊙input is also not converging.

$\text{LRP}_{\alpha\beta}$ separates the positive and negative influences:

$$r_l^{\alpha\beta}(\boldsymbol{x}) = \left(\alpha Z_l^+ - \beta Z_l^-\right) r_{l+1}^{\alpha\beta}(\boldsymbol{x}), \tag{8}$$

where Z_l^+ and Z_l^- correspond to the positive and negative entries of the matrix Z. (Bach et al., 2015) propose to



(a) PatternNet (b) $W = U_l \Sigma_l V_l$ (c) $T_l = \sqrt{\Sigma_l} V_l U_{l+1} \sqrt{\Sigma_{l+1}}$

Figure 3: PatternNet & PatternAttr.: **(a)(b)** Ratio between the first and second singular value σ_1/σ_2 for A_l, W_l, and $A_l \odot W_l$. **(c)** σ_1/σ_2 of inter-layer derivation matrices. For (b) (c), we sliced the 3x3 convolutional kernels to 1x1 kernels.

weight positives more: $\alpha \geq 1$ and $\alpha - \beta = 1$. For LRP$_{\alpha 1 \beta 0}$, this rule corresponds to the z^+-rule, which converges. For $\alpha > 1$ and $\beta > 0$, the matrix $Z_l = \alpha Z_l^+ - \beta Z_l^-$ can contain negative entries. Our empirical results show that LRP$_{\alpha\beta}$ still converges for the most commonly used parameters $\alpha = 2, \beta = 1$ and even for a higher $\alpha = 5$ it converges considerable on the ResNet-50.

Deep Taylor Decomposition uses the z^+-rule if the input to a convolutional or dense layer is in $[0, \infty]$, i.e. if the layer follows a ReLU activation. For inputs in $\mathbb{R}$, DTD also proposed the w^2-rule and the so-call $w^{\mathcal{B}}$ rule for bounded inputs. Both rules were specifically designed to produce non-negative outputs. Theorem 1 applies and DTD converges to a rank-1 matrix necessarily.

PatternNet & PatternAttribution takes into account that the input $\boldsymbol{h}_l$ contains noise. If $\boldsymbol{d}_l$ corresponds to the noise and $\boldsymbol{s}_l$ to the signal, than $\boldsymbol{h}_l = \boldsymbol{s}_l + \boldsymbol{d}_l$. To assign the relevance towards the signal direction, it is estimated using the following equation:

$$\boldsymbol{a}_i = \frac{\text{cov}[\boldsymbol{h}]\boldsymbol{w}_i}{\boldsymbol{w}_i^T \text{cov}[\boldsymbol{h}]\boldsymbol{w}_i}, \tag{9}$$

where a_i is the estimated signal direction for the $i-th$ neuron with input $\boldsymbol{h}$ and weight vector $\boldsymbol{w}_i = W_{[i:]}$. PatternNet is designed to recover the relevant signal in the data. Let $A_{l[i:]} = \boldsymbol{a}_i$ be the corresponding signal matrix to the weight matrix W_l, the rule for PatternNet is:

$$r_l^{\text{PN}}(\boldsymbol{x}) = A_l^T \cdot r_{l+1}^{\text{PN}}(\boldsymbol{x}), \tag{10}$$

PatternNet is also prone to converge to a rank-1 matrix. To recover the relevant signal, it might be even desired to converge to the a single direction – the signal direction.

The convergence of PatternNet follows from the computation of the pattern vectors $\boldsymbol{a}_i$ in equation 9. It is similar to a single step of the power iteration method $\boldsymbol{v}_{k+1} = C\boldsymbol{v}_k / \|C\boldsymbol{v}_k\|$. In appendix C, we provide details on the relationship to power iteration and also derive equation 9 from the equation given in (Kindermans et al., 2018). The power iteration method converges to the eigenvector with the largest eigenvalue exponentially fast.

All column vectors in $A_{[i:]} = \boldsymbol{a}_i$ underwent a single step of the power iteration and therefore tend to point towards the first eigenvector of $\text{cov}[\boldsymbol{h}]$. This can also be verified empirically: the ratio of the first and second singular value $\sigma_1(A)/\sigma_2(A) > 6$ for almost all the VGG-16 patterns (see Figure 3a), indicating a strong convergence of the matrix chain towards a single direction.

The findings from PatternNet are hard to transfer to PatternAttribution. The rule for PatternAttribution uses the Hadamard product of A_l and W_l:

$$r_l^{\text{PA}}(\boldsymbol{x}) = (W_l \odot A_l)^T \cdot r_{l+1}^{\text{PA}}(\boldsymbol{x}), \tag{11}$$

The Hadamard product complicates any analytic argument using the properties of A_l or W_l. The theoretical results available (Ando et al., 1987; Zhan, 1997) did not allow us to show that PatternAttribution converges to a rank-1 matrix necessarily.

We provide a mix of theoretical and empirical insights on why it converges. The conditions of convergence can be studied well on the singular value decomposition: $(W_l \odot A_l)^T = U_l \Sigma_l V_l$. Loosely speaking, the matrix chain will converge to a rank-1 matrix if the first σ_1 and second σ_2 singular values in Σ_l differ and if V_l and U_{l+1} are aligned such that higher singular values of Σ_l and Σ_{l+1} are multiplied together such that the ratio σ_1/σ_2 grows.

To see how well the per layer matrices align, we look at the inter-layer chain members: $T_l = \sqrt{\Sigma_l} V_l U_{l+1} \sqrt{\Sigma_{l+1}}$. In Figure 3, we display the ratio between the first and second singular values $\sigma_1(T_l)/\sigma_2(T_l)$. For $W \odot A$, the first singular value is considerably larger than for the plain weights W. Interestingly, the singular value ratio of inter-layer matrices shrinks for the plain W matrix. Whereas for PatternAttribution, the ratio increases for some layers indicating that the Hadamard product leads to more alignment of the matrices.

DeepLIFT is the only tested modified BP method which does not converge to a rank-1 matrix. It is an extension of the backpropagation algorithm to finite differences:

$$\frac{f(\boldsymbol{x}) - f(\boldsymbol{x}^0)}{\boldsymbol{x} - \boldsymbol{x}^0} \tag{12}$$

For the gradient, one would take the limit $\boldsymbol{x}^0 \to \boldsymbol{x}$. DeepLIFT uses a so-called reference point for $\boldsymbol{x}^0$ instead, such as zeros or for images a blurred version of $\boldsymbol{x}$. The finite differences are backpropagated, similar to infinitesimal differences. The final relevance is the difference in the k-th logit: $r_l^{DL}(\boldsymbol{x}) = f_k(x) - f_k(x^0)$.

Additionally to the vanilla gradient, DeepLIFT separates positive and negative contributions. For ReLU activations, DeepLIFT uses either the RevealCancel or the Rescale rule. Please refer to (Shrikumar et al., 2017) for a description. The rule for linear layers is most interesting because it is the reason why DeepLIFT does not converge:

$$\begin{aligned} r_l^{DL+}(\boldsymbol{x}, \boldsymbol{x}^0) = & M_{>0}^T \odot \Big({W_l^+}^T r_{l+1}^{DL+}(\boldsymbol{x}, \boldsymbol{x}^0) \\ & + {W_l^-}^T r_{l+1}^{DL-}(\boldsymbol{x}, \boldsymbol{x}^0) \Big) \end{aligned} \tag{13}$$

where the mask $M_{>0}$ selects the weight rows corresponding to positive deltas ($0 < \Delta \boldsymbol{h}_l = \boldsymbol{h}_l - \boldsymbol{h}_l^0$). For negative relevance r_l^{DL-}, the rule is defined analogously. An interesting property of the rule (13) is that negative and positive relevance can influence each other.

If the intermixing is removed by only considering W^+ for the positive rule and W^- for the negative rule, the two matrix chains become decoupled and converge. For the positive chain, this is clear. For the negative chain, observe that the multiplication of two non-positive matrices gives a non-negative matrix. Non-positive vectors $\boldsymbol{b}, \boldsymbol{c}$ have an angle $\leq 90°$ and $\boldsymbol{c}^T\boldsymbol{b} = \|\boldsymbol{c}\| \|\boldsymbol{b}\| \cos(\boldsymbol{c}, \boldsymbol{b}) \geq 0$. In the evaluation, we included this variant as *DeepLIFT Ablation*, and as predicted by the theory, it converges.

Guided BP & Deconv & RectGrad apply an additional ReLU to the gradient and it was shown to be invariant to the randomization of later layers previously in (Adebayo et al., 2018) and analyzed theoretically in (Nie et al., 2018):

$$r_l^{GBP}(\boldsymbol{x}) = W_l^T \left[M_l r_{l+1}^{GBP}(\boldsymbol{x}) \right]^+ . \tag{14}$$

$M_l = \operatorname{diag}(1_{h_1 > 0})$ denotes the gradient mask of the ReLU operation. For Deconv, the mask of the forward ReLU is omitted, and the gradients are rectified directly. RectGrad (Kim et al., 2019) is related to GuidedBP and set the lowest q percentile of the gradient to zero. As recommended in the paper, we used $q = 98$.

As a ReLU operation is applied to the gradient, the backpropagation is no longer a linear function. The ReLU also results in a different failure than before. (Nie et al., 2018) provides a theoretical analysis for GuidedBP. Our results align with them.

3. Evaluation

Setup We report results on a small network trained on CIFAR-10 (4x conv., 2x dense, see appendix D), a VGG-16 (Simonyan & Zisserman, 2014), and ResNet-50 (He et al., 2016). The last two are trained on the ImageNet dataset (Russakovsky et al., 2015), the standard dataset to evaluate attribution methods. The different networks cover different concepts: shallow vs. deep, forward vs. residual connections, multiple dense layers vs. a single one, using batch normalization. All results were computed on 200 images from the validation set. To justify the sample size, we show bootstrap confidence intervals in Figure 4b (Efron, 1979). We used the implementation from the *innvestigate* and *deeplift* package (Alber et al., 2019; Shrikumar et al., 2017) and added support for residual connections. The experiments were run on a single machine with two graphic cards and take about a day to complete.

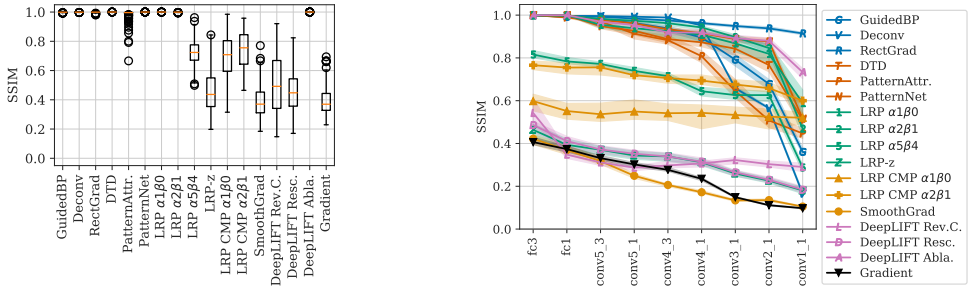
Random Logit We display the difference of saliency maps explaining the ground-truth and a random logit in Figure 4a. As the logit value is responsible for the predicted class, the saliency maps should change. We use the SSIM metric (Wang et al., 2004) as in (Adebayo et al., 2018).

Sanity Check We followed (Adebayo et al., 2018) and randomized the parameters starting from the last layer to the first layer. For DTD and $\text{LRP}_{\alpha 1 \beta 0}$, randomizing the last layer flips the sign of the saliency map sometimes. We, therefore, compute the SSIM also between the inverted saliency map and report the maximum. In Figure 4b, we report the SSIM between the saliency maps (see also Figure 1a and appendix G).[1]

Cosine Similarity Convergence Metric (CSC) Instead of randomizing the parameters, we randomize the backpropagated relevance vectors directly. We select layer k and set the corresponding relevance to $r_k(\boldsymbol{x}) := \boldsymbol{v}_1$ where $\boldsymbol{v}_1 \sim \mathcal{N}(0, I)$ and then backpropagate it as before. For example, for the gradient, we would do: $\frac{\partial h_k}{\partial h_1} \frac{\partial f(x)}{\partial h_k} := \frac{\partial h_k}{\partial h_1} \boldsymbol{v}_1$. We use the notation $r_l(\boldsymbol{x} | r_k := \boldsymbol{v}_1)$ to describe the relevance r_l at layer l when the relevance of layer k is set to $\boldsymbol{v}_1$.

Using two random relevance vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \sim \mathcal{N}(0, I)$, we measure the convergence using the cosine similarity. A rank-1 matrix $C = \boldsymbol{c}\boldsymbol{\gamma}^T$ always yields the same direc-

[1] For GuidedBP, we report different saliency maps than shown in Figure 2 of (Adebayo et al., 2018). We were able to confirm a bug in their implementation, resulting in saliency maps of GuidedBP and Guided-GradCAM to remain identical for early layers.



(a) Random Logits

(b) Parameter Randomization

Figure 4: **(a)** SSIM between saliency maps explaining the ground-truth or a random logit. **(b)** The parameters of the VGG-16 are randomized, starting from the last to the first layer. SSIM quantifies the difference to the saliency map from the original model. Intervals show 99% bootstrap confidences.

tion: $C\boldsymbol{v} = \boldsymbol{c}\boldsymbol{\gamma}^T = \lambda\boldsymbol{c}$. If the matrix chain converges, the backpropagated relevance vectors of $\boldsymbol{v}_1, \boldsymbol{v}_2$ will align more and more. We quantify their alignment using the cosine similarity $s_{\cos}(r_l(\boldsymbol{x}|r_k{:=}\boldsymbol{v}_1), r_l(\boldsymbol{x}|r_k{:=}\boldsymbol{v}_2)))$ where $s_{\cos}(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T\boldsymbol{b} \;/\; (\|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\|)$.

Suppose the relevance matrix chain would converge to a rank-1 matrix perfectly, than we have for both $\boldsymbol{v}_1, \boldsymbol{v}_2$: $r_l(\boldsymbol{x}|r_k{:=}\boldsymbol{v}_i) = C\boldsymbol{v}_i = \boldsymbol{c}\boldsymbol{\gamma}^T\boldsymbol{v}_i = \lambda_i\boldsymbol{c}$ where $\lambda_i = \boldsymbol{\gamma}^T\boldsymbol{v}_i$ and their cosine similarity will be one. The opposite direction is also true. If C has shape $n \times m$ with $n \leq m$ and if for n linearly independent vectors $\boldsymbol{v}_i$, the cosine similarity $s_{\cos}(C\boldsymbol{v}_i, C\boldsymbol{v}_j) = 1$, then C is a rank-1 matrix.

An alternative way to measure convergence would have been to construct the derivation matrix $C_k = \prod_{l=1}^{k} Z_l$ and measure the ratio $\sigma_1(C_k)/\sigma_2(C_k)$ of the first to the second-largest singular value of C_k. Although this approach is well motivated theoretically, it has some performance downsides. C_k would be large and computing the singular values costly.

We use five different random vectors per sample – in total 1000 convergence paths. As the vectors are sampled randomly, it is unlikely to miss a region of non-convergence (Bergstra & Bengio, 2012).

For convolution layers, we compute the cosine similarity per feature map location. For a shape of (h, w, c), we obtain $h \cdot w$ values. The jump in cosine similarity for the input is a result of the input's low dimension of 3 channels. In Figure 5, we plot the median cosine similarity for different networks and attribution methods (see appendix F for additional Figures). We also report the histogram of the CSC at the first convolutional layer in Figures 5e-5g.

4. Results

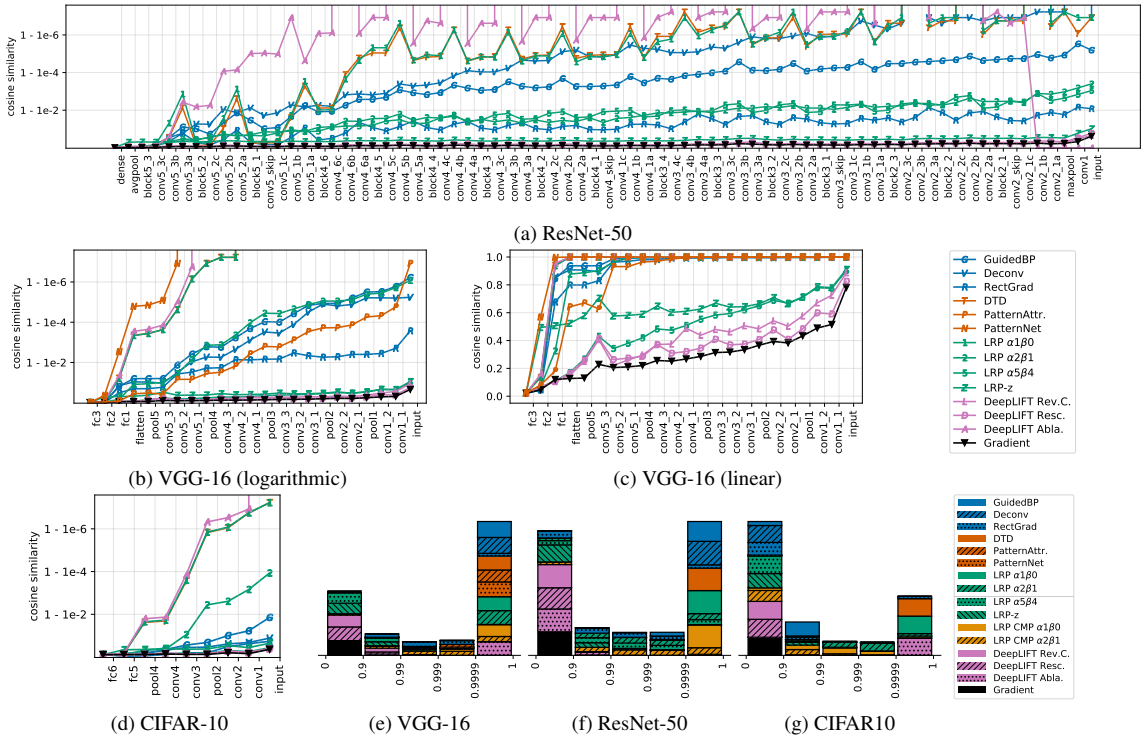
Our random logit analysis reveals that converging methods produce almost identical saliency maps, independently of the output logit (SSIM very close to 1). The rest of the field (SSIM between 0.4 and 0.8) produces saliency maps different from the ground-truth logit's map (see Figure 4a).

We observe the same distribution in the sanity check results (see Figure 4b). One group of methods produces similar saliency maps even when convolutional layers are randomized (SSIM close to 1). Again, the rest of the field is sensitive to parameter randomization. The same clustering can be observed for ResNet-50 (appendix E, Figure 8).

Our CSC analysis confirms that random relevance vectors align throughout the backpropagation steps (see Figure 5). Except for LRP$_z$ and DeepLIFT, all methods show convergence up to at least 0.99 cosine similarity. LRP$_{\alpha5\beta4}$ converges less strongly for VGG-16. Among the converging methods, the rate of convergence varies. LRP$_{\alpha1\beta0}$, PatternNet, the ablation of DeepLIFT converges fastest. PatternAttribution has a slower convergence rate – still exponential. For DeepLIFT Ablation, numerical instabilities result in a cosine similarity of 0 for the first layers of the ResNet-50. Even on the small 6-layer network, the median CSC is greater than 1-1e-6 for LRP$_{\alpha1\beta0}$ (see Figure 5d).

5. Discussion

When many modified BP methods do not explain the network faithfully, why was this not widely noticed before? First, it is easy to blame the network for unreasonable explanations – no ground truth exists. Second, MNIST, CIFAR, and ImageNet contain only a single object class per image – not revealing the class insensitivity. Finally, it might not be too problematic for some applications if the saliency maps are independent of the later network's layers. For example, to explain Alzheimer's disease (Böhle et al., 2019), local low-level features are sufficient as they are predictive for the disease and the data lacks conflicting evidences (i.e. the whole brain is affected).



(a) ResNet-50

(b) VGG-16 (logarithmic)

(c) VGG-16 (linear)

(d) CIFAR-10

(e) VGG-16

(f) ResNet-50

(g) CIFAR10

Figure 5: **(a)-(d)** Median of the cosine similarity convergence (CSC) per layer between relevance vectors obtained from randomizing the relevance vectors of the final layer. **(e)-(g)** histogram of the distribution of the CSC after the first layer.

When noticed, different ways to address the issue were proposed and an improved class sensitivity was reported (Kohlbrenner et al., 2019; Gu et al., 2018; Zhang et al., 2018). We find that the underlying convergence problem remains unchanged and discuss the methods below.

LRP$_{CMP}$ (Kohlbrenner et al., 2019; Lapuschkin et al., 2017) use LRP$_z$ for the final dense layers and LRP$_{\alpha\beta}$ for the convolutional layer. We report results for $\alpha = 1, 2$ as in (Kohlbrenner et al., 2019) in Figure 6a.

For VGG-16, the saliency maps change when the network parameters are randomized. However, structurally, the underlying image structure seems to be scaled only locally (see Figure 6a). Inspecting the CSC path of the two LRP$_{CMP}$ variants in Figure 6c, we can see why. For dense layers, both methods do not converge as LRP$_z$ is used, but the convergence start when LRP$_{\alpha\beta}$ is applied. The relevance vectors of the dense layer can change the coarse local scaling. However, they cannot alter the direction of the relevance vectors of earlier layers to highlight different details.

In the backward-pass of the ResNet-50, the global-averaging layer assigns the identical gradient vector to each location of the last convolutional layer. Furthermore, the later convolutional layers operate on (7x7), where even a few 3x3 convolutions have a dense field-of-view. LRP$_{CMP}$ does not resolve the global convergence for the ResNet-50.

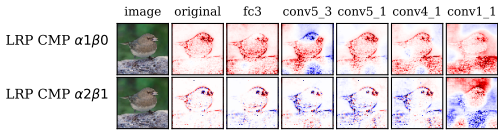
Contrastive LRP (Gu et al., 2018) noted the lack of class sensitivity and proposed to increase it by subtracting two saliency maps. The first saliency map explains only the logit $\boldsymbol{y}_k = \boldsymbol{y} \odot \boldsymbol{m}_k$, where $\boldsymbol{m}_k$ is a one-hot vector and the second explains the opposite $\boldsymbol{y}_{\neg k} = \boldsymbol{y} \odot (1 - \boldsymbol{m}_k)$:

$$\begin{aligned} \max(0, & n(r_{\boldsymbol{x}}^{z^+}(\boldsymbol{x}|r_{\text{logits}} := \boldsymbol{y}_k)) \\ & - n(r_{\boldsymbol{x}}^{z^+}(\boldsymbol{x}|r_{\text{logits}} := \boldsymbol{y}_{\neg k})) \end{aligned} \tag{15}$$

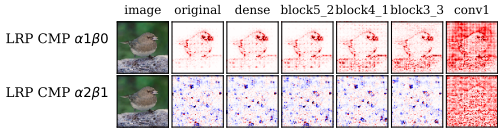
$n(.)$ normalizes each saliency map by its sum. The results of Contrastive LRP are similar to Figre 1e, no $\max$ is applied. The underlying convergence problem is not resolved.

Contrastive Excitation BP The lack of class sensitivity of the z^+-rule was noted in (Zhang et al., 2018) and to increase it, they proposed to change the backpropagation rule of the final fully-connected layer to:

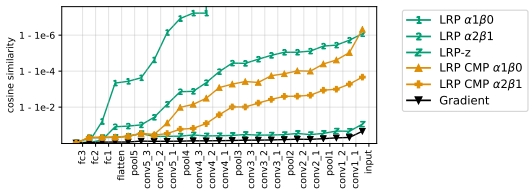
$$r_{\text{final fc}}^{\text{cEBP}}(x) = (Z_{\text{final fc}}^{+} - N_{\text{final fc}}^{+})\boldsymbol{m}_k, \tag{16}$$



(a) VGG-16



(b) ResNet-50



(c) VGG-16

Figure 6: **(a-b)** Sanity checks and **(c)** CSC for LRP$_{CMP}$.

where $\boldsymbol{m}_k$ is a one-hot vector selecting the explained class. The added $N^+_{\text{final fc}}$ is computed as the $Z^+_{\text{final fc}}$ but on the negative weights $-W_{\text{final fc}}$. Note that the combination of the two matrices introduces negative entries. Class sensitivity is increased. It does also not resolve the underlying convergence problem. If, for example, more fully-connected layers would be used, the saliency maps would become globally class insensitive again.

Texture vs. Contours (Geirhos et al., 2019) found that deep convolutional networks are more sensitive towards texture and not the shape of the object. For example, the shape of a cat filled with an elephant texture will be wrongly classified as an elephant. However, modified BP methods highlight the contours of objects rather.

Recurrent Neural Networks Modified BP methods are focused on convolutional neural networks and are mostly applied on vision tasks. The *innvestigate* package does not yet support recurrent models. To our knowledge, (Arras et al., 2017) is the only work that applied modified BP rules to RNNs (LRP$_z$ for LSTMs). Training, and applying modified backpropagation rules to RNNs, involves unrolling the network, essentially transforming it to a feed-forward architecture. Due to our theoretical results, modified BP rules that yield positive relevance matrices (e.g. z^+-rule) will converge. However, further work would be needed to measure how RNN architectures (LSTM, GRU) differ in their specific convergence behavior.

Not Converging Attribution Methods Besides modified BP attribution methods, there also exist gradient averaging and black-box methods. *SmoothGrad* (Smilkov et al., 2017) and *Integrated Gradients* (Sundararajan et al., 2017) average the gradient. *CAM* and *Grad-CAM* (Zhou et al., 2016; Selvaraju et al., 2017) determine important areas by the activation of the last convolutional layer. Black-box attribution methods only modify the model's input but do not rely on the gradient or other model internals. The most prominent black-box methods are *Occlusion*, *LIME*, *SHAP* (Zeiler & Fergus, 2014; Ribeiro et al., 2016; Lundberg & Lee, 2017). *IBA* (Schulz et al., 2020) applies an information bottleneck to remove unimportant information. *TCAV* (Kim et al., 2018) explains models using higher-level concepts.

All here mentioned attribution methods do *not* converge, as they either rely on the gradient or treat the model as black-box. Only when the BP algorithm is modified, the convergence problem can occur. The here mentioned algorithms might still suffer from other limitations.

Limitations Also, we tried to include most modified BP attribution methods, we left some out for our evaluation (Nam et al., 2019; Wang et al., 2019; Huber et al., 2019). In our theoretical analysis of PatternAttribution, we based our argument on why it converges on empirical observations performed on a single set of pattern matrices.

6. Related Work

Limitations of attribution The limitations of explanation methods were studied before. (Viering et al., 2019) alter the explanations of Grad-CAM arbitrarily by modifying the model architecture only slightly. Similarly, (Slack et al., 2020) construct a biased classifier that can hide its biases from LIME and SHAP. The theoretic analysis (Nie et al., 2018) indicates that GuidedBP tends to reconstruct the input instead of explaining the network's decision. (Adebayo et al., 2018) showed GuidedBP to be independent of later layers' parameters. (Atrey et al., 2020) tested saliency methods in a reinforcement learning setting.

(Kindermans et al., 2018) show that LRP, GuidedBP, and Deconv produce incorrect explanations for linear models if the input contains noise. (Rieger, 2017; Zhang et al., 2018; Gu et al., 2018; Kohlbrenner et al., 2019; Montavon et al., 2019; Tsunakawa et al., 2019) noted the class-insensitivity of different modified BP methods, but they rather proposed ways to improve the class sensitivity than to provide correct reasons why modified BP methods are class insensitive. Other than argued in (Gu et al., 2018), the class insensitivity is not caused by missing ReLU masks and Pooling switches. To the best of our knowledge, we are the first to identify the reason why many modified BP methods do not explain the decision of deep neural networks faithfully.

Evaluation metrics for attribution As no ground-truth data exists for feature importance, different proxy tasks were proposed to measure the performance of attribution algorithms. One approach is to test how much relevance falls into ground-truth bounding boxes (Schulz et al., 2020; Zhang et al., 2018).

The *MoRF* and *LeRF* evaluation removes the *mo*st and *le*ast relevant input features and measures the change in model performance (Samek et al., 2016). The relevant image parts are masked usually to zero. On these modified samples, the model might not be reliable. The *ROAR* score improves it by retraining the model from scratch (Hooker et al., 2018). While computationally expensive, it ensures the model performance does not drop due to out-of-distribution samples. The ROAR performance of Int.Grad. and GuidedBP is equally bad, worse than a random baseline (see Figre 4 in (Hooker et al., 2018)). Thus, ROAR does not separate converging from non-converging methods.

Our CSC measure has some similarities with the work (Balduzzi et al., 2017), which analyzes the effect of skip connections on the gradient. They measure the convergence between the gradient vector from different samples using the effective rank (Vershynin, 2012). The CSC metric applies to modified BP methods and is an efficient tool to trace the degree of convergence.

A different approach to verify attribution methods is to measure how helpful they are for humans (Alqaraawi et al., 2020; Doshi-Velez & Kim, 2017; Lage et al., 2018).

7. Conclusion

In our paper, we analyzed modified BP methods, which aim to explain the predictions of deep neural networks. Our analysis revealed that most of these attribution methods have theoretical properties contrary to their goal. PatternAttribution and LRP cite Deep Taylor Decomposition as the theoretical motivation. In the light of our results, revisiting the theoretical derivation of Deep Taylor Decomposition may prove insightful. Our theoretical analysis stresses the importance of negative relevance values. A possible way to increase class-sensitivity and resolve the convergence problem could be to backpropagate negative relevance similar to DeepLIFT, the only method passing our test.

8. Acknowledgements

We are grateful to the comments by our reviewers, which help to improve the manuscript further. We thank Benjamin Wild and David Dormagen for stimulating discussions. We also thank Avanti Shrikumar for answering our questions and helping us with the DeepLIFT implementation. The comments by Agathe Balayn, Karl Schulz, and Julian Stastny improved the manuscript. Furthermore, we thank Günter Rote for highlighting an inaccuracy in Theorem 1, leading to its current revision and update. A special thanks goes to the anonymous reviewer 1 of our paper (Schulz et al., 2020), who encouraged us to report results on the sanity checks — the starting point of this paper. The Elsa-Neumann-Scholarship by the state of Berlin supported LS. We are also grateful to Nvidia for a Titan Xp and to ZEDAT for access their HPC system.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pp. 9505–9515, 2018.

Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K. T., Montavon, G., Samek, W., Müller, K.-R., Dähne, S., and Kindermans, P.-J. innvestigate neural networks! *Journal of Machine Learning Research*, 20(93):1–8, 2019.

Alqaraawi, A., Schuessler, M., Weiß, P., Costanza, E., and Berthouze, N. Evaluating saliency map explanations for convolutional neural networks: A user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, pp. 263–274, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.1145/3377325.3377519.

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. A unified view of gradient-based attribution methods for deep neural networks. In *NIPS 2017-Workshop on Interpreting, Explaining and Visualizing Deep Learning*. ETH Zurich, 2017.

Ando, T., Horn, R. A., and Johnson, C. R. The singular values of a hadamard product: a basic inequality. *Linear and Multilinear Algebra*, 21(4):345–365, 1987. doi: 10.1080/03081088708817810.

Arras, L., Montavon, G., Müller, K.-R., and Samek, W. Explaining Recurrent Neural Network Predictions in Sentiment Analysis. In *Proceedings of the EMNLP 2017 Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 159–168. Association for Computational Linguistics, 2017.

Atrey, A., Clary, K., and Jensen, D. Exploratory Not Explanatory: Counterfactual Analysis of Saliency Maps for Deep Reinforcement Learning. In *International Conference on Learning Representations*, April 2020.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance

propagation. *PLoS ONE*, 10(7), 2015. doi: 10.1371/journal.pone.0130140.

Balduzzi, D., Frean, M., Leary, L., Lewis, J., Ma, K. W.-D., and McWilliams, B. The shattered gradients problem: If resnets are the answer, then what is the question? In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 342–350. JMLR.org, 2017.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Böhle, M., Eitel, F., Weygandt, M., and Ritter, K. Layer-wise relevance propagation for explaining deep neural network decisions in mri-based alzheimer's disease classification. *Frontiers in Aging Neuroscience*, 11:194, 2019. ISSN 1663-4365. doi: 10.3389/fnagi.2019.00194.

Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv: 1702.08608*, 2017.

Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. ISSN 00905364.

Eitel, F., Soehler, E., Bellmann-Strobl, J., Brandt, A. U., Ruprecht, K., Giess, R. M., Kuchling, J., Asseyer, S., Weygandt, M., Haynes, J.-D., Scheel, M., Paul, F., and Ritter, K. Uncovering convolutional neural network decisions for diagnosing multiple sclerosis on conventional mri using layer-wise relevance propagation. *NeuroImage: Clinical*, 24, 2019. doi: https://doi.org/10.1016/j.nicl.2019.102003.

Friedland, S. Convergence of products of matrices in projective spaces. *Linear Algebra and its Applications*, 413 (2):247 – 263, 2006. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2004.06.021. Special Issue on the 11th Conference of the International Linear Algebra Society, Coimbra, 2004.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

Gu, J., Yang, Y., and Tresp, V. Understanding individual decisions of cnns via contrastive backpropagation. In *Asian Conference on Computer Vision*, pp. 119–134. Springer, 2018.

Hajnal, J. On products of non-negative matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 79(3):521–530, May 1976. ISSN 0305-0041, 1469-8064.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hooker, S., Erhan, D., Kindermans, P.-J., and Kim, B. Evaluating Feature Importance Estimates. *arXiv: 1806.10758*, 2018.

Huber, T., Schiller, D., and André, E. Enhancing explainability of deep reinforcement learning through selective layer-wise relevance propagation. In *KI 2019: Advances in Artificial Intelligence*, pp. 188–202, Cham, 2019. Springer International Publishing. ISBN 978-3-030-30179-8.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pp. 2673–2682, 2018.

Kim, B., Seo, J., Jeon, S., Koo, J., Choe, J., and Jeon, T. Why are Saliency Maps Noisy? Cause of and Solution to Noisy Saliency Maps. *arXiv: 1902.04893*, 2019.

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

Kindermans, P.-J., Schütt, K. T., Alber, M., Müller, K.-R., Erhan, D., Kim, B., and Dähne, S. Learning how to explain neural networks: Patternnet and patternattribution. In *International Conference on Learning Representations*, 2018.

Kohlbrenner, M., Bauer, A., Nakajima, S., Binder, A., Samek, W., and Lapuschkin, S. Towards best practice in explaining neural network decisions with lrp, 2019.

Lage, I., Ross, A., Gershman, S. J., Kim, B., and Doshi-Velez, F. Human-in-the-loop interpretability prior. In *Advances in Neural Information Processing Systems*, pp. 10159–10168, 2018.

Lapuschkin, S., Binder, A., Muller, K.-R., and Samek, W. Understanding and comparing deep neural networks for age and gender classification. In *The IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct 2017.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017.

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

Montavon, G., Binder, A., Lapuschkin, S., Samek, W., and Müller, K.-R. Layer-wise relevance propagation: an overview. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 193–209. Springer, 2019.

Nam, W.-J., Gur, S., Choi, J., Wolf, L., and Lee, S.-W. Relative attributing propagation: Interpreting the comparative contributions of individual units in deep neural networks. *arXiv:1904.00605*, Nov 2019.

Nie, W., Zhang, Y., and Patel, A. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, pp. 3806–3815, 2018.

Ribeiro, M. T., Singh, S., and Guestrin, C. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939778.

Rieger, L. Separable explanations of neural network decisions. In *NIPS 2017 Workshop on Interpretable Machine Learning*, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

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

Schiller, D., Huber, T., Lingenfelser, F., Dietz, M., Seiderer, A., and André, E. Relevance-Based Feature Masking: Improving Neural Network Based Whale Classification Through Explainable Artificial Intelligence. In *Proc. Interspeech 2019*, pp. 2423–2427, 2019.

Schulz, K., Sixt, L., Tombari, F., and Landgraf, T. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

Shrikumar, A., Greenside, P., Shcherbina, A., and Kundaje, A. Not just a black box: Learning important features through propagating activation differences. *arXiv: 1605.01713*, 2016.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3145–3153. JMLR.org, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, pp. 180–186, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.1145/3375627.3375830.

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

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for Simplicity: The All Convolutional Net. *arXiv: 1412.6806*, 2014.

Sturm, I., Lapuschkin, S., Samek, W., and Müller, K.-R. Interpretable deep neural networks for single-trial eeg classification. *Journal of neuroscience methods*, 274:141–145, 2016.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328. JMLR. org, 2017.

Tsunakawa, H., Kameya, Y., Lee, H., Shinya, Y., and Mitsumoto, N. Contrastive relevance propagation for interpreting predictions by a single-shot object detector. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, July 2019. doi: 10.1109/IJCNN.2019.8851770.

Vershynin, R. *Introduction to the non-asymptotic analysis of random matrices*, pp. 210–268. Cambridge University Press, 2012. doi: 10.1017/CBO9780511794308.006.

Viering, T., Wang, Z., Loog, M., and Eisemann, E. How to manipulate cnns to make them lie: the gradcam case, 2019.

Wang, S., Zhou, T., and Bilmes, J. Bias also matters: Bias attribution for deep neural network explanation. In *International Conference on Machine Learning*, pp. 6659–6667, 2019.

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

Yang, Y., Tresp, V., Wunderle, M., and Fasching, P. A. Explaining therapy predictions with layer-wise relevance propagation in neural networks. In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 152–162, June 2018.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zhan, X. Inequalities for the Singular Values of Hadamard Products. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1093–1095, October 1997. ISSN 0895-4798, 1095-7162. doi: 10.1137/S0895479896309645.

Zhang, J., Bargal, S. A., Lin, Z., Brandt, J., Shen, X., and Sclaroff, S. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10): 1084–1102, 2018.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2921–2929, 2016.

A. Proof of Theorem 1

In (Friedland, 2006) the theorem is proven for square matrices. In fact Theorem 1 can be deduced from this case by the following argument:

For a sequence of non-square matrices $(A_k)_{k\in\mathbb{N}} \in \mathbb{R}^{m_k \times l_k}$ of finite size $m_k, l_k \leq L$ we can always find a finite set of subsequent matrices that when multiplied together are a square matrix.

$$\underbrace{A_1 \cdot \ldots \cdot A_{n_1}}_{=:\bar{A}_1 \in \mathbb{R}^{m\times m}} \cdot \underbrace{A_{n_1+1} \cdot \ldots \cdot A_{n_2}}_{=:\bar{A}_2 \in \mathbb{R}^{m\times m}} \cdot \underbrace{A_{n_2+1} \cdot \ldots \cdot A_{n_3}}_{=:\bar{A}_3 \in \mathbb{R}^{m\times m}} \cdot \ldots \tag{17}$$

The matrices $\bar{A}_1, \bar{A}_2, \bar{A}_3, \ldots$ define a sequence of non-negative square matrices that fulfill the conditions in (Friedland, 2006) and therefore converge to a rank-1 matrix.

We provide another proof using the cosine similarity to show convergence. First, we outline the conditions on the matrix sequence A_n. Then, we state the theorem again and sketch our proof to give the reader a better overview. Finally, we prove the theorem in 4 steps.

Conditions on A_n The first obvious condition is that the $(A_n)_{n\in\mathbb{N}}$ is a sequence of non-negative matrices such that A_i, A_{i+1} have the correct size to be multiplied together. Secondly, as we calculate angles between column vector in our proof, no column of A_n should be zero. The angle between a zero vector and any other vector is undefined. Furthermore, we assume that the entries of A_n cannot grow infinitely. They are bound such that $||\boldsymbol{a}||_1 \leq L \in \mathbb{R}^+$ for all column vectors $\boldsymbol{a}$ of any A_n. Finally, the size of A_n should not increase infinitely, i.e. an upper bound on the size of the A_i's exists such that $A_i \in \mathbb{R}^{m\times k}$ where $m, k \leq K$ for some $K \in \mathbb{N}$.

Theorem 1. Let $A_1, A_2, A_3 \ldots$ be a sequence of non-negative matrices as described above. We require that every column vector $\boldsymbol{a}$ of A_n has a norm $||\boldsymbol{a}|| \geq \epsilon_0$ and that infinite many matrices A_i with $i \in I$ and $|I| = |\mathbb{N}|$ exists for which two column vectors have a dot product of at least $\epsilon_{\langle\cdot,\cdot\rangle}$, i.e. $\langle \boldsymbol{a}, \boldsymbol{b}\rangle \geq \epsilon_{\langle\cdot,\cdot\rangle}$, where both $\epsilon_0, \epsilon_{\langle\cdot,\cdot\rangle} > 0$. Then the product of all terms of the sequence converges to a rank-1 matrix $\bar{C}$:

$$\bar{C} := \lim_{n\to\infty} \prod_{i=1}^{n} \frac{A_i}{||\prod_{i=1}^{n} A_i||} = \bar{\boldsymbol{c}}\boldsymbol{\gamma}^T \,. \tag{18}$$

Example Infinite many matrices in A_n must not be too orthogonal $\left(\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle < \epsilon_{\langle\cdot,\cdot\rangle}\right)$ or be too close to the zero vector $(|\boldsymbol{v}_i| < \epsilon_0)$. Matrices of the following form are *not in* the sequence A_n:

$$\Bigg(\underbrace{\boldsymbol{v}_1 \quad \ldots \quad \boldsymbol{v}_l}_{\text{arbitrary}} \quad \underbrace{\boldsymbol{v}_{l+1} \quad \ldots \quad \boldsymbol{v}_m}_{\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle < \epsilon_{\langle\cdot,\cdot\rangle}} \quad \underbrace{\boldsymbol{v}_{m+1} \quad \ldots \quad \boldsymbol{v}_n}_{|\boldsymbol{v}_i| < \epsilon_0}\Bigg)$$

up to ordering of the columns.

For example, this concrete example could only occur finite times:

$$\begin{pmatrix} 0 & 1 & 0.5 & \epsilon_{\langle\cdot,\cdot\rangle} & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & \underbrace{0}_{\langle \boldsymbol{v}_3, \boldsymbol{v}_4\rangle < \epsilon_{\langle\cdot,\cdot\rangle}} & \underbrace{1}_{} & \underbrace{0}_{||\boldsymbol{v}_i|| < \epsilon_0} & 0.5\epsilon_0 \end{pmatrix}.$$

Proof sketch To show that $\prod_i^\infty A_i$ converges to a rank-1 matrix, we do the following steps:

(1) We define a sequence s_n as the cosine of the maximum angle between the column vectors of $M_n := \prod_{i=1}^{n} A_i$.

(2) We show that the sequence s_n is monotonic and bounded and therefore converging.

(3) We introduce a complementary sequence $\varepsilon_n = 1 - s_n$ and show $s_{n+1} \geq s_n + \lambda\varepsilon_n$ with $\lambda > 0$

(4) We assume that $\lim_{n\to\infty} s_n = 1 - \varepsilon^* < 1$ and show that than $\lim_{n\to\infty} s_n \to \infty$ is diverging which is a contradiction and therefore $s_n \to 1$.

Proof To simplify the proof, we assume that all matrices in the sequence A_i are in I. We will show that any finite number of $(A_i)_{i\notin I}$ can be added without changing the result.

(1) Let $M_n := \prod_{i=1}^{n} A_i$ be the product of the matrices $A_1 \cdot \ldots \cdot A_n$. We define a sequence on the angles of column vectors of M_n using the cosine similarity. Let $\boldsymbol{v}_1(n), ..., \boldsymbol{v}_{k(n)}(n)$ be the column vectors of M_n. Note, the angles are well defined between the columns of M_n. The columns of M_n cannot be a zero vector as we required A_n to have no zero columns. Let s_n be the cosine of the maximal angle between the columns of M_n:

$$s_n = \min_{i\neq j} \mathrm{s}_{\cos}(\boldsymbol{v}_i(n), \boldsymbol{v}_j(n)) := \min_{i,j} \frac{\langle \boldsymbol{v}_i(n), \boldsymbol{v}_j(n)\rangle}{\|\boldsymbol{v}_i(n)\|\|\boldsymbol{v}_j(n)\|}, \tag{19}$$

where $\langle\cdot,\cdot\rangle$ denotes the dot product. We show that the maximal angle converges to 0 as $\lim_{n\to\infty} s_n = 1$, which is equivalent to M_n converging to a rank-1 matrix. In the following, we take a look at two consecutive elements of the sequence s_n and check by how much the sequence increases.

(2) We show that the sequence s_n is monotonic and bounded and therefore converging. Assume $\boldsymbol{a}_{n+1}$ and $\boldsymbol{b}_{n+1}$ are the two columns of A_{n+1} which produce the columns $\boldsymbol{v}_m(n+1)$ and $\boldsymbol{v}_{m'}(n+1)$ of M_{n+1} with the maximum angle:

$$s_{n+1} = \mathrm{s}_{\cos}(\boldsymbol{v}_m(n+1), \boldsymbol{v}_{m'}(n+1)) = \mathrm{s}_{\cos}(M_n\boldsymbol{a}_{n+1}, M_n\boldsymbol{b}_{n+1}). \tag{20}$$

We also assume that $\|\boldsymbol{v}_i(n)\| = 1$ for all i, since the angle is independent of length. To declutter notation, we write $\boldsymbol{v}_i(n) =: \boldsymbol{v}_i$, $\boldsymbol{a}_n = \boldsymbol{a} = (a_1, ..., a_k)^T$, $\boldsymbol{b}_n = \boldsymbol{b} = (b_1, ..., b_k)^T$

Substituting $M_n\boldsymbol{a} = \sum_i a_i\boldsymbol{v}_i$ into the the definition of the cosine similarity, we show that s_n is monotonic:

$$s_{n+1} = \frac{\sum_{ij} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle}{\|\sum_i a_i\boldsymbol{v}_i\|\|\sum_i b_i\boldsymbol{v}_i\|} \tag{21}$$

Using the triangle inequality $\|\sum_i a_i\boldsymbol{v}_i\| \leq \sum_i a_i\|\boldsymbol{v}_i\|$ we get:

$$s_{n+1} \geq \frac{\sum_{ij} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle}{(\sum_i a_i\|\boldsymbol{v}_i\|)(\sum_i b_i\|\boldsymbol{v}_i\|)} \tag{22}$$

As we assumed that the $\|\boldsymbol{v}_i\| = 1$, we know that $\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle = s_{\cos}(\boldsymbol{v}_i, \boldsymbol{v}_j)$ which must be greater than the smallest cosine similarity s_n:

$$s_{n+1} \geq \frac{\sum_{ij} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle}{(\sum_i a_i)(\sum_i b_i)} \geq \frac{\sum_{ij} a_i b_j}{(\sum_i a_i)(\sum_i b_i)} s_n = s_n \tag{23}$$

Therefore s_n is monotonically increasing and upper-bounded by 1 as the cosine of the maximal angle. Due to the monotone convergence theorem, it will converge. The rest of the proof investigates if the sequence s_n converges to 1 and if so, under which conditions. Equation 23 makes it also clear that we can ignore any $(A_i)_{i\notin I}$, as the factor before s_n can never be lower than 1. All values are non-negative, $\sum_i a_i > 0$, and $\sum_i b_i > 0$.

(3) We now introduce a complementary sequence $\varepsilon_n = 1 - s_n$.

Using the result from equation 21, we write s_{n+1} as:

$$s_{n+1} = \frac{\sum_{ij} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle}{\|\sum_i a_i\boldsymbol{v}_i\|\|\sum_i b_i\boldsymbol{v}_i\|} = \frac{\sum_{i\neq j} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle + \sum_i a_i b_i \langle \boldsymbol{v}_i, \boldsymbol{v}_i\rangle}{\|\sum_i a_i\boldsymbol{v}_i\|\|\sum_i b_i\boldsymbol{v}_i\|} = \frac{\sum_{i\neq j} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle + \langle \boldsymbol{a}, \boldsymbol{b}\rangle}{\|\sum_i a_i\boldsymbol{v}_i\|\|\sum_i b_i\boldsymbol{v}_i\|}, \tag{24}$$

since we assumed that $\|\boldsymbol{v}_i\| = 1$. Now, we apply $1 = s_n + \varepsilon_n$ and $\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle \geq s_n$

$$s_{n+1} \geq \frac{(\sum_{i\neq j} a_i b_j)s_n + \langle \boldsymbol{a}, \boldsymbol{b}\rangle(s_n + \varepsilon_n)}{\|\sum_i a_i\boldsymbol{v}_i\|\|\sum_i b_i\boldsymbol{v}_i\|} \geq \frac{(\sum_{i\neq j} a_i b_j)s_n + \langle \boldsymbol{a}, \boldsymbol{b}\rangle(s_n + \varepsilon_n)}{(\sum_i |a_i|\|\boldsymbol{v}_i\|)(\sum_i |b_i|\|\boldsymbol{v}_i\|)} = \frac{(\sum_{i\neq j} a_i b_j)s_n + \langle \boldsymbol{a}, \boldsymbol{b}\rangle(s_n + \varepsilon_n)}{(\sum_i a_i)(\sum_i b_i)} \tag{25}$$

Here, we applied the triangle inequality and the fact that the matrix entries a_i and b_i are positive.

$$s_{n+1} \geq \frac{(\sum_{i \neq j} a_i b_j)}{(\sum_i a_i)(\sum_i b_i)} s_n + \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{(\sum_i a_i)(\sum_i b_i)}(s_n + \varepsilon_n) = \frac{(\sum_{ij} a_i b_j)}{(\sum_i a_i)(\sum_i b_i)} s_n + \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{(\sum_i a_i)(\sum_i b_i)} \varepsilon_n \tag{26}$$

As the factor before s_n is equals one, we have:

$$s_{n+1} \geq s_n + \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{(\sum_i a_i)(\sum_i b_i)} \varepsilon_n \geq s_n + \lambda \epsilon_n, \tag{27}$$

where $\lambda = \frac{\epsilon_{\langle \cdot, \cdot \rangle}}{L^2} < \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{(\sum_i a_i)(\sum_i b_i)}$, as $\epsilon_{\langle \cdot, \cdot \rangle}$ is the minimal dot product of any two column vectors and $\sum_i a_i = ||\boldsymbol{a}||_1 < L$, as L is the maximum matrix L1-norm of any column vector of A_n.

(4) Suppose, $\lim_{n \to \infty} s_n = 1 - \varepsilon^*$, where $\varepsilon^* > 0$. Then, for all ε_n, it must be $\varepsilon_n > \varepsilon^*$. Using the result from equation 27, we get:

$$s_{n+1} \geq \varepsilon_n \geq s_n + \lambda \varepsilon^* \tag{28}$$

We would therefore have:

$$\lim_{n \to \infty} s_n \geq \lim_{n \to \infty} n \lambda \varepsilon^* \to \infty. \tag{29}$$

This is a contradiction. Therefore, $\varepsilon^* = 0$ and $\lim_{n \to \infty} s_n = 1$. The matrix entries of M_n could grow to infinity, therefore M_∞ may not be defined. However, we normalize the product $\bar{M}_n = M_n / \|M_n\|$, then $\|\bar{M}_\infty\|_1 = 1$ and all the columns of $\bar{M}_\infty$ are the same up to a scalar multiple. □

Update to the Theorem 1 and the Proof: In a previous version of this manuscript, we did not explicitly require the dot-product $\langle a, b \rangle \geq \epsilon_{\langle \cdot, \cdot \rangle}$ to be greater than a constant, and also missed a similar constraint for the zero vector. As a result, we did not exclude cases where the matrix sequence A_n converged to such edge-cases. For example, the sequence $A_n = \begin{pmatrix} 1 & 0 \\ \frac{1}{n} & 1 \end{pmatrix}$ fulfilled the criteria of the previous version but is now excluded. We also made the normalization more explicit. An inaccurate statement about the "convergence of the matrix chain A_n", which is not actually required, is now removed. Furthermore, the updated version now contains a new derivation of the convergence speed. We thank Günter Rote for bringing these issues to our attention.

B. Convergence Speed

For the derivation of the convergence speed, we assume that all matrices A_n are in the set I. We start with the intermediate result from equation 27:

$$s_{n+1} \geq s_n + \lambda \varepsilon_n = s_n + \lambda(1 - s_n) = (1 - \lambda) s_n + \lambda. \tag{30}$$

We now define a new sequence $\xi_{n+1} = (1 - \lambda)\xi_n + \lambda$ with $\xi_0 = s_0$ that is a lower bound of s_{n+1}, i.e. for all $s_n \geq \xi_n$

Analyzing the first steps of ξ_n, we have:

$$\xi_1 = (1 - \lambda) s_0 + \lambda \tag{31}$$

$$\xi_2 = (1 - \lambda)\left((1 - \lambda) s_0 + \lambda\right) = (1 - \lambda)^2 s_0 + (1 - \lambda)\lambda \tag{32}$$

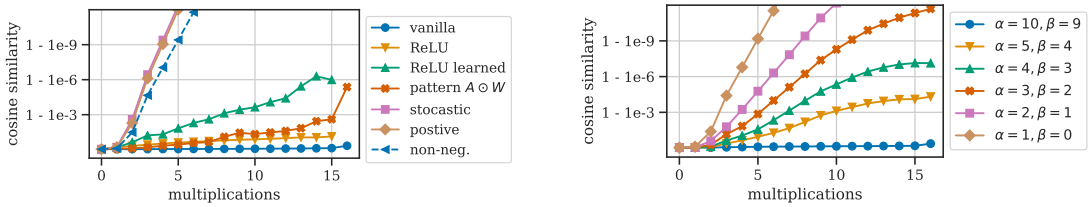
$$\xi_3 = (1 - \lambda)\left((1 - \lambda)^2 s_0 + (1 - \lambda)\lambda\right) = (1 - \lambda)^3 s_0 + (1 - \lambda)^2 \lambda \tag{33}$$

The general form of ξ_n is therefore:

$$\xi_n = (1 - \lambda)^n s_0 + \lambda \sum_{k=0}^{n-1} (1 - \lambda)^k. \tag{34}$$

The sum of a geometric series $\sum_{k=0}^{n-1} r^k$ is given by $\frac{1 - r^n}{1 - r}$. Applying this to the term of our series ξ_n:

$$\lambda \sum_{k=0}^{n-1} (1 - \lambda)^k = \lambda \left(\frac{1 - (1 - \lambda)^n}{1 - (1 - \lambda)} \right) = 1 - (1 - \lambda)^n. \tag{35}$$



(a) different matrix properties (see text)

(b) $\alpha W^+ + \beta W^-$ where $W_{[ij]} \sim \mathcal{N}(0, 1)$

Figure 7: Simulated convergence for a matrix chain.

Therefore, the closed-form solution for ξ_n is:

$$\xi_n = 1 - (1 - \lambda)^n (1 - s_0). \tag{36}$$

Therefore, s_n must also convergence exponentially fast in λ. We bounded $\lambda = \frac{\epsilon_{\langle \cdot, \cdot \rangle}}{L^2}$ by the minimum dot-product $\epsilon_{\langle \cdot, \cdot \rangle}$ and the maximum L1-norm of the columns of A_n. In the next section, we conduct an empirical analysis and find that the value of λ is large enough that typical matrices converge towards a rank-1 matrix up to floating-point impression within a few steps.

Empirical evaluation We further investigate the convergence speed using a simulation of random matrices and find that non-negative matrices decay exponentially fast towards 1.

We report the converging behavior for matrix chains which resembles a VGG-16. As in the backward pass, we start from the last layer. The convolutional kernels are considered to be 1x1, e.g. for a kernel of size (3, 3, 256, 128), we use a matrix of size (256, 128).

We test out the effect of different matrix properties. For *vanilla*, we sample the matrix entries from a normal distribution. Next, we apply a *ReLU* operation after each multiplication. For *ReLU learned*, we used the corresponding learned VGG parameters. We generate *non-negative* matrices containing 50% zeros by clipping random matrices to $[0, \infty]$. And *positive* matrices by taking the absolute value. We report the median cosine similarity between the column vectors of the matrix.

The y-axis of Figure 7a has a logarithmic scale. We observe that the positive, stochastic, and non-negative matrices yield a linear path, indicating an exponential decay of the form: $1 - \exp(-\lambda n)$. The 50% zeros in the non-negative matrices only result in a bit lower convergence slope. After 7 iterations, they converged to a single vector up to floating point imprecision.

We also investigated how a slightly negative matrix influences the convergence. In Figure 7b, we show the converges of matrices: $\alpha W^+ + \beta W^-$ where $W^+ = \max(0, W), W^- = \min(0, W)$ and $W \sim \mathcal{N}(0, I)$. We find that for small enough $\beta < 4$ values the matrix chains still converge. This simulation motivated us to include $\text{LRP}_{\alpha 5 \beta 4}$ in our evaluation which show less convergence on VGG-16, but its saliency maps also contain more noise.

C. Pattern Attribution

We derive equation 9 from the original equation given in (Kindermans et al., 2018). We will use the notation from the original paper and denote a weight vector with $\boldsymbol{w} = W_{l_{[i,:]}}$ and the corresponding pattern with $\boldsymbol{a} = A_{l_{[i,:]}}$. The output is $y = \boldsymbol{w}^T \boldsymbol{x}$.

Derivation of Pattern Computation For the positive patterns of the two-component estimator $S_{\boldsymbol{a}+-}$, the expectation is taken only over $\{\boldsymbol{x} | \boldsymbol{w}^T \boldsymbol{x} > 0\}$. We only show it for the positive patterns $\boldsymbol{a}_+$. As our derivation is independent of the subset of $\boldsymbol{x}$ considered, it would work analogously for negative patterns or the linear estimator $S_{\boldsymbol{a}}$.

The formula to compute the pattern $\boldsymbol{a}_+$ is given by:

$$\begin{aligned}\boldsymbol{a}_+ &= \frac{\mathbb{E}_+[\boldsymbol{x}y] - \mathbb{E}_+[\boldsymbol{x}]\,\mathbb{E}_+[y]}{\boldsymbol{w}^T\mathbb{E}_+[\boldsymbol{x}y] - \boldsymbol{w}^T\mathbb{E}_+[\boldsymbol{x}]\,\mathbb{E}_+[y]} \\ &= \frac{\operatorname{cov}[\boldsymbol{x}, \boldsymbol{w}^T\boldsymbol{x}]}{\boldsymbol{w}^T\operatorname{cov}[\boldsymbol{x}, \boldsymbol{w}^T\boldsymbol{x}]},\end{aligned} \tag{37}$$

where $\operatorname{cov}[\boldsymbol{x}, \boldsymbol{w}^T\boldsymbol{x}] = \mathbb{E}_+[\boldsymbol{x}y] - \mathbb{E}_+[\boldsymbol{x}]\mathbb{E}_+[y]$. Using the bilinearity of the covariance matrix ($\operatorname{cov}[\boldsymbol{b}, \boldsymbol{c}^T\boldsymbol{d}] = \operatorname{cov}[\boldsymbol{b}, \boldsymbol{d}]\boldsymbol{c}$), gives:

$$\boldsymbol{a}_+ = \frac{\operatorname{cov}[\boldsymbol{x}, \boldsymbol{x}]\boldsymbol{w}}{\boldsymbol{w}^T\operatorname{cov}[\boldsymbol{x}, \boldsymbol{x}]\boldsymbol{w}}. \tag{38}$$

Using the notation $\operatorname{cov}[\boldsymbol{h}] = \operatorname{cov}[\boldsymbol{x}, \boldsymbol{x}]$ gives equation 9.

Connection to power iteration A step of the power iteration is given by:

$$\boldsymbol{v}_{k+1} = \frac{M\boldsymbol{v}_k}{\|M\boldsymbol{v}_k\|} \tag{39}$$

The denominator in equation 9 is $\boldsymbol{w}^T\operatorname{cov}[\boldsymbol{h}]\boldsymbol{w}$. Using the symmetry of $\operatorname{cov}[\boldsymbol{h}]$, we have:

$$\left\|\operatorname{cov}[\boldsymbol{h}]^{1/2}\boldsymbol{w}\right\| = (\boldsymbol{w}^T\operatorname{cov}[\boldsymbol{h}]^{1/2}\operatorname{cov}[\boldsymbol{h}]^{1/2}\boldsymbol{w})^{1/2} = (\boldsymbol{w}^T cov[\boldsymbol{h}]\boldsymbol{w})^{1/2} \tag{40}$$

This should be similar to the norm $\|\operatorname{cov}[h]\boldsymbol{w}\|$. As only a single step of the power iteration is performed, the scaling should not matter that much. The purpose of the scaling in the power-iteration algorithm is to keep the vector v_k from exploding or converging to zero.

D. CIFAR-10 Network Architecture

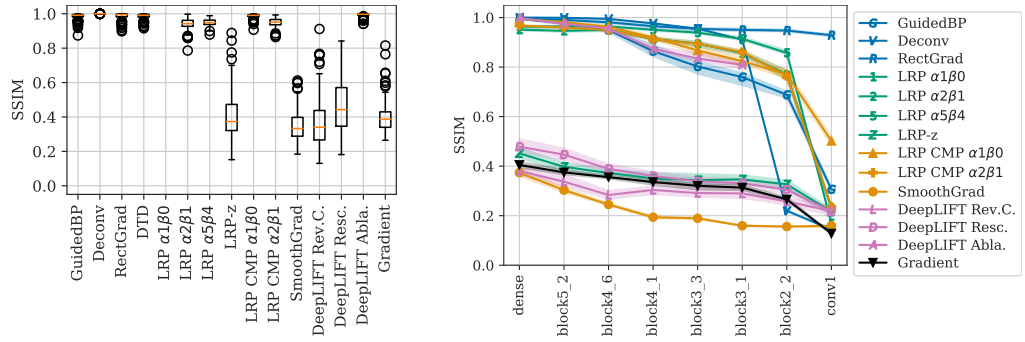
```
# network architecture as a keras model
model = Sequential()

model.add(InputLayer(input_shape=(32, 32, 3), name='input'))
model.add(Conv2D(32, (3, 3), padding='same', name='conv1'))
model.add(Activation('relu', name='relu1'))
model.add(Conv2D(64, (3, 3), padding='same', name='conv2'))
model.add(Activation('relu', name='relu2'))
model.add(MaxPooling2D(pool_size=(2, 2), name='pool2'))

model.add(Conv2D(128, (3, 3), padding='same', name='conv3'))
model.add(Activation('relu', name='relu3'))
model.add(Conv2D(128, (3, 3), padding='same', name='conv4'))
model.add(Activation('relu', name='relu4'))
model.add(MaxPooling2D(pool_size=(2, 2), name='pool4'))

model.add(Flatten(name='flatten'))
model.add(Dropout(0.5, name='dropout5'))
model.add(Dense(1024, name='fc5'))
model.add(Activation('relu', name='relu5'))
model.add(Dropout(0.5, name='dropout6'))
model.add(Dense(10, name='fc6'))
model.add(Activation('softmax', name='softmax'))
```

E. Results on ResNet-50

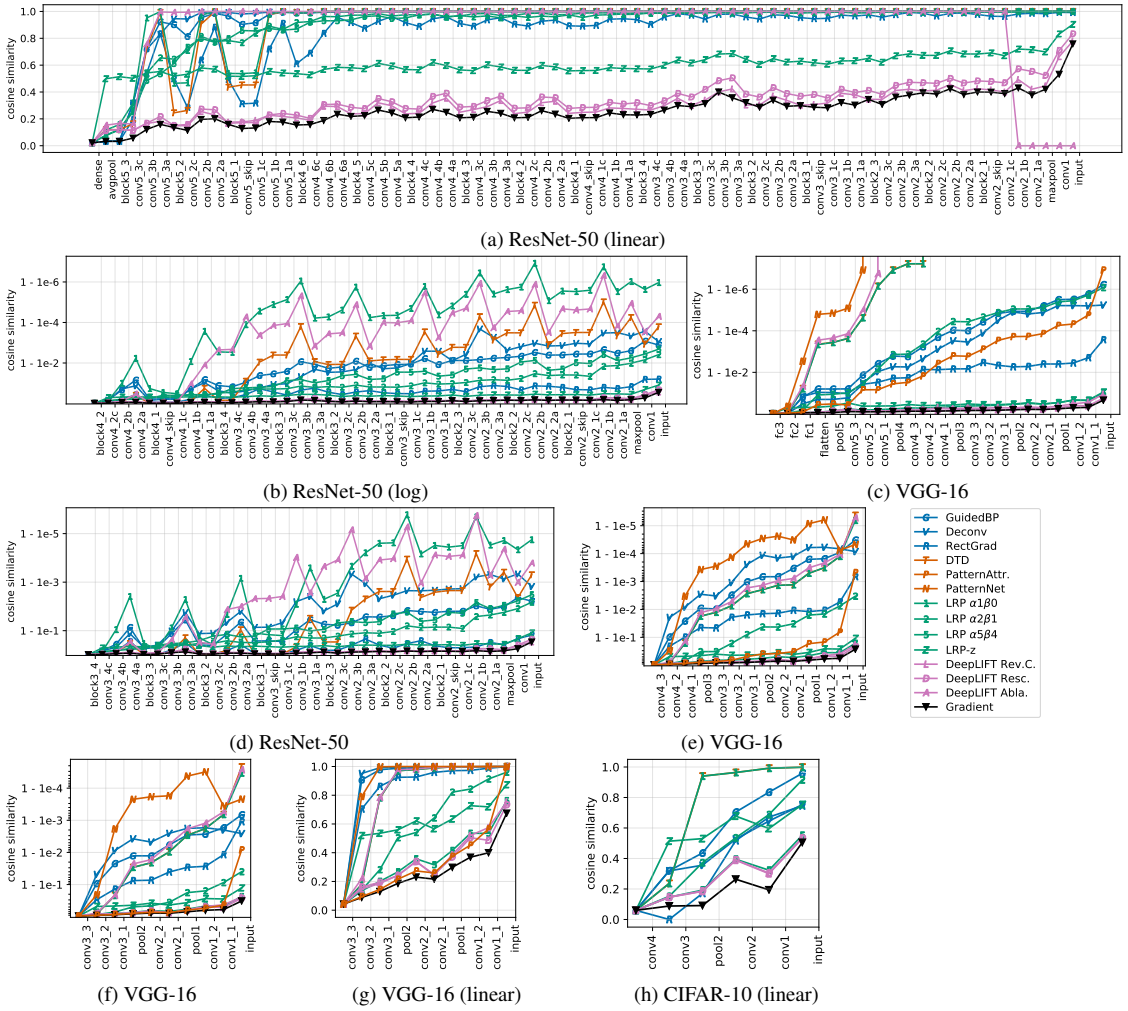


(a) Random Logits

(b) Cascading Parameter Randomization

Figure 8: Effect of (a) randomizing the logits or (b) the parameters on a ResNet-50.

F. Additional Cosine Similarity Figures



(a) ResNet-50 (linear)

(b) ResNet-50 (log)

(c) VGG-16

(d) ResNet-50

(e) VGG-16

(f) VGG-16

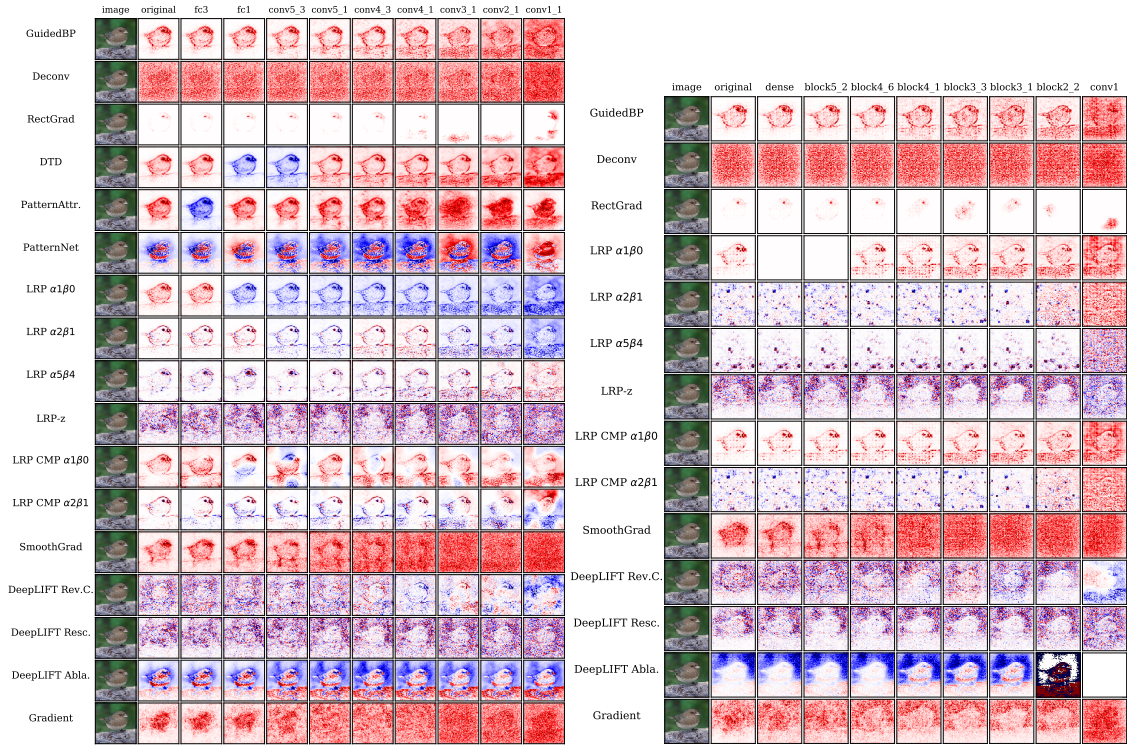
(g) VGG-16 (linear)

(h) CIFAR-10 (linear)

Figure 9: Convergence measured using the CSC for different starting layers.

G. Saliency maps for Sanity Checks

For visualization, we normalized the saliency maps to be in $[0, 1]$ if the method produce only positive relevance. If the method also estimates negative relevance, than it is normalized to $[-1, 1]$. The negative and positive values are scaled equally by the absolute maximum. For the sanity checks, we scale all saliency maps to be in $[0, 1]$.



(a) VGG-16. (b) ResNet-50.

Figure 10: Saliency maps for sanity checks. Parameters are randomized starting from last to first layer.

5 A Rigorous Study Of The Deep Taylor Decomposition

The previous chapter discussed that various modified backpropagation attribution methods failed to adequately explain the network's decision. For $\text{LRP}_{\alpha 1 \beta 0}$, it was shown that the problem is caused by a chain of positive matrices.

The authors of LRP have responded to our findings with their manuscript *CLEVR-XAI: A benchmark dataset for the ground truth evaluation of neural network explanations* (Arras et al. 2022). In this manuscript, they designed a synthetic benchmark for explainable AI: different geometric objects are arranged in a scene, and the task is to answer a question about the scene. A typical question might be "What color is the spherical object?". The saliency map is then expected to highlight the spherical object. In (Arras et al. 2022), the authors report that $\text{LRP}_{\alpha 1 \beta 0}$ performs better than any other method tested (Excitation Backprop, Integrated Gradients, Guided Backprop, Guided Grad-CAM, SmoothGrad, VarGrad, Gradient, Gradient$\times$Input, Deconvnet, Grad-CAM). Furthermore, based on their good empirical performance, they concluded that no convergence problem, as described in (Sixt et al. 2020) would exist:

> "In our VQA setup we instead find empirically a high connection between the relevance heatmaps and the target objects of each question, which even increases with the model's confidence (i.e., the output of the last layer in the network). Maybe the phenomenon described in (Sixt et al. 2020) becomes predominant in the asymptotic case of a neural network with a very high number of layers, and might be relevant for further improving XAI methods. However, in our practical use-case of a neural network with 12 layers, we could not confirm that this behavior alters the quality of modified backpropagation based explanations in comparison to gradient-based ones." — (Arras et al. 2022)

In (Arras et al. 2022), it was not discussed how this was possible, although we provided proof of this failure and showed that the matrix chain converges to a rank-1 matrix exponentially fast for their case of fully connected layers.

In another publication (Holzinger et al. 2022), LRP was advertised as follows:

> "The main advantages of LRP are its high computational efficiency [...], its theoretical underpinning making it a trustworthy and robust explanation method [...], and its long tradition and high popularity [...]." — (Holzinger et al. 2022)

I am convinced that this statement is biased since the authors of LRP did not refute our results from (Sixt et al. 2020); furthermore, they even refute discussing the limitations of the LRP method entirely.

The situation is further complicated by a claimed theoretical underpinning and justification of the LRP method, namely the Deep Taylor Decomposition (DTD), (Montavon et al. 2017). The main idea of the DTD is to recursively apply the Taylor Theorem to a relevance function. As a start the explained logit is used relevance and then the relevance is redistributed according to an application of the Taylor Theorem. The choice of root point leads to different derivation rules such as the $\text{LRP}_{\alpha 1 \beta 0}$ or the γ-rule.

The limitations uncovered in (Sixt et al. 2020) apply only to propagation rules with positive matrices and do not necessarily invalidate the entire DTD theory. This raised the question of whether the limitations are inherent to the DTD or whether they only apply to $\text{LRP}_{\alpha 1 \beta 0}$.

Our work (Sixt et al. 2022a) provides the previously missing theoretical analysis of the DTD. The result of this analysis yields an interesting duality for ReLU networks: (1) if the root points (a critical parameter of the DTD) are independent of the network's activations, then the DTD is equivalent to gradient$\times$input. (2) if the root points are independent of the network's activations, then the attribution values are determined by the Jacobian of the root points w.r.t. the network's activations. Those results are concerning: in case (1), we could have omitted the whole theory of DTD, and used gradient$\times$input right away; for case (2), we could justify any saliency value as the selection of the root point is left to the user.

(Sixt et al. 2022a) demonstrates the power of rigorous mathematical analysis in debunking the claims of a method and serves as a crucial component in my overall dissertation by reinforcing my criticism towards LRP and DTD methods. This work is an essential contribution to the scientific dispute about the DTD, highlighting the potential pitfalls and limitations of these methods

and emphasizing the need for caution when using saliency maps based on LRP and DTD for explainability purposes.

A Rigorous Study Of The Deep Taylor Decomposition

Leon Sixt *leon.sixt@fu-berlin.de*
Department of Computer Science
Freie Universität Berlin

Tim Landgraf *tim.landgraf@fu-berlin.de*
Department of Computer Science
Freie Universität Berlin

Reviewed on OpenReview: *https://openreview.net/forum?id=Y4mgmw9OgV*

Code Repository: *https://github.com/berleon/A-Rigorous-Study-Of-The-Deep-Taylor-Decomposition*

Abstract

Saliency methods attempt to explain deep neural networks by highlighting the most salient features of a sample. Some widely used methods are based on a theoretical framework called Deep Taylor Decomposition (DTD), which formalizes the recursive application of the Taylor Theorem to the network's layers. However, recent work has found these methods to be independent of the network's deeper layers and appear to respond only to lower-level image structure. Here, we investigate the DTD theory to better understand this perplexing behavior and found that the Deep Taylor Decomposition is equivalent to the basic gradient×input method when the Taylor root points (an important parameter of the algorithm chosen by the user) are locally constant. If the root points are locally input-dependent, then one can justify any explanation. In this case, the theory is under-constrained. In an empirical evaluation, we find that DTD roots do not lie in the same linear regions as the input – contrary to a fundamental assumption of the Taylor theorem. The theoretical foundations of DTD were cited as a source of reliability for the explanations. However, our findings urge caution in making such claims.

1 Introduction

Post-hoc explanations are popular for explaining Machine Learning models as they do not require changing the model's architecture or training procedure. In particular, feature attribution methods are widely used. They assign a saliency score to each input dimension, reflecting their relevance for the model's output. For images, the saliency scores can be visualized as heatmaps (see Figure 2).

Evaluating post-hoc explanations is challenging because it is inherently circular: As we do not understand the internal workings of the model, which we are trying to explain, we cannot judge the quality of the explanation. The situation is further complicated as many methods simplify the model's complexity to render explanations accessible to the human eye. For example, most methods focus on the local neighborhood of an input sample, and rely on assumptions such as linearity (e.g. gradient-based methods) or independence of the input features (e.g. approximation of Shapley values (Štrumbelj & Kononenko, 2011; 2014; Lundberg & Lee, 2017; Kumar et al., 2020)).

These factors and the complexity of deep neural networks make it difficult to assess whether an explanation is correct or not. We can not disentangle failures of the explanation method and unexpected behavior of the model. While it is acceptable for methods to introduce simplifications or rely on assumptions, their existence, purpose, and violation should be made transparent. In the best case, a method would be based on a solid theoretical foundation providing guarantees regarding an explanation's correctness.

Such a theoretical foundation is the *Deep Taylor Decomposition* (DTD, Montavon et al. (2017)). DTD recursively applies the Taylor Theorem to the network's layers, and backpropagates modified gradients to the input, thereby computing the input's relevance. It was used as theoretical foundation of LRP (Bach et al., 2015), a poplar method to explain image models. LRP was repeatedly advertised as a sound and reliable explanation technique (Montavon et al., 2019; Samek et al., 2021b; Holzinger et al., 2022). For example, Holzinger et al. (2022) stated: *"The main advantages of LRP are its high computational efficiency [...], its theoretical underpinning making it a trustworthy and robust explanation method [...], and its long tradition and high popularity [...]."*

However, Sixt et al. (2020) has shown that certain LRP and DTD backpropagation rules create explanations partially independent of the model's parameters: the explanation will remain the same even if the last layer's parameters are randomized. The theoretical analysis in (Sixt et al., 2020) revealed that the propagation matrices, which correspond to the layers' Jacobian matrices, are all positive and their product converges to a rank-1 matrix quickly. To obtain the saliency map, the result is usually normalized, and thereby even the last single degree of freedom is lost. Thus, the explanation does not change when explaining a different class or the parameters of the deeper layers is changed.

This perplexing behavior questions the consistency of DTD directly. While Sixt et al. (2020) described the convergence to the rank-1 matrix in detail, the failure was not related to DTD's theory such as the choice of root points[1] and the recursive application of the Taylor Theorem. Here, we fill this gap: *Can we identify flaws in DTD's theory that would explain the perplexing behavior of ignoring the network's parameters? Does DTD provide transparency regarding its assumptions and guarantees about the explanation's correctness?*

Before we approach these questions, we summarize the relevant background of the Deep Taylor Decomposition in Section 3. For completeness, we start with the well-known Taylor theorem and then discuss how the theorem connects to DTD's relevances. We then continue with stating the recursive application of the Taylor Theorem formally and recapitulate the so-called *train-free DTD* approximation, which allows to compute layer-wise relevances efficiently.

In section 4, we present our theoretical analysis of DTD. In particular, we contribute: **(C1)** a proof that the root points must be contained in the same linear region as the input; **(C2)** we generalize a previous observation about LRP_0 (Shrikumar et al., 2016): if the layers' root points are chosen locally constant w.r.t the layers' input, then DTD's relevances take a similar form as input×gradient; **(C3)** DTD is underconstrained: if the root points depend on the layers' input, then the Deep Taylor Decomposition can be used to create any arbitrary explanation; **(C4)** we also find that DTD cannot be extended easily to analytic activation functions (e.g. Softplus), without introducing complex higher-order derivatives with the same order as the number of network layers.

In an empirical evaluation (Section 5), we applied the theoretical insights from the previous section and studied the *train-free DTD* approximation in several experiments: **(C5)** The train-free DTD does not enforce the root points to be located in the valid local linear region of the network; **(C6)** We also validated this empirically using a small multilayered perceptron, where we found a substantial number of samples having roots located outside the valid local linear region; **(C7)** Additionally, we include a reproducibility study of (Arras et al., 2022) that claimed that DTD's explanations would not suffer from the problems reported in Sixt et al. (2020). This reproducibility study also highlights DTD's black-box character and how difficult it is to evaluate explanation quality empirically.

Given the theoretical and empirical evidence, we conclude that DTD obscures its simplifications and violates its own assumptions. DTD is underconstrained and even allows justifying virtually any explanation.

2 Related Work

The theoretical analysis of explanation methods is a small research area. PatternAttribution (Kindermans et al., 2018) investigated the insensitiveness of DTD rules to input noise and then proposed a way to learn

[1]Following Montavon et al. (2017), we name the points used for the Taylor Theorem *root points*. For example, for a function $f : \mathbb{R} \to \mathbb{R}$, the first-order Taylor approximation is $f(x) \approx f(\tilde{x}) + f'(\tilde{x})(x - \tilde{x})$, where $\tilde{x}$ is the root point.

the root points from data. Other lines of work are the manipulation of saliency maps (Dombrowski et al., 2019; Viering et al., 2019; Wang et al., 2020), the runtime-complexity of explanation methods (Waeldchen et al., 2021), or the explanations methods with provable guarantees (Chen et al., 2019).

Previous works have analyzed the theoretical properties of various saliency methods. For example, the insenstivity of Guided-Backprop (Springenberg et al., 2014) was analyzed in (Nie et al., 2018), and (Lundstrom et al., 2022) found flaws in the theoretical motivation of integrated gradients Sundararajan et al. (2017). (Kumar et al., 2020) discussed the issues from an independence assumption between input variables, often introduced in sampling algorithms (Štrumbelj & Kononenko, 2011; 2014; Lundberg & Lee, 2017). In (Shah et al., 2021), it was empirically analyzed and proven for a specific dataset that the Gradient's magnitude will not correspond to relevant features. Our work also analyzes the theoretical properties of saliency methods but differs from previous works as it focuses on the DTD and LRP methods.

Although different review articles (Montavon et al., 2018; 2019; Samek et al., 2021b;a) and extensions of LRP and Deep Taylor (Binder et al., 2016; Kohlbrenner et al., 2020; Hui & Binder, 2019; Ali et al., 2022) have been published, none discussed the theoretical issues brought forward in our manuscript.

3 Background

In this section, we provide the necessary background on Deep Taylor Decomposition to understand the theoretical analysis in Section 4. We mainly reproduce the derivations given in (Montavon et al., 2017; 2018). If we comment on the derivations, we do this in the *Remark* sections.

3.1 Taylor Theorem for multivariate functions

Taylor Theorem for multivariate functions can be concisely stated using multi-index notation. A multi-index $\boldsymbol{\alpha} \in \mathbb{N}_0^k$ is a vector of non-negative integers ($\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_k]$). The following operations are defined as: $\boldsymbol{\alpha}! = \alpha_1!\alpha_2! \ldots \alpha_k!$, $|\boldsymbol{\alpha}| = \sum_{i=1}^{k} \alpha_i$, $\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_k^{\alpha_k}$, and $\partial^{\boldsymbol{\alpha}} f = \partial^{|\boldsymbol{\alpha}|} f / (\partial^{\alpha_1} x_1 \partial^{\alpha_2} x_2 \ldots \partial^{\alpha_k} x_k)$, where $\boldsymbol{x} \in R^k$ and $f : \mathbb{R}^k \to \mathbb{R}$. The following theorem is adapted from Folland (2002, Theorem 2.68):

Theorem 1 (Multivariate Taylor Theorem)**.** *Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is of differentiability class C^k on an open convex set S. If $\boldsymbol{x} \in S$ and $\tilde{\boldsymbol{x}} \in S$, then:*

$$f(\boldsymbol{x}) = \sum_{|\boldsymbol{\alpha}| \leqslant k} \frac{\partial^{\boldsymbol{\alpha}} f(\tilde{\boldsymbol{x}})}{\boldsymbol{\alpha}!} (\boldsymbol{x} - \tilde{\boldsymbol{x}})^{\boldsymbol{\alpha}} + g_k(\boldsymbol{x}, \tilde{\boldsymbol{x}}), \tag{1}$$

where the remainder is given by:

$$g_k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = k \sum_{|\boldsymbol{\alpha}| = k} \frac{(\boldsymbol{x} - \tilde{\boldsymbol{x}})^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} \int_0^1 (1-t)^{k-1} \Big[\partial^{\boldsymbol{\alpha}} f\big(t\boldsymbol{x} + (1-t)\,\tilde{\boldsymbol{x}}\big) - \partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}) \Big] dt. \tag{2}$$

As the Deep Taylor Decomposition focuses on neural networks with ReLU activations, we will mainly look at the first-order Taylor Theorem:

$$f(\boldsymbol{x}) = f(\tilde{\boldsymbol{x}}) + \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x} = \tilde{\boldsymbol{x}}} \cdot (\boldsymbol{x} - \tilde{\boldsymbol{x}}) \tag{3}$$

where $\tilde{\boldsymbol{x}}$ is the root point, and $|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}}$ denotes the gradient evaluated at the root point $\tilde{\boldsymbol{x}}$. The higher order terms are zero due to the local linearity of ReLU networks. As the Taylor Theorem requires $f \in C^1$ (i.e., all partial derivatives $\partial f(\boldsymbol{x}) / \partial \boldsymbol{x}$ must be continuous in the local neighborhood S), the root point must be within the same linear region as the input.

Definition 1 (Linear Region)**.** A linear region of a function $f : \mathbb{R}^d \to \mathbb{R}$ is the set $N_f(\boldsymbol{x})$ of all points $\boldsymbol{x'} \in N_f(\boldsymbol{x})$ that (1) have the same gradient at $\boldsymbol{x}$: $\nabla f(\boldsymbol{x}) = \nabla f(\boldsymbol{x'})$, and (2) can be reached from $\boldsymbol{x}$ without passing through a point $\boldsymbol{a}$ with a different gradient, i.e., $\nabla f(\boldsymbol{a}) \neq \nabla f(\boldsymbol{x})$.

In case of a ReLU network, the approximation error cannot be bounded when selecting a root point $\tilde{\boldsymbol{x}} \notin N_f(\boldsymbol{x})$, as that linear region's gradient might differ substantially.

3.2 Taylor Theorem and Relevances

In the previous section, we have recapitulated the Taylor Theorem. We now discuss how the Taylor Theorem can be used to compute input relevances. A common approach explaining deep neural network is to contrast the network's output with a similar point predicted differently. A user could then study pairs $(\boldsymbol{x}, f(\boldsymbol{x}))$ and $(\tilde{\boldsymbol{x}}, f(\tilde{\boldsymbol{x}}))$ and relate the input differences to the output differences. To guide the user's attention, it would be desirable to highlight which changes between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ were responsible for the difference in the output. The first observation in (Montavon et al., 2017) is that the network's output differences can be redistributed to the input by using the Taylor Theorem. If the point $\tilde{\boldsymbol{x}}$ is in the local neighborhood $N_f(\boldsymbol{x})$, we can use the first-order Taylor Theorem (equation 3) to write the difference $f(\boldsymbol{x}) - f(\tilde{\boldsymbol{x}})$ as:

$$f(\boldsymbol{x}) - f(\tilde{\boldsymbol{x}}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}} \cdot (\boldsymbol{x} - \tilde{\boldsymbol{x}}) \tag{4}$$

The relevance of the input $R : \mathbb{R}^d \to \mathbb{R}^d$ is then defined to be the point-wise product of the partial derivatives with the input differences:

$$R(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}} \odot (\boldsymbol{x} - \tilde{\boldsymbol{x}}) \tag{5}$$

While this would be a simple way to compute the relevances, the following reasons are given in (Montavon et al., 2017; 2019), to not directly use the Taylor Theorem on the network output:

1. **Adversarial perturbations** (Szegedy et al., 2013): Small input perturbations can lead to a large change in the output. Therefore, the difference in the output might be enormous but $|\boldsymbol{x} - \tilde{\boldsymbol{x}}|$ tiny and uninterpretable.

2. **Finding a root point might be difficult**: *"It is also not necessarily solvable due to the possible non-convexity of the minimization problem"* (Montavon et al., 2017).

3. **Shattered gradients** (Balduzzi et al. (2017)): *"While the function value $f(x)$ is generally accurate, the gradient of the function is noisy"* (Montavon et al., 2019).

Remark 1. We want to point out that the more general problem seems to be that the local linear regions are tiny, or rather the number of linear regions grows exponentially with the depth of the network in the worst case (Arora et al., 2018; Xiong et al., 2020; Montufar et al., 2014). This restricts the valid region for the root point to a small neighborhood around the input.

3.3 Deep Taylor: Recursive Application of Taylor Theorem

The main idea of (Montavon et al., 2017) is to recursively apply the Taylor Theorem to each network layer. Before we present this in detail, we first we need to clarify the notation of an n-layered ReLU network shortly:

Definition 2 (ReLU network)**.** An n-layered ReLU network $f : \mathbb{R}^{d_1} \to \mathbb{R}^{d_{n+1}}_{\geqslant 0}$ is the composition of n functions $f = f_n \circ \ldots \circ f_1$, where each function $f_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}_{\geqslant 0}$ has the form $f_l(\boldsymbol{a}_l) = [W_l \boldsymbol{a}_l]^+$, and where $[.]^+$ is the ReLU activation.

Instead of directly calculating the relevance of the input as done in the previous section, we can apply Taylor Theorem to the final network layer and then apply the Taylor Theorem again to the resulting relevance. By recursively applying the Taylor Theorem per individual layer, we can calculate the relevance of the input. As the base case of the recursive application, the relevance of the network output is set to the value of the explained logit $f_{[\xi]}(\boldsymbol{x}) = \boldsymbol{a}_{n+1_{[\xi]}}$:

$$R^{n+1}(\boldsymbol{a}_{n+1}) = \boldsymbol{a}_{n+1_{[\xi]}}, \tag{6}$$

where R^{n+1} denotes the relevance of the $n+1$-th network activation. We decided to use superscripts for the relevance functions as their individual dimensions are often index as in $R^{n+1}_{[j]}$. Suppose that we already

know the relevance function $R^{l+1}(\boldsymbol{a}_{l+1}) \in \mathbb{R}^{d_{l+1}}$ for the layer $l+1$. We can then calculate the relevance of $\boldsymbol{a}_l$ (the input to layer l) to the j-th coordinate of $R^{l+1}(\boldsymbol{a}_{l+1})$:

$$R^{l+1}_{[j]}(\boldsymbol{a}_{l+1}) = R^{l+1}_{[j]}\left(f_l(\tilde{\boldsymbol{a}}_l)\right) + \left.\frac{\partial R^{l+1}_{[j]}\left(f_l(\boldsymbol{a}_l)\right)}{\partial \boldsymbol{a}_l}\right|_{\boldsymbol{a}_l=\tilde{\boldsymbol{a}}_l(\boldsymbol{a}_l)} \cdot (\boldsymbol{a}_l - \tilde{\boldsymbol{a}}_l(\boldsymbol{a}_l)), \tag{7}$$

where we used $\boldsymbol{a}_{l+1} = f_l(\boldsymbol{a}_l)$. The root point is selected in dependency of the layers' input $\boldsymbol{a}_l$, i.e., it is a function $\tilde{\boldsymbol{a}}_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$. The total relevance of the input to layer l is given by the sum over all d_{l+1} hidden neurons.

Definition 3 (Recursive Taylor)**.** Given a function $f : \mathbb{R}^{d_1} \to \mathbb{R}^{d_{n+1}}$, which can be written as a composition of n functions $f = f_1 \circ \ldots \circ f_n$ with $f_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}$, the input to each function f_l are denoted by $\boldsymbol{a}_l$ and $\boldsymbol{a}_{n+1}$ specifics f's output. Additionally, a root point function $\tilde{\boldsymbol{a}}_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$ is defined for each layer, which must only return admissible values $\tilde{\boldsymbol{a}}_l(\boldsymbol{a}_l) \in N_{R_l}(\boldsymbol{x})$ and $\tilde{\boldsymbol{a}}_l(\boldsymbol{a}_l) \neq \boldsymbol{a}_l$. Then, the base case is given by $R^{n+1}(\boldsymbol{a}_{n+1}) = \boldsymbol{a}_{n+1_{[\xi]}}$ and the relevance function $R^l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$ of layer $l \neq n$ is recursively defined by:

$$R^l(\boldsymbol{a}_l) = \sum_{j=1}^{d_{l+1}} \left(\left.\frac{\partial R^{l+1}_{[j]}\left(f_l(\boldsymbol{a}_l)\right)}{\partial \boldsymbol{a}_l}\right|_{\boldsymbol{a}_l=\tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l)} \odot \left(\boldsymbol{a}_l - \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l)\right) \right) \tag{8}$$

The above definition corresponds to equation 6 in (Montavon et al., 2017). Except for the root point selection, which will be discussed in the following sections, definition 3 contains all information to implement the recursive decomposition using an automatic differentiation library. An exemplary pseudo-code can be found in Algorithm 1. Before continuing with the approximations of the Deep Taylor Decomposition, we want to make a few remarks:

Remark 2 (No Axiomatic Motivation)**.** Only some vague arguments are provided to motivate the recursive application of the Taylor Theorem:

> *The deep Taylor decomposition method is inspired by the divide-and-conquer paradigm, and exploits the property that the function learned by a deep network is decomposed into a set of simpler subfunctions, either enforced structurally by the neural network connectivity, or occurring as a result of training.* – Montavon et al. (2017, Section 3)

In contrast, Shapely values (Shapley, 1951) are motivated by four axiomatic properties, which are uniquely fulfilled by the Shapely values. A comparable set of axioms with uniqueness result does not exist for the Deep Taylor Decomposition.

3.4 Deep Taylor Decomposition For A One-Layer Network and DTD's rules

In the previous section, we introduced the recursive application of the Taylor Theorem. For a concrete example, we will now discuss how DTD is applied to a one-layered network. It will also explain the propagation rules of DTD. This subsection corresponds to Section 4 in Montavon et al. (2017), and we will refer to the corresponding equations with the notation (DTD eq. 11).

The one-layered network consists of a linear layer with a ReLU activation followed by a sum-pooling layer, i.e. $f : \mathbb{R}^d \to \mathbb{R}$, $f(\boldsymbol{x}) = \sum_j [W\boldsymbol{x} + \boldsymbol{b}]^+_j$, where $[.]^+$ is the ReLU activation. We will denote the output of the ReLU layer as $\boldsymbol{h}(\boldsymbol{x}) = [W\boldsymbol{x} + \boldsymbol{b}]^+$, and the sum-pooling layer as $y(\boldsymbol{h}) = \sum_j \boldsymbol{h}_j$. For this subsection, we will denote the relevance function with $R^{\boldsymbol{x}}$, $R^{\boldsymbol{h}}$, and R^y for the input, hidden, and output layer, respectively.

The relevance of the final layer is simply given by the network's output (equation 6; DTD eq. 8):

$$R^y(\boldsymbol{h}(\boldsymbol{x})) = y(\boldsymbol{h}(\boldsymbol{x})). \tag{9}$$

DTD suggests to select a root point $\tilde{\boldsymbol{h}}$ such that $y(\tilde{\boldsymbol{h}}) = 0$. The advantage of $y(\tilde{\boldsymbol{h}}) = 0$ is that the network's output $y(\boldsymbol{h})$ is absorbed to the first-order term (i.e., $f(\tilde{\boldsymbol{x}}) = 0$ in equation 3). We then have $y(\boldsymbol{h}) = \frac{\partial R^y(\tilde{\boldsymbol{h}})}{\partial \tilde{\boldsymbol{h}}} \cdot (\boldsymbol{h} - \tilde{\boldsymbol{h}})$ such that the network output is fully redistributed to the hidden layer's relevance. Additionally,

the root point should be a valid input to the layer. As $y(\boldsymbol{h})$'s input comes from the ReLU layer $\boldsymbol{h}(\boldsymbol{x})$, it is positive and only $\tilde{\boldsymbol{h}} = 0$ solves $\sum_j \tilde{\boldsymbol{h}}_j = 0$. The derivative $\partial R^f_{\boldsymbol{h}}(\tilde{\boldsymbol{h}})/\partial\tilde{\boldsymbol{h}} = \partial y(\tilde{\boldsymbol{h}})/\partial\tilde{\boldsymbol{h}} = 1$ and therefore, we can use equation 8 to write the relevance of the ReLU layer's output as (DTD eq. 10):

$$R^{\boldsymbol{h}}(\boldsymbol{x}) = \frac{\partial R^y(\boldsymbol{h})}{\partial \boldsymbol{h}}\Big|_{\tilde{\boldsymbol{h}}=0} \odot \boldsymbol{h} = \boldsymbol{h} \tag{10}$$

As $\boldsymbol{h}$ is the ReLU output, we can write $R^{\boldsymbol{h}}(\boldsymbol{x})$ also as (DTD eq. 11):

$$R^{\boldsymbol{h}}(\boldsymbol{x}) = [W\boldsymbol{x} + \boldsymbol{b}]^+ \tag{11}$$

The next step is to connect the input relevance $R^{\boldsymbol{x}}(\boldsymbol{x})$ with the ReLU neurons' relevances $R^{\boldsymbol{h}}$. We will use equation 8 and apply the Taylor theorem to the relevance of each hidden neuron $\boldsymbol{h}_{[j]}$:

$$R^{\boldsymbol{x}}(\boldsymbol{x}) = \sum_{j=1}^{d} \left(\frac{\partial R^{\boldsymbol{h}}_{[j]}(\boldsymbol{x})}{\partial \boldsymbol{x}}\bigg|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}^{(j)}} \odot (\boldsymbol{x} - \tilde{\boldsymbol{x}}^{(j)}) \right) = \sum_{j=1}^{d} \left(\boldsymbol{w}_j \odot (\boldsymbol{x} - \tilde{\boldsymbol{x}}^{(j)}) \right), \tag{12}$$

where we used that the derivative of the hidden neuron $\boldsymbol{h}_{[j]}$ w.r.t the input is the weight vector $\boldsymbol{w}_j = W_{[j:]}$.

Relevance Propagation Rules For the root point $\tilde{\boldsymbol{x}}$, we could, in theory, select any point in the half-space $\boldsymbol{w}_j\tilde{\boldsymbol{x}} + b_j > 0$, as they are all valid according to the Taylor Theorem. However, as it is beneficial to fully redistribute the relevance, DTD proposed selecting a point that sets the j-th neuron relevance to zero, i.e., any point on the hyperplane $\boldsymbol{w}_j^T\tilde{\boldsymbol{x}} + \boldsymbol{b}_j = 0$. The non-differentiability at the ReLU hinge is resolved by picking the gradient from the case $\boldsymbol{w}_j\tilde{\boldsymbol{x}} + b_j > 0$.

As there is no unique solution, DTD derives different points by starting at the input $\boldsymbol{x}$ and moving along a direction $\boldsymbol{v}_j$ such that $\tilde{\boldsymbol{h}}_j = \boldsymbol{x} - t\boldsymbol{v}_j$ with $t \in \mathbb{R}$. The root point $\tilde{\boldsymbol{x}}_j$ is then the intersection of the line $\boldsymbol{x} + t\boldsymbol{v}_j$ with the hyperplane $\boldsymbol{w}_j^T\boldsymbol{x} + \boldsymbol{b}_j = 0$. Combining these equations yields $t = -\frac{\boldsymbol{w}_j^T\boldsymbol{x}+\boldsymbol{b}_j}{\boldsymbol{w}_j^T\boldsymbol{v}_j}$ and therefore the root point $\tilde{\boldsymbol{x}}^{(j)} = \boldsymbol{x} - \frac{\boldsymbol{w}_j^T\boldsymbol{x}+\boldsymbol{b}_j}{\boldsymbol{w}_j^T\boldsymbol{v}_j}\boldsymbol{v}_j$. Substituting this into equation 12 yields: (DTD-Appendix eq. 6, 7)

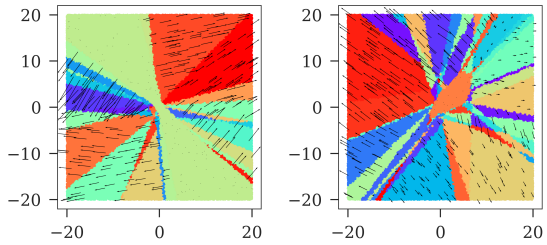
$$R^{\boldsymbol{x}}(\boldsymbol{x}) = \sum_{j=1}^{d} \left(\boldsymbol{w}_j \odot \frac{\boldsymbol{w}_j^T\boldsymbol{x} + \boldsymbol{b}_j}{\boldsymbol{w}_j^T\boldsymbol{v}_j}\boldsymbol{v}_j \right) = \sum_{j=1}^{d} \left(\boldsymbol{w}_j \odot \frac{\boldsymbol{v}_j}{\boldsymbol{w}_j^T\boldsymbol{v}_j} R^{\boldsymbol{h}}_{[j]}(\boldsymbol{x}) \right). \tag{13}$$

While almost all choices of $\boldsymbol{v}$ yield a root point with $R^{\boldsymbol{x}}(\tilde{\boldsymbol{x}}) = 0$ (except $\boldsymbol{v} \perp \boldsymbol{w}$), a few special directions exists:

- *The w^2-rule* chooses the closest root point in L2 metric: $\boldsymbol{v}_j = \boldsymbol{w}_j$. This will yield the root point $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \frac{\boldsymbol{w}_j}{\boldsymbol{w}_j^T\boldsymbol{w}_j}R^h(x)$ and the following relevance propagation rule: $R^x(\boldsymbol{x}) = \sum_{j=1}^{d} \frac{\boldsymbol{w}_j^2}{\boldsymbol{w}_j^T\boldsymbol{w}_j}R^{\boldsymbol{h}}(\boldsymbol{x})$, where $\boldsymbol{w}_j^2 = \boldsymbol{w}_j \odot \boldsymbol{w}_j$.
- *The z^+ rule* uses a direction that always yields a positive root: $\boldsymbol{v}_j = \mathbb{1}_{\boldsymbol{w}_j \geqslant 0}\boldsymbol{x}$, which is preferred for positive inputs (e.g. from ReLU activations). The resulting root point is $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \frac{\boldsymbol{w}\mathbb{1}_{\boldsymbol{w}_j \geqslant 0}\boldsymbol{x}}{\boldsymbol{w}_j^T(\mathbb{1}_{\boldsymbol{w}_j \geqslant 0}\boldsymbol{x})}R^h_{[j]}(x)$ and the following relevance propagation rule: $R^x(\boldsymbol{x}) = \sum_{j=1}^{d} \frac{\boldsymbol{z}_j^+}{\sum_i \boldsymbol{z}_{ji}^+} R^{\boldsymbol{h}}_{[j]}(\boldsymbol{x})$, where $\boldsymbol{z}_j^+ = \mathbb{1}_{\boldsymbol{w}_j \geqslant 0}\boldsymbol{x} \odot \boldsymbol{w}_j^+$.
- *The gamma rule* proposed in Montavon et al. (2019) uses the search direction $\boldsymbol{v}_j = 1 + \gamma\mathbb{1}_{\boldsymbol{w}_j \geqslant 0} \odot \boldsymbol{x}$, where $\gamma \in \mathbb{R}^+$. The corresponding relevance propagation rule is then: $R^x(\boldsymbol{x}) = \sum_{j=1}^{d} \frac{\boldsymbol{w}_j + \gamma\boldsymbol{z}_j^+}{\boldsymbol{w}_j^T(1+\gamma\boldsymbol{z}_j^+)} R^{\boldsymbol{h}}_{[j]}(\boldsymbol{x})$. In the limit $\gamma \to \infty$, the gamma rule becomes the z^+-rule.
- A special case is the *LRP_0* rule which does not use any vector to find a root point but chooses $\tilde{\boldsymbol{x}} = 0$. Although zero is not a valid root point in general, it was shown that LRP_0 corresponds to gradient×input Shrikumar et al. (2016); Ancona et al. (2018); Kindermans et al. (2016). The LRP_ε rule is an extension of LRP_0 that adds a small ε to increase numeric stability.

3.4.1 Which rule should be chosen?

The computed relevance values depend substantially on the rule. For example, the LRP_0 rule can compute negative relevance values, whereas the z^+ rule will always return positive relevance values. In Montavon et al. (2017, Sec. 4.1, 4.2, 4.3), the input domain was used as the primary selection criterion. For example, it was suggested to pick z^+ rule for $\mathbb{R}_0^+$ and the w^2 rule for the domain $\mathbb{R}$. The input domain is not a sufficient selection for the root points, as it does not provide a unique solution. In the later work Montavon et al. (2019), other selection criterion were proposed for deep neural networks, which we will analyze in Section 3.5.1. For now, we can conclude that no principled way to pick the roots and rules exists.



(a) Non-positive biases (b) Unrestricted biases

Figure 1: Local linear regions of an randomly initialized neural network (3 layers, ReLU, 2 inputs, 10 hidden neurons). The biases are initialized **(a)** non-positive and **(b)** unrestricted. The gradient are visualized as arrows for a random selection of points.

3.4.2 Non-positive biases

In Montavon et al. (2017), it was proposed to constrain the biases of the linear layer to be non-positive, i.e. $b_j \leqslant 0$. The main motivation was to guarantee that the origin is a root point of the function f. However, this is not the case, as the following simple counter-example will show. Suppose the bias $b = -\vec{1}$. Then the function $f(\vec{0}) = 0$, as $[W\vec{0} - \vec{1}]^+ = 0$, but the origin $\vec{0}$ is not a valid root point as the gradient is zero there. However, any input $\boldsymbol{x}$ with $f(\boldsymbol{x}) \geqslant 0$ will have a non-zero gradient and will therefore be in a different local region. In Figure 1, we visualized the local regions of a small 3-layered network for non-positive and unrestricted bias.

3.5 DTD for Deep Neural Neworks: The Training-Free Relevance Model

Applying the recursive Taylor decomposition to a one-layered network yielded a set of easily applied relevance propagation rules, which allowed to skip computing the root points explicitly. Of course, it would be desirable to skip the computation of roots for deep neural networks too. As solution, Montavon et al. (2017) proposed a so-called *training-free* relevance model. We follow the derivation from the review article (Montavon et al., 2018).

Let $R^{l+1}(\boldsymbol{a}_l)$ be the relevance computed for an upper layer. Montavon et al. (2017) then makes the following assumption:

Assumption 1 (Positive Linear Relevance)**.** The relevance of the upper layer $R^{l+1}(\boldsymbol{a}_{l+1})$ can be written as $R^{l+1}(\boldsymbol{a}_{l+1}) = \boldsymbol{a}_{l+1} \odot \boldsymbol{c}_{l+1}$, where $\boldsymbol{c}_{l+1} \in \mathbb{R}^+$ should be a constant and positive vector.

As $R^{l+1}(\boldsymbol{a}_l) = \boldsymbol{c}_{l+1} \odot [W_l \boldsymbol{a}_l + b_l]^+$, we can construct a so-called relevance neuron:

$$\hat{R}^{l+1}(\boldsymbol{a}_{l+1}) = [\hat{W}_{l+1}\boldsymbol{a}_{l+1} + \hat{\boldsymbol{b}}_{l+1}]^+, \tag{14}$$

where we pulled $\boldsymbol{c}$ into the layer's parameters: $\hat{W}_{l+1} = W_{l+1} \odot C_{l+1}$ and where $C_{l+1} = [\boldsymbol{c}_{l+1}, \ldots, \boldsymbol{c}_{l+1}]$ is a repeated version of $\boldsymbol{c}_{l+1}$ and $\hat{\boldsymbol{b}}_{l+1} = \boldsymbol{b}_{l+1} \odot \boldsymbol{c}_{l+1}$.

This formulation is similar to the relevance of the hidden layer of the one-layer network in equation 11. The difference is that the root point and search direction will be based on the modified weights $\hat{W}_{l+1}$ and $\hat{\boldsymbol{b}}_{l+1}$. Using $\hat{\boldsymbol{w}}_j = \hat{W}_{l+1_{[j:]}} = \boldsymbol{c}_{l+1_{[j]}} W_{l+1_{[j:]}}$, we can write the general relevance propagation rule of equation 13 as:

$$\hat{R}^l(\boldsymbol{a}_l) = \sum_{j=1}^{d} \left(\frac{\hat{\boldsymbol{w}}_j \odot \boldsymbol{v}_j}{\hat{\boldsymbol{w}}_j^T \boldsymbol{v}_j} \hat{R}^{l+1}_{[j]}(f_l(\boldsymbol{a}_l)) \right) = \sum_{j=1}^{d} \left(\frac{\boldsymbol{w}_j \odot \boldsymbol{v}_j}{\boldsymbol{w}_j^T \boldsymbol{v}_j} \hat{R}^{l+1}_{[j]}(f_l(\boldsymbol{a}_l)) \right), \tag{15}$$

where the $\boldsymbol{c}_{l+1_j}$ canceled out. The corresponding root point would be: $\tilde{\boldsymbol{a}}_l^{(j)} = \boldsymbol{a}_l - \frac{\boldsymbol{w}_j^T \boldsymbol{x} + \boldsymbol{b}_j}{\boldsymbol{w}_j^T \boldsymbol{v}_j} \boldsymbol{v}_j$. Interestingly, this deviation recovers the one-layer case from equation 13. Thus, Montavon et al. (2018) argues that all

the rules from the linear case (Section 3.4) can be applied to a deep neural network too. This result can be easily extended to sum-pooling layers, as they are equivalent to a linear layer with the weights of value 1.

Remark 3 (Is it correct that $\boldsymbol{c}_k$ is constant?)**.** A global constant $\boldsymbol{c}_k$ cannot exist, as changing the input vector can result a in totally different output, which would change the relevance magnitude. A local approximation of $\boldsymbol{c}_k$ could be correct if root points stays within the same local linear region where the function's gradient ∇f is locally constant.

Remark 4 (**C5** no mechanism to enforce that the root point $\tilde{\boldsymbol{a}}_l^{(j)} \in N_{R^l}(\boldsymbol{a}_l)$?)**.** The corresponding root point to equation 15 would be: $\tilde{\boldsymbol{a}}_l^{(j)} = \boldsymbol{a}_l - \frac{\boldsymbol{w}_j^T \boldsymbol{x} + \boldsymbol{b}_j}{\boldsymbol{w}_j^T \boldsymbol{v}_j} \boldsymbol{v}_j$. Will this root point be in the local region $N_f(\boldsymbol{a}_l)$ of $\boldsymbol{a}_l$? Probably not, as there is no mechanism enforcing this. We test this in more detail in the empirical evaluation.

3.5.1 Which Root Points To Choose For A Deep Neural Network?

In Section 3.4.1, we already discussed that there is no principled way to select the root points or corresponding rules. For deep neural networks, DTD Montavon et al. (2017) originally proposed to pick the z^+-rule for all layers except the first one. In a more recent work Montavon et al. (2019), it was argued to use a combination of LRP_0, LRP-ε and the γ-rule. This was motivated by rather vague properties such as the "activations' entanglement", "spurious variations", and "spreading of relevance". It is concluded that: "*Overall, in order to apply LRP successfully on a new task, it is important to carefully inspect the properties of the neural network layers, and to ask the human what kind of explanation is most understandable for him.*" – Montavon et al. (2019, Sec. 10.3). Thus, the choice of the rules lies in the hands of the user who might choose any rule or root point.

Kohlbrenner et al. (2020) introduced a similar combination of rules as *LRP-Composite*. For the convolution layers, they used the z^+-rule (or the $\text{LRP}_{\alpha 1 \beta 0}$) and for the fully-connected layers the LRP_0. An improvement over using the z^+-rule for all layers, they found that this combination of rules did not suffer from class-insensitivity, i.e., the saliency map do change when the explained output class is changed. However, it must be noted that this combination relies on the particular properties of the convolutional neural network. Specifically, there is little information mixing between more distant locations. Furthermore, the explanations are still insensitive to the later convolutional layers: the z^+ rule creates a fixed saliency map for the convolutional layers, which, however, can be scaled by the output of the LRP_0-rule. For example, if the final convolutional output has shape (8, 8) than the saliency map can be scaled in an 8x8 grid.

4 Analysis of the Recursive Application of the Taylor Theorem

In the previous sections, we recapitulated how DTD applies the Taylor Theorem recursively to a one-layer and a deep neural network, and explained how the different propagation rules were derived. In this section, we provide a theoretical analysis of the recursive application of the Taylor Theorem. In particular, we study the definition 3 from Section 3.3. As this definition is the most general formulation of the DTD theory, we ensure that the results of our analysis are applicable to all the propagation rules and are also not caused by one specific approximation but are rather inherent to the recursive application of the Taylor theorem.

The following propositions are proven in the Appendix A. The main idea of the proof is to apply the product rule to equation 8 and then analyze the individual terms.

4.1 Size of admissable regions for the root points cannot be increased

Proposition 1 (**C1**: Recursively applying the Taylor Theorem cannot increase the size of admissible regions)**.** *Given a ReLU network $f : \mathbb{R}^{d_1} \to \mathbb{R}^{d_{n+1}}_{\geqslant 0}$, recursive relevance functions $R^l(\boldsymbol{a}_l)$ with $l \in \{1, \dots, n\}$ according to definition 3, and let ξ index the explained logit, then it holds for the admissible region $N_{R^l}(\boldsymbol{a}_l)$ for the root points $\tilde{\boldsymbol{a}}_l^{(j)}$ of the relevance function R^l that $N_{R^l}(\boldsymbol{a}_l) \subseteq N_{f_{n_\xi} \circ \dots \circ f_l}(\boldsymbol{a}_l)$.*

As the valid region for root points is restricted by the network f, we then we cannot evade the local region. This motivates a simple empirical test in Section 5.1: for each root point, we can check whether it is contained

in the correct admissible region. This result questions the motivation that the distance $|\boldsymbol{x} - \tilde{\boldsymbol{x}}|$ might be small T from 3.2, as this distance remains bounded by the local linear region of the network.

4.2 Locally Constant Roots Imply Equivalence of Recursive Taylor and Gradient×Input

It is well known that LRP_0 is equivalent to gradient×input for ReLU networks. This was first noted in (Shrikumar et al., 2016) and later also in (Kindermans et al., 2016; Ancona et al., 2018). We proof the following generalization for the recursive application of the Taylor Theorem in Appendix A.1.

Proposition 2 (C2). *Let $f : \mathbb{R}^{d_1} \to \mathbb{R}_{\geqslant 0}^{d_{n+1}}$ be a ReLU network, ξ be the index of the explained logit, and $R^l(\boldsymbol{a}_l)$ (with $l \in 1 \dots n+1$) are recursive relevance functions according to definition 3. If the root points $\tilde{\boldsymbol{a}}_l(\boldsymbol{a}_l)$ are locally constant w.r.t. the layer's input ($\forall l \in 1 \dots n : \partial \tilde{\boldsymbol{a}}_l / \partial \boldsymbol{a}_l = 0$), then:*

$$R(\boldsymbol{x}) = R(\tilde{\boldsymbol{x}}) + \nabla f_{[\xi]}(\boldsymbol{x}) \odot (\boldsymbol{x} - \tilde{\boldsymbol{x}}), \tag{16}$$

where $\boldsymbol{x} = \boldsymbol{a}_1$ is the input vector and $R(\boldsymbol{x}) = R^1(\boldsymbol{x})$.

The similarity with gradient×input can be seen when choosing a root point $\tilde{\boldsymbol{x}} = \mathbf{0}$ such that $R(\mathbf{0}) = \mathbf{0}$. Then, the resulting relevance would be $\nabla f_{[\xi]}(\boldsymbol{x}) \odot \boldsymbol{x}$.

A fixed root point for each linear region would be a valid and even desirable choice. For example, from an efficiency perspective, it would be preferable to search for a valid root point in each linear region only once. Or one might want to select the one root point corresponding to the lowest network output. We also want to emphasize that no continuous constraint for selecting the root points exists. Jumps at the boundaries between the linear region are allowed. This result contradicts DTD's motivation described in Section 3.2, as it explicitly aimed to find something more "stable" than the gradient.

4.3 Locally dependent root points

As a next case, we will look at the more general case of root points depending locally on the layer's input:

Proposition 3 (C3). *For a ReLU network $f : \mathbb{R}^{d_1} \to \mathbb{R}_{\geqslant 0}^{d_{n+1}}$ with n layers, and layer activations $\boldsymbol{a}_l = f_{l-1}(\boldsymbol{a}_{l-1})$, the relevance functions $R^{l-1}(a_{l-1})$ of the recursive applications of the Taylor Theorem as given in equation 8 can be written as:*

$$R^{l-1}(a_{l-1}) = \sum_{j=1}^{d_l} \sum_{m=1}^{d_{l+1}} \left[\left(\frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_{l_{[j]}}} - \frac{\partial \tilde{\boldsymbol{a}}_{l_{[j]}}^{(m)}(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} \cdot \frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} \right) \cdot \frac{\partial R_{[m]}^{l+1}\left(f_l(\boldsymbol{a}_l)\right)}{\partial f_l(\boldsymbol{a}_l)} \right] \odot \left(\boldsymbol{a}_{l-1} - \tilde{\boldsymbol{a}}_{l-1}^{(j)}(\boldsymbol{a}_{l-1}) \right), \tag{17}$$

The relevance function R^{l-1} is determined by the next layer's Jacobian $\partial f_l(\boldsymbol{a}_l)/\partial \boldsymbol{a}_l$, and also a term including root point Jacobian $\partial \tilde{\boldsymbol{a}}_l^{(j)}/\partial \boldsymbol{a}_l$. Although some directions are recommended, the choice of root point is not restricted per se. It is merely recommended to choose it within the layer's input domain[2] and it should minimize the explained relevance. Any root point could be chosen, as long as it is from the linear region $N_{R_{[k]}^l}(\boldsymbol{a}_l)$. However, this also means that $R^{l-1}(\boldsymbol{a}_{l-1})$ can be influenced arbitrarily by the root point's Jacobian. Therefore, any explanation could be justified. A theory under which anything can be justified is clearly insufficient.

4.4 Why Not Use Analytic Activation Functions (Softplus)?

For ReLu networks, the Deep Taylor Decomposition suffers from the problem that the root point must be from the local linear region around the layer input $\boldsymbol{a}_l$. A possible solution would be to use an analytic activation function, e.g. the Softplus activation. This would allow to choose any root point in $\mathbb{R}^{d_l}$, although a sufficiently good approximation might require an unreasonable amount of higher-order terms. The main obstacle would be that with each decomposition, higher-order derivatives are accumulated:

[2] In Montavon et al. (2017, Section 4.1.), the different rules were selected based on the input domain. However, the γ rule, introduced in a more recent work Montavon et al. (2019), can lead to root points outside the ReLU's input domain $\mathbb{R}^+$, e.g. for $\gamma = 0$ the root point is given by $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \frac{\boldsymbol{x}}{\boldsymbol{w}^T \boldsymbol{x}} R^l(\boldsymbol{x})$ which can become negative for large relevance values $R^l(\boldsymbol{x})$.

Table 1: Empirical results of different DTD rules on a small neural network (3 layers, 10 input dimensions, 10 hidden dimensions). They show that the root points picked by the rules are not within the local region of the input, as each rule produced outputs below 100%. It is also the case that some root points will have the exact same network output as the original input.

Evaluation \ Rule	LRP_0	$\gamma = 1.0$	w^2	z^+
Same local linear region [expected 100%]	41.20%	38.70%	37.70%	41.10%
Same network output [expected 0%]	14.62%	13.85%	14.01%	13.99%

Proposition 4 (C4). *Let $f : \mathbb{R}^{d_1} \mapsto \mathbb{R}^{d_{n+1}}$ be a neural network, contains an analytic activation function, then each recursive application of the Taylor Theorem yields a higher-order derivative of the form:*

$$\frac{\partial}{\partial \boldsymbol{a}_l} \cdot \left[\frac{\partial R_{[j]}^{l+1}\left(f_l(\boldsymbol{a}_l)\right)}{\partial \boldsymbol{a}_l} \right]_{\boldsymbol{a}_l = \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l)} \cdot \left\{ \boldsymbol{a}_l - \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l) \right\}_{[k]} . \tag{18}$$

Thus, for a n-layered network, we would get n-ordered derivatives. The problem is that it is unclear how these chains of higher-order derivatives behave.

5 Experiments

5.1 (C6) DTD-Train-Free

We implemented the train-free DTD using an explicit computation of the root points. The network consists of 3 linear layers, each with a ReLU activation. The input and each layer has 10 dimensions. We initialized the network with random weights and used non-positive biases, as Montavon et al. (2017) suggested (even though we have shown that this has not the same consequences as claimed in Montavon et al. (2017). As we are only interested in disproving claims, it is acceptable to show that there exists one of neural network on which the DTD delivers inconsistent results. Therefore, we also did not train the network on any task.

We provide pseudocode for our implementation in Algorithm 2. The main simplifications of the implementation are that (1) the relevance of the higher layers is computed with the input of the layer and not the root point, and (2) the root points are computed using the search directions outlined in section 3.4. We tested our implementation against Captum's implementation of the DTD (Kokhlikyan et al., 2020) and found the deviation to be less than 1×10^{-8}.

Verifying that the two points are within the local region would require to show that the gradients are equal and that there is a path between the two points with all points on the path also having equal gradients. As the last part is more difficult to show, we only test the necessary condition of equal gradients. Therefore, we compare the gradient of the input with the gradient of the root points $|\nabla f(\boldsymbol{x}) - \nabla f(\tilde{\boldsymbol{x}})|$ on 1000 random inputs. The input points were sampled such that it has a network output greater than 0.1.

We reported the numerical results in Table 1. Less than 100% of all root points have gradient differences that are zero, thus root point exists which must be from a different local region. This violates Proposition 1, which requires all root points to be within the function's local region. Although we only show results on an exemplary 3-layered network, the situation would only be worse for more complex networks as the number of local regions increases exponentially with layer depth (Montufar et al., 2014).

As a second analysis, we tested how the root points influence the network output. One might assume that a root point will alter the network output. However, this is not always the case (see row "Same network output" in Table 1). At least, these root points do not explain the output of the neural network.

5.2 (C7) Applying Sanity Checks to (Arras et al., 2022)

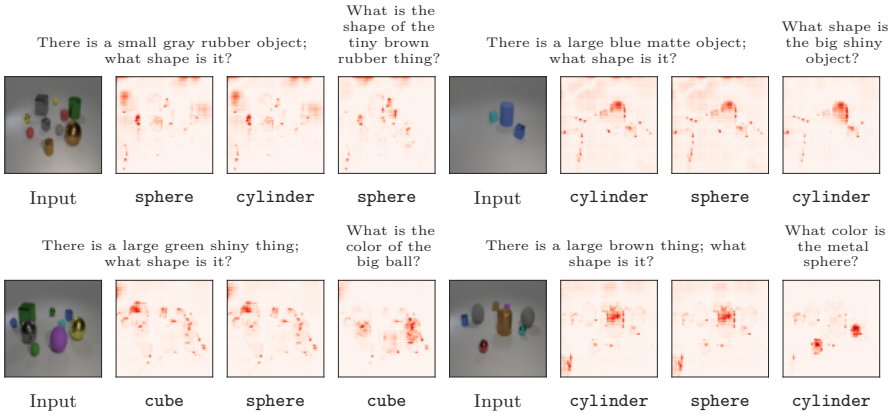


Figure 2: (1) Input images from the CLEVER-XAI dataset with the $\text{LRP}_{\alpha 1 \beta 0}$ saliency maps computed for the (2) correct class, (3) an incorrect class, and (4) a different question. The original question is written above. The saliency maps do not change visually when a different output class is explained. However, changing the question highlights other regions.

A recent work (Arras et al., 2022) evaluated different saliency methods on the CLEVR VQA dataset using ground truth segmentation masks. Interestingly, they found $\text{LRP}_{\alpha 1 \beta 0}$ (equivalent to the DTD z^+-rule) to highlight the object of interest particularly well: *"[...] a high connection between the relevance heatmaps and the target objects of each question"*. This finding seems to contradict Sixt et al. (2020), which found that $\text{LRP}_{\alpha 1 \beta 0}$ becomes independent of the network's deeper layer. In Arras et al. (2022), it was therefore concluded: *"Maybe the phenomenon described in (Sixt et al., 2020) becomes predominant in the asymptotic case of a neural network with a very high number of layers, [...]"*.

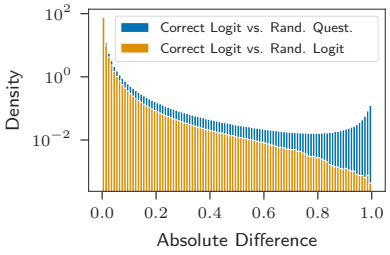


Figure 3: The histogram of absolute differences between the saliency maps for *Correct Logit vs. Random Question* and *Correct Logit vs. Random Logit*.

A simple empirical test would have been to check if $\text{LRP}_{\alpha 1 \beta 0}$'s saliency maps change when the network's last layer is changed. To perform this test, we replicated their setup and trained a relation network (Santoro et al., 2017) on the CLEVR V1.0 dataset Johnson et al. (2017). The network reached an accuracy of 93.47%, comparable to 93.3% (Arras et al., 2022) and 95.5% (Santoro et al., 2017). We included more details about the model in Appendix B.

We then compared 1000 $\text{LRP}_{\alpha 1 \beta 0}$'s saliency maps for the correct answer, an incorrect answer (but from the same category), and the correct answer but a different question. It is valid to ask for an explanation of a different class, for example, to understand which evidence is present for `sphere` instead of `cube`. The saliency maps were scaled to cover the range [0,1], and the differences were measured using the mean absolute difference. In Figure 3, a histogram of the differences is shown. While the saliency maps are very similar in both cases, there seems to be more variability in the question: for "correct logit vs. random question" there is an order of magnitude more pixels with a difference of ≈ 1. When looking at the resulting saliency maps in Figure 2, one can see that the saliency maps differ quite significantly when changing the question. In contrast, the saliency map of the wrong answer does not change.

First, these results validate the claim in (Sixt et al., 2020) that $\text{LRP}_{\alpha 1 \beta 0}$ is independent of the network's deeper layer. Second, they indicate that an information leakage between the question and $\text{LRP}_{\alpha 1 \beta 0}$'s saliency maps is present.

The reason for this information leakage can be found in the specific architecture of the relation networks: pairs are formed between all feature-map locations of the convolutional output. As the feature map has shape

$(8, 8, 24)$, $64 \cdot 64$ pairs are formed (i.e., height$^2 \cdot$width^2). Additionally, the question embedding produced by an LSTM layer is concatenated to each pair. This yields triples of $(\boldsymbol{v}_{ij}, \boldsymbol{v}_{kl}, \boldsymbol{o}_{\text{LSTM}})$, where $i, j, k, l \in \{1, \ldots, 8\}$, and $\boldsymbol{v} \in \mathbb{R}^{8\times8\times24}$ is the convolutional stack's output. As the convolutional layers and the LSTM layer are trained together, their representations are aligned. Thus, changing the LSTM embedding will change the internal representation in the subsequent layers. For relevance locations, the question embedding will match better with the convolutional activation and, therefore will lead to a higher saliency map at relevant locations. However, the saliency maps will still become independent of the network's deeper layer.

The implications are substantial: for example, if the model's final layers' were fine-tuned on a new task, the $\text{LRP}_{\alpha1\beta0}$ explanation would not change and could not be used to explain this model. Even worse, if your model was altered to predict spheres instead of cubes, the LRP explanation would not reflect this.

It is quite fascinating that the $\text{LRP}_{\alpha1\beta0}$ explanations highlight the right object according to the ground truth, but fail to highlight evidence for the wrong object. This result also shows how difficult it is to evaluate explanation methods empirically.

6 Conclusion

We have shown that DTD, which has been cited as the theoretical foundation of numerous follow-up post-hoc explanation techniques (Ali et al., 2022; Binder et al., 2016; Kindermans et al., 2018; Arras et al., 2017; Hui & Binder, 2019; Huber et al., 2019; Eberle et al., 2020), exhibits serious flaws that explain why saliency maps created with these methods are independent of the output. From Sixt et al. (2020), we know that that the positive matrices produced by the z^+-rule will converge to a rank-1 matrix. These positive matrices stem from a specific selection of the root-point, and as the selection of the root-points is not restricted, the z^+-rule can be justified by the DTD theory, as every other explanation could be - by picking an appropriate root.

DTD as a theoretical framework for explanations is under-constraint, and can be considered insufficient. Caution must be used when using explanations derived from this theory. At the core of the problem, there is no restriction and little guidance on choosing the root points. Under certain conditions (constant root points), DTD reduces to backpropagating the gradient, albeit hidden behind a complex mathematical structure. In the other case (input-dependent root points), DTD leaves open a backdoor through which virtually any explanation can be created by crafting the root point's Jacobian accordingly. However, this again is obfuscated by the theory rather than made transparent.

Since its first ArXiv submission (Montavon et al., 2015), the DTD publication has been cited numerous times. Although even the authors have reported class-insensitive behavior (Kohlbrenner et al., 2020; Montavon et al., 2018)[3], follow-up works have readily used DTD's key concepts, motivated by the seemingly robust mathematical foundation, instead of searching for the underlying reasons. Furthermore, explanations based on DTD were used in various applications, for example, for validating their model (Andresen et al., 2020), gain insights into geoscientific questions (Toms et al., 2020), or conduct user studies (Alqaraawi et al., 2020).

While we were able to discover serious issues of DTD, we do not see a solution how to solve them. We therefore want to point out that other theoretically well-justified methods exist: *(Deletion of Information)* which information can be deleted without changing the network output? One approach uses noise (Schulz et al., 2020), other discrete deletion of the input (Macdonald et al., 2019). *(Testing prediction capabilities)*: We can test whether certain concepts are present in the network by trying to predict them from intermediate features (Kim et al., 2018). *(Model inversion)*: How would the input need to change to predict another class? This question can be answered using invertible models (Hvilshøj et al., 2021; Dombrowski et al., 2021) or conditional generative models (Singla et al., 2020). *(Simple Models)*: If a similar performance is achieved by a simpler, more interpretable model, why not simply use that? For example, (Zhang et al., 2021) replaced part of the network with a linear model. All these approaches do not come with a complicated mathematical superstructure, rather they have a simple and intuitive motivation.

[3] *"A reduced number of fully-connected layers avoids that the relevance, when redistributed backwards, looses its connection to the concept being predicted."* – Montavon et al. (2018, Sec 5.1.)

Broader Impact Statement

Although our work focuses on the theoretical foundations of a particular explanation method, we see broader implications of this work. Our work demonstrates that the theoretical foundation of explanation methods need rigorous analysis before they can support the trust that developers, users, and even regulatory bodies may put in it. This is especially important in the field of explainable AI since empirically evaluating explanations is difficult.

The field offers a variety of explanation methods, and ways to test the quality of explanations. We recommend using more than just one method and employing a range of metrics and user tests to make sure explanations are helpful in potentially critical use-cases such as medical decision making or the screening of job applications.

Acknowledgement

We want to thank the reviewers for their helpful feedback that improved this manuscript further. The computation were done on the Curta cluster provided by the Zedat, Freie Universtität Berlin (Bennett et al., 2020). Finally, we thank Jonas Köhler for discussions about the manuscript idea.

References

Ameen Ali, Thomas Schnake, Oliver Eberle, Grégoire Montavon, Klaus-Robert Müller, and Lior Wolf. XAI for transformers: Better explanations through conservative propagation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 435–451. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/ali22a.html.

Ahmed Alqaraawi, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze. Evaluating saliency map explanations for convolutional neural networks: A user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, pp. 275–285, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371186. doi: 10.1145/3377325.3377519. URL https://doi.org/10.1145/3377325.3377519.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. In *International Conference on Learning Representations*, February 2018. URL https://openreview.net/forum?id=Sy21R9JAW.

Niek Andresen, Manuel Wöllhaf, Katharina Hohlbaum, Lars Lewejohann, Olaf Hellwich, Christa Thöne-Reineke, and Vitaly Belik. Towards a fully automated surveillance of well-being status in laboratory mice using deep learning: Starting with facial expression analysis. *PLoS One*, 15(4):e0228059, 2020.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding Deep Neural Networks with Rectified Linear Units. In *International Conference on Learning Representations (ICLR)*, February 2018. URL https://openreview.net/forum?id=B1J_rgWRW.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining recurrent neural network predictions in sentiment analysis. *EMNLP 2017*, pp. 159, 2017.

Leila Arras, Ahmed Osman, and Wojciech Samek. Clevr-xai: A benchmark dataset for the ground truth evaluation of neural network explanations. *Information Fusion*, 81:14–40, 2022. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2021.11.008. URL https://www.sciencedirect.com/science/article/pii/S1566253521002335.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7), 2015. ISSN 19326203. doi: 10.1371/journal.pone.0130140. URL http://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0130140&type=printable. 00067.

David Balduzzi, Marcus Frean, Lennox Leary, J. P. Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The Shattered Gradients Problem: If resnets are the answer, then what is the question? In *Proceedings of the 34th International Conference on Machine Learning*, pp. 342–350. PMLR, July 2017. URL https://proceedings.mlr.press/v70/balduzzi17b.html. ISSN: 2640-3498.

Loris Bennett, Bernd Melchers, and Boris Proppe. Curta: A general-purpose high-performance computer at zedat, freie universität berlin. http://dx.doi.org/10.17169/refubium-26754, 2020.

Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers. In *Artificial Neural Networks and Machine Learning – ICANN 2016*, Lecture Notes in Computer Science, pp. 63–71. Springer International Publishing, 2016. ISBN 978-3-319-44781-0. doi: 10.1007/978-3-319-44781-0_8.

Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1E3Ko09F7.

Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Advances in Neural Information Processing Systems*, 32, 2019.

Ann-Kathrin Dombrowski, Jan E Gerken, and Pan Kessel. Diffeomorphic explanations with normalizing flows. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021. URL https://openreview.net/forum?id=ZBR9EpEl6G4.

Oliver Eberle, Jochen Büttner, Florian Kräutli, Klaus-Robert Müller, Matteo Valleriani, and Grégoire Montavon. Building and interpreting deep similarity models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

G. B. Folland. *Advanced calculus*. Prentice Hall, Upper Saddle River, NJ, 2002. ISBN 978-0-13-065265-2.

Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. *Explainable AI Methods - A Brief Overview*, pp. 13–38. Springer International Publishing, Cham, 2022. ISBN 978-3-031-04083-2. doi: 10.1007/978-3-031-04083-2_2. URL https://doi.org/10.1007/978-3-031-04083-2_2.

Tobias Huber, Dominik Schiller, and Elisabeth André. Enhancing explainability of deep reinforcement learning through selective layer-wise relevance propagation. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pp. 188–202. Springer, 2019.

Lucas Y. W. Hui and Alexander Binder. BatchNorm Decomposition for Deep Neural Network Interpretation. In *Advances in Computational Intelligence*, volume 11507, pp. 280–291. Springer International Publishing, 2019. doi: 10.1007/978-3-030-20518-8_24. URL http://link.springer.com/10.1007/978-3-030-20518-8_24. Series Title: Lecture Notes in Computer Science.

Frederik Hvilshøj, Alexandros Iosifidis, and Ira Assent. Ecinn: Efficient counterfactuals from invertible neural networks. *ArXiv*, abs/2103.13701, 2021.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. Technical Report arXiv:1611.07270, arXiv, November 2016. URL http://arxiv.org/abs/1611.07270. arXiv:1611.07270 [cs, stat] type: article.

Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: PatternNet and PatternAttribution. February 2018. URL https://openreview.net/forum?id=Hkn7CBaTW.

Maximilian Kohlbrenner, Alexander Bauer, Shinichi Nakajima, Alexander Binder, Wojciech Samek, and Sebastian Lapuschkin. Towards best practice in explaining neural network decisions with lrp. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2020.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.

I. Elizabeth Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler. Problems with shapley-value-based explanations as feature importance measures. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5491–5500. PMLR, 2020. URL https://proceedings.mlr.press/v119/kumar20e.html.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Daniel D Lundstrom, Tianjian Huang, and Meisam Razaviyayn. A rigorous study of integrated gradients method and extensions to internal neuron attributions. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14485–14508. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/lundstrom22a.html.

Jan Macdonald, Stephan Wäldchen, Sascha Hauch, and Gitta Kutyniok. A rate-distortion framework for explaining neural network decisions, 2019.

Grégoire Montavon, Sebastian Bach, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *arXiv preprint arXiv:1512.02479*, 2015.

Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pp. 193–209, 2019.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65: 211–222, 2017. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2016.11.008. URL https://www.sciencedirect.com/science/article/pii/S0031320316303582.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, February 2018. ISSN 10512004. doi: 10.1016/j.dsp.2017.10.011. URL https://linkinghub.elsevier.com/retrieve/pii/S1051200417302385.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the Number of Linear Regions of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://arxiv.org/abs/1402.1869.

Weili Nie, Yang Zhang, and Ankit Patel. A Theoretical Explanation for Perplexing Behaviors of Backpropagation-based Visualizations. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3809–3818. PMLR, July 2018. URL https://proceedings.mlr.press/v80/nie18a.html. ISSN: 2640-3498.

Wojciech Samek, Leila Arras, Ahmed Osman, Grégoire Montavon, and Klaus-Robert Müller. Explaining the decisions of convolutional and recurrent neural networks. 2021a.

Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J. Anders, and Klaus-Robert Müller. Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. *Proceedings of the IEEE*, 109(3):247–278, March 2021b. ISSN 1558-2256. doi: 10.1109/JPROC.2021.3060483. Conference Name: Proceedings of the IEEE.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/e6acf4b0f69f6f6e60e9a815938aa1ff-Paper.pdf.

Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xWh1rYwB.

Harshay Shah, Prateek Jain, and Praneeth Netrapalli. Do input gradients highlight discriminative features? In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 2046–2059. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/0fe6a94848e5c68a54010b61b3e94b0e-Paper.pdf.

Lloyd S. Shapley. *Notes on the N-Person Game – II: The Value of an N-Person Game*. RAND Corporation, Santa Monica, CA, 1951. doi: 10.7249/RM0670.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences, 2016. URL https://arxiv.org/abs/1605.01713.

Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. Explanation by progressive exaggeration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1xFWgrFPS.

Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified BP attributions fail. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9046–9057. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sixt20a.html.

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

Erik Štrumbelj and Igor Kononenko. A general method for visualizing and explaining black-box regression models. In *International Conference on Adaptive and Natural Computing Algorithms*, pp. 21–30. Springer, 2011.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Benjamin A Toms, Elizabeth A Barnes, and Imme Ebert-Uphoff. Physically interpretable neural networks for the geosciences: Applications to earth system variability. *Journal of Advances in Modeling Earth Systems*, 12(9):e2019MS002002, 2020.

Tom Viering, Ziqi Wang, Marco Loog, and Elmar Eisemann. How to manipulate cnns to make them lie: the gradcam case. *arXiv preprint arXiv:1907.10901*, 2019.

Stephan Waeldchen, Jan Macdonald, Sascha Hauch, and Gitta Kutyniok. The computational complexity of understanding binary classifier decisions. *J. Artif. Int. Res.*, 70:351–387, may 2021. ISSN 1076-9757. doi: 10.1613/jair.1.12359. URL https://doi.org/10.1613/jair.1.12359.

Junlin Wang, Jens Tuyls, Eric Wallace, and Sameer Singh. Gradient-based analysis of NLP models is manipulable. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 247–258, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.24. URL https://aclanthology.org/2020.findings-emnlp.24.

H. Xiong, L. Huang, M. Yu, L. Liu, F. Zhu, and L. Shao. On the Number of Linear Regions of Convolutional Neural Networks. In *ICML*, 2020.

Ruihan Zhang, Prashan Madumal, Tim Miller, Krista A Ehinger, and Benjamin IP Rubinstein. Invertible concept-based explanations for cnn models with non-negative concept activation vectors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11682–11690, 2021.

A Proofs

A.1 Proof of proposition 3 and 2

We will proof Propositions 2, 3, and 4 together.

We start with the partial derivative of the relevance function at layer l:

$$\frac{\partial R^l_{[k]}(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} = \sum_{j=1}^{d_{l+1}} \frac{\partial}{\partial \boldsymbol{a}_l} \left(\left[\frac{\partial R^{l+1}_{[j]}\left(f_l(\boldsymbol{a}_l)\right)}{\partial \boldsymbol{a}_{l_{[k]}}} \right]_{\boldsymbol{a}_l = \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l)} \cdot \left\{ \boldsymbol{a}_l - \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l) \right\}_{[k]} \right) \tag{19}$$

$$= \sum_{j=1}^{d_{l+1}} \left(\frac{\partial \left(\boldsymbol{a}_{l_{[k]}} - \tilde{\boldsymbol{a}}^{(j)}_{l_{[k]}}(\boldsymbol{a}_l) \right)}{\partial \boldsymbol{a}_l} \cdot \left[\frac{\partial R^{l+1}_{[j]}\left(f_l(\boldsymbol{a}_l)\right)}{\partial \boldsymbol{a}_l} \right]_{\boldsymbol{a}_l = \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l)} \right. \tag{20}$$

$$\left. + \underbrace{\frac{\partial}{\partial \boldsymbol{a}_l} \cdot \left[\frac{\partial R^{l+1}_{[j]}\left(f_l(\boldsymbol{a}_l)\right)}{\partial \boldsymbol{a}_l} \right]_{\boldsymbol{a}_l = \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l)} \cdot \left\{ \boldsymbol{a}_l - \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l) \right\}_{[k]}}_{= 0, \text{ for ReLU networks}} \right) \tag{21}$$

In this first step, we applied the product rule. For ReLU networks, the higher-order terms are zero. For other networks (Transformer, LSTMs), the higher-order terms will not be zero. The terms which are zero for ReLU networks are exactly the terms from Proposition 4.

In the next step, we will apply the chain rule and rewrite $\partial \boldsymbol{a}_{l_{[k]}} / \partial \boldsymbol{a}_l$ as the k-standard basis e_k (a one-hot vector where the k-th dimension is 1):

$$\frac{\partial R^l_{[k]}(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} = \sum_{j=1}^{d_{l+1}} \left(e_k - \frac{\partial \tilde{\boldsymbol{a}}^{(j)}_{l_{[k]}}(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} \right) \cdot \left[\frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} \cdot \frac{\partial R^{l+1}_{[j]}\left(f_l(\boldsymbol{a}_l)\right)}{\partial f_l(\boldsymbol{a}_l)} \right]_{\boldsymbol{a}_l = \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l)}, \tag{22}$$

The next observation is that the gradients inside the $[\ldots]_{\boldsymbol{a}_l = \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l)}$ must be the same for $\boldsymbol{a}_l$ and the root point $\tilde{\boldsymbol{a}}_l^{(j)}$ as both are in the same local region of $f_l \circ R^{l+1}_{[j]}$. Therefore, we can safely drop the evaluation of the gradient at the root point ($[\ldots]_{\boldsymbol{a}_l = \tilde{\boldsymbol{a}}_l^{(j)}(\boldsymbol{a}_l)}$) and write:

$$\frac{\partial R^l_{[k]}(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} = \sum_{j=1}^{d_{l+1}} \left(\frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_{l_{[k]}}} - \frac{\partial \tilde{\boldsymbol{a}}^{(j)}_{l_{[k]}}(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} \cdot \frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} \right) \cdot \frac{\partial R^{l+1}_{[j]}\left(f_l(\boldsymbol{a}_l)\right)}{\partial f_l(\boldsymbol{a}_l)} \tag{23}$$

Substituting this result into the definition of $R^{l-1}(\boldsymbol{a}_{l-1})$ from equation 8 yields the result of Proposition 3.

To proof proposition 2, we use $\partial \tilde{\boldsymbol{a}}^{(j)}_{l_{[k]}}(\boldsymbol{a}_l) / \partial \boldsymbol{a}_l = 0$ and get:

$$\frac{\partial R^l_{[k]}(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_l} = \sum_{j=1}^{d_{l+1}} \frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_{l_{[k]}}} \cdot \frac{\partial R^{l+1}_{[j]}\left(f_l(\boldsymbol{a}_l)\right)}{\partial f_l(\boldsymbol{a}_l)} \tag{24}$$

$$= \frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_{l_{[k]}}} \cdot \sum_{j=1}^{d_{l+1}} \frac{\partial R^{l+1}_{[j]}\left(f_l(\boldsymbol{a}_l)\right)}{\partial f_l(\boldsymbol{a}_l)} \tag{25}$$

$$= \frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_{l_{[k]}}} \cdot \sum_{j=1}^{d_{l+1}} \frac{\partial f_{l+1}(\boldsymbol{a}_{l+1})}{\partial \boldsymbol{a}_{l+1_{[j]}}} \cdot \sum_{i=1}^{d_{l+2}} \frac{\partial R^{l+2}_{[i]}\left(f_{l+1}(\boldsymbol{a}_{l+1})\right)}{\partial f_{l+1}(\boldsymbol{a}_{l+1})} \tag{26}$$

$$= \frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_{l_{[k]}}} \cdot \frac{\partial f_{l+1}(\boldsymbol{a}_{l+1})}{\partial \boldsymbol{a}_{l+1}} \cdot \sum_{i=1}^{d_{l+2}} \frac{\partial R^{l+2}_{[i]}\left(f_{l+1}(\boldsymbol{a}_{l+1})\right)}{\partial f_{l+1}(\boldsymbol{a}_{l+1})} \tag{27}$$

$$= \frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_{l_{[k]}}} \cdot \frac{\partial f_{l+1}(\boldsymbol{a}_{l+1})}{\partial \boldsymbol{a}_{l+1}} \cdot \ldots \cdot \frac{\partial f_{n-1}^n(\boldsymbol{a}_{n-1})}{\partial \boldsymbol{a}_{n-1}} \cdot \sum_{i=1}^{d_{n+1}} \frac{\partial R_{[i]}^{n+1}\left(f_n^{n+1}(\boldsymbol{a}_n)\right)}{\partial f_n^{n+1}(\boldsymbol{a}_n)} \tag{28}$$

$$= \frac{\partial f_l(\boldsymbol{a}_l)}{\partial \boldsymbol{a}_{l_{[k]}}} \cdot \frac{\partial f_{l+1}(\boldsymbol{a}_{l+1})}{\partial \boldsymbol{a}_{l+1}} \cdot \ldots \cdot \frac{\partial f_{n-1}(\boldsymbol{a}_{n-1})}{\partial \boldsymbol{a}_{n-1}} \cdot \frac{\partial f_n(\boldsymbol{a}_n)}{\partial \boldsymbol{a}_n} \cdot e_\xi \tag{29}$$

$$= \nabla f_{l_{[\xi]}}(\boldsymbol{a}_l) \tag{30}$$

Substituting this into the relevance function of the input $R(\boldsymbol{x}) = R^1(\boldsymbol{a}_1)$ and using $\frac{\partial R(\boldsymbol{x})}{\partial \boldsymbol{x}}\big|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}(\boldsymbol{x})} = \frac{\partial R(\boldsymbol{x})}{\partial \boldsymbol{x}}$ (as $\tilde{\boldsymbol{x}}$ must be in the same linear region), yields:

$$R(\boldsymbol{x}) = R(\tilde{\boldsymbol{x}}(\boldsymbol{x})) + \left.\frac{\partial R(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}(\boldsymbol{x})} \odot (\boldsymbol{x} - \tilde{\boldsymbol{x}}) = R(\tilde{\boldsymbol{x}}(\boldsymbol{x})) + \nabla f_{[\xi]}(\boldsymbol{x}) \odot (\boldsymbol{x} - \tilde{\boldsymbol{x}}(\boldsymbol{x})), \tag{31}$$

which finished the proof of Proposition 2.

A.2 Admissible Region for the root points of the Relevance Function

We now proof Proposition 1 which is restated here:

Proposition 1 (C1: Recursive Taylor cannot increase the size of admissible regions) *Given a ReLU network $f : \mathbb{R}^{d_1} \to \mathbb{R}_{\geqslant 0}^{d_{n+1}}$, recursive relevance functions $R^l(\boldsymbol{a}_l)$ with $l \in \{1, \ldots, n\}$ according to definition 3, and let ξ index the explained logit. Then it holds for the admissible region $N_{R^l}(\boldsymbol{a}_l)$ for the root points $\tilde{\boldsymbol{a}}_l^{(j)}$ of the relevance function R^l that $N_{R^l}(\boldsymbol{a}_l) \subseteq N_{f_{n_\xi} \circ \ldots \circ f_l}(\boldsymbol{a}_l)$.*

Let $\tilde{\boldsymbol{a}}_1, \ldots, \tilde{\boldsymbol{a}}_n$ fix the root points. Proof by induction over the number of layers. We start with the induction base case at the final layer. There, we have $R^n(\tilde{\boldsymbol{a}}_n) = f_{[\xi]}(\tilde{\boldsymbol{a}}_n)$, which follows from the recursion base case. Clearly, $N_{R^n}(\tilde{\boldsymbol{a}}_n) \subseteq N_{f_{n_\xi}}(\tilde{\boldsymbol{a}}_n)$. Induction step: We assume $N_{R_{l+1}}(\tilde{\boldsymbol{a}}_{l+1}) \subseteq N_{f_{l+1}}(\tilde{\boldsymbol{a}}_{l+1})$. For the layer l, the root points must be valid for the function $R^{l+1}(f_l(\tilde{\boldsymbol{a}}_l))$. As we know that $N_{R_{l+1}}(\tilde{\boldsymbol{a}}_{l+1}) \subseteq N_{f_{l+1}}(\tilde{\boldsymbol{a}}_{l+1})$, it must also be the case that $N_{R^l}(\tilde{\boldsymbol{a}}_l) \subseteq N_{f_l}(\tilde{\boldsymbol{a}}_l)$.

B Details About the Relation Network for the CLEVR dataset

Our code-base builds upon a public available implementation of relation networks[4] and utilized Captum for computing $\text{LRP}_{\alpha 1 \beta 0}$ explanations Kokhlikyan et al. (2020). We also setup the CLEVR XAI dataset released on Github[5]

[4] https://github.com/rosinality/relation-networks-pytorch
[5] https://github.com/ahmedmagdiosman/clevr-xai/releases/tag/v1.0

C Pseudo-Code

C.1 Full-backward DTD

Algorithm 1 Pseudocode for the recursive application of the Taylor Theorem. The global state contains the following variables: $f_1, \ldots, f_n$ the layer functions of the network, $d_1, \ldots, d_{n+1}$ the dimension of the input to each layer, and ξ the index of the output neuron.

function GET_RELEVANCE(l: layer index, $\boldsymbol{a}_l$: the layer input)
  if $l = n + 1$ **then**
    return $\boldsymbol{a}_{l_{[\xi]}}$
  end if
  $\tilde{\boldsymbol{a}}_l \leftarrow$ FIND_ROOT_POINT$(f, l, \boldsymbol{a}_l)$
  $R^{l+1} \leftarrow$ GET_RELEVANCE$(l + 1, f_l(\tilde{\boldsymbol{a}}_l))$
  for $j \in 1 \ldots d_{l+1}$ **do**
    $\tilde{\boldsymbol{a}}$.grad $\leftarrow 0$
    $R^{l+1}_{[j]}$.backward()
    $\boldsymbol{r}_j \leftarrow \tilde{\boldsymbol{a}}$.grad $\odot (\boldsymbol{a}_l - \tilde{\boldsymbol{a}}_l)$
  end for
  return $\sum_{j=1}^{d_{l+1}} \boldsymbol{r}_j$
end function

C.2 Train-free DTD

Algorithm 2 DTD Train-Free

function FIND_ROOT_POINT(l: layer index, $\boldsymbol{a}_l$: the layer input, $R_{[j]}^{l+1}$: the relevance)
  $\boldsymbol{w}_j = W_{[j:]}$
  if z^+-rule **then**
    $\boldsymbol{v} = \boldsymbol{a}_l \odot \mathbb{1}_{\boldsymbol{w}_j \geqslant 0}$
  else if w^2-rule **then**
    $\boldsymbol{v} = \boldsymbol{w}_j$
  else if γ-rule **then**
    $\boldsymbol{v} = \boldsymbol{a}_l(1 + \gamma \mathbb{1}_{\boldsymbol{w}_j \geqslant 0})$
    ...
  end if
  $t = R_{[j]}^{l+1} / (\boldsymbol{w}_j^T \boldsymbol{v})$
  return $\boldsymbol{a}_l - t\boldsymbol{v}$ ▷ Ensures that $\boldsymbol{w}_j^T(\boldsymbol{a}_l - \tilde{\boldsymbol{a}}_l) = \boldsymbol{w}_j^T \frac{R_{[j]}}{\boldsymbol{w}_j^T \boldsymbol{v}} \boldsymbol{v} = R_{[j]}$
end function

function GET_RELEVANCE(l: layer index, $\boldsymbol{a}_l$: the layer input)
  if $l = n + 1$ **then**
    return $\boldsymbol{a}_{l_{[\xi]}}$
  end if
  $R^{l+1} \leftarrow$ GET_RELEVANCE$(l + 1, f_l(\boldsymbol{a}_l))$ ▷ relevance of input instead of root point $\tilde{\boldsymbol{a}}_l$
  for $j \in 1 \dots d_{l+1}$ **do**
    $\tilde{\boldsymbol{a}}_l^{(j)} \leftarrow$ FIND_ROOT_POINT$(f, l, \boldsymbol{a}_l, R_{[j]}^{l+1})$
    $\tilde{\boldsymbol{a}}$.grad $\leftarrow 0$
    $\boldsymbol{o} \leftarrow [W_{l_{[j]}} \boldsymbol{a}_l^{(j)} + \boldsymbol{b}_{l_{[j]}}]$
    $\boldsymbol{o}$.backward()
    $\boldsymbol{r}_j \leftarrow \tilde{\boldsymbol{a}}$.grad $\odot (\boldsymbol{a}_l - \tilde{\boldsymbol{a}}_l)$
  end for
  return $\sum_{j=1}^{d_{l+1}} \boldsymbol{r}_j$
end function

6 Do Users Benefit From Interpretable Vision? A User Study, Baseline, And Dataset

The previous three chapters were mainly concerned with the technical and theoretical aspects of explainable AI. We were able to see that some explainability methods have theoretical and empirical shortcomings.

One reason why those limitations went unnoticed for quite some time is that the method's utility was not evaluated in a user study. Only recently, a study (Alqaraawi et al. 2020) conducted a user study of saliency maps, specifically focusing on the LRP method.

They discovered that while saliency maps aided users in predicting the model's output more accurately, displaying the model's logit scores alone yielded a comparable increase in performance.

For the following publication, I collaborated with Martin Schüßler, one author of Alqaraawi et al. 2020. Extending on their work and inspired by my previous negative results regarding explainability methods, we asked what would be the fairest way to test the utility of an explainability method. In particular, we were concerned with the following:

(1) No prior knowledge of the data domain should be required: This ensures that participants do not have any preconceived notions about the data. For example, if the participants are asked about gender bias, they might be biased by their individual political views.

(2) The task should be approachable for the participants: The task should be understandable and easy to perform. Participants take less than 15 minutes to learn the task.

(3) The task should be grounded in the model: Performing well on the task should require knowledge of the model.

(4) Explanation methods should be more helpful than a simple baseline: Any technical explanation method should be compared to a simple and naive baseline such as displaying samples and the model's output.

To address the first concern, we built a synthetic dataset generator for creating images of abstract animals. These abstract animals had various adjustable attributes, including color, position, orientation, and building blocks. The synthetic datasets ensured that participants were naive about the data domain.

We decided to use a bias detection task in the study design. The users had to make a binary decision about the model's output depending on a certain property of the animal. These attributes were controlled by the dataset generator and we use them to inject arbitrary biases into the dataset. This task fulfills (2) and (3): it is easy to understand and perform, and as the biased attribute is not revealed to the participants, it requires knowledge of the model to perform well.

Previous studies did not use a bias detection task; instead, they used forward simulation (Alqaraawi et al. 2020), subjective quality evaluation (Hase et al. 2020), or similarity scoring (Zhang et al. 2021a; Ghorbani et al. 2019).

In the forward-simulation task, users are asked to predict the model's output using the explanations given. Although the task is inherently linked to the model by design, it is relatively challenging to execute and may require users to evaluate a considerable number of test samples. Other work has used a weak grounding of the task, where the user had to score the subjective quality of the explanation. However, as noted in (Hase et al. 2020), it appears that subjective user ratings of the explanation's quality do not predict explanation effectiveness in forward-simulation tests.

Other user-studies (Zhang et al. 2021a; Ghorbani et al. 2019) put more focus on the accessibility of the task. For example, the user had to sort images to the matching concept category (Ghorbani et al. 2019). However, determining the concept category does not test the knowledge about the model. Therefore, the actual goal of the explanation is not tested. Instead, the bias-detection task allows the user to reason about more high-level properties than on individual samples.

Another technical aspect of our study is that we measured the biased properties of the model using interventions. This is important as the biases of a dataset are not necessarily adopted by the model. These properties can be used to create biases in the dataset which are challenging to detect visually.

To address the last concern, we developed a baseline method that arranges images in a grid based on their logit scores. Each column within the grid

corresponds to a specific range of logit scores. This image grid allows users to scan many images quickly for similarities and differences. For instance, the previous work by (Ribeiro et al. 2016) lacked such a baseline method, which drew criticism from others, such as (Hase et al. 2020).

For my thesis, this publication is relevant because while my other publications focus on the technical aspects of explainability methods, user studies of explainability are the only way to ensure that the methods are actually useful for users.

DO USERS BENEFIT FROM INTERPRETABLE VISION? A USER STUDY, BASELINE, AND DATASET

Leon Sixt[*1], **Martin Schuessler**[*23], **Oana-Iuliana Popescu**[1], **Philipp Weiß**[3], **Tim Landgraf**[1]
Freie Universität Berlin[1] Weizenbaum Institut Berlin[2] TU Berlin[3]
leon.sixt@fu-berlin.de, martin.schuessler@tu-berlin.de
[*] Equal Contribution

ABSTRACT

A variety of methods exist to explain image classification models. However, it remains unclear whether they provide any benefit to users over simply comparing various inputs and the model's respective predictions. We conducted a user study (N=240) to test how such a baseline explanation technique performs against concept-based and counterfactual explanations. To this end, we contribute a synthetic dataset generator capable of biasing individual attributes and quantifying their relevance to the model. In a study, we assess if participants can identify the relevant set of attributes compared to the ground-truth. Our results show that the baseline outperformed concept-based explanations. Counterfactual explanations from an invertible neural network performed similarly as the baseline. Still, they allowed users to identify some attributes more accurately. Our results highlight the importance of measuring how well users can reason about biases of a model, rather than solely relying on technical evaluations or proxy tasks. We open-source our study and dataset so it can serve as a blue-print for future studies.

1 INTRODUCTION

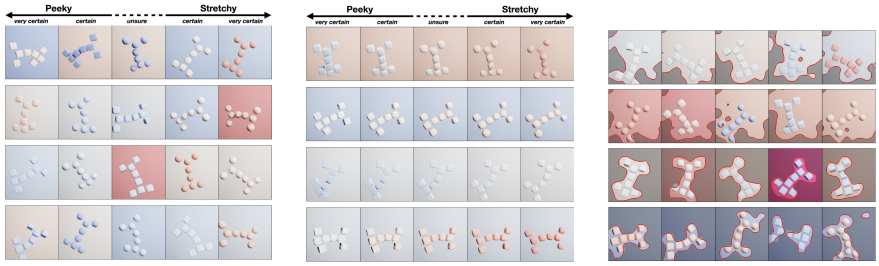
Deep neural networks have been widely adopted in many domains. Yet, for some applications, their use may be limited by how little we understand which features are relevant. Whether engineer or user, insurance company, or regulatory body; all require reliable information about what the model has learned or why the model provides a certain output. Numerous methods have been proposed to explain deep neural networks (Gilpin et al., 2018; Molnar et al., 2020).

Ultimately, to evaluate whether such explanations are helpful, we need user studies (Doshi-Velez & Kim, 2017; Wortman Vaughan & Wallach, 2020). In fact, some studies provided evidence that interpretable ML techniques may be helpful to find biases or spurious correlations (Ribeiro et al., 2016b; Adebayo et al., 2020a). However, a substantial body of work shows that they may not be as helpful as claimed (Kaur et al., 2020; Alqaraawi et al., 2020; Chu et al., 2020; Shen & Huang, 2020). Consequently, it seems that in real-world applications, biases are often found by simply inspecting the model's predictions rather than applying interpretable ML. A recent example is the Twitter image cropping algorithm: it was the users who discovered that it favored white people over people of color (Yee et al., 2021). In this work, we ask: do modern interpretability methods enable users to discover biases better than by simply inspecting input/output pairs?

To investigate this question empirically, we first propose TWO4TWO: a synthetic dataset depicting two abstract animals. Its data-generating factors can be correlated with the binary target class, thereby creating arbitrarily strong biases. We designed a baseline explanation technique for bias discovery using only the model's output: input images are arranged in a grid grouped by the model's logit predictions. This design allows a user to inspect all attributes that potentially predict the target class.

In an initial user study (N=50), we validated that participants were struggling to find both biases contained in our dataset using this technique. Hence, more elaborate methods can improve over the baseline on this dataset. In the main study (N=240), we compared the baseline against two state-of-the-art explanations: automatically-discovered concepts and counterfactual interpolations generated with an invertible neural network.

Published in International Conference on Learning Representations (ICLR), 2022



(a) Baseline (b) Invertible Neural Networks (c) Concepts (Zhang et al., 2021)

Figure 1: We tested whether users can identify the class-relevant features of images showing two types of animals. We biased attributes like the animal's color to be predictive of the class and investigated whether explanation techniques enabled users to discover these biases. We tested a simple baseline **(a)** which shows random samples grouped by the model's output logit, counterfactual samples generated by an invertible neural network **(b)**, and automatically discovered concepts **(c)**. A participant viewed only one of the above conditions.

We found that none of these explanations outperformed the baseline, even though some features were identified more accurately with counterfactuals. The textual justifications of participants revealed several usability issues in all methods. This highlights the necessity to validate any claims about the benefits of explanation techniques in user studies.

This work represents substantial empirical novelty and significance in the field of interpretable ML:

- The TWO4TWO dataset generator provides control over features and biases. It is designed specifically for human subject studies and to challenge existing interpretability approaches,
- Methods to quantify ground-truth feature importance when the data-generating process is known,
- A study design that provides a unified approach to evaluating interpretable vision methods on the task of bias discovery. It is suitable for lay-users and includes several measures to ensure high-quality crowd-sourced responses,
- A strong and simple baseline explanation technique using only the model output, which we propose as a benchmark for future studies,
- We open-source our dataset, explanation techniques, model, study design, including instructions and videos to support replicating our results as well as adapting our design to other explanation techniques.

2 RELATED WORK

Interpretable ML for Vision Different explanation approaches have been proposed: saliency maps (Bach et al., 2015; Ancona et al., 2018; Sundararajan et al., 2017), example-based explanations (Cai et al., 2019), counterfactual examples (Singla et al., 2020), activation-concept approaches (Kim et al., 2018), or models with built-in interpretability (Chen et al., 2019; Brendel & Bethge, 2018). For a detailed review about the field, we refer to (Gilpin et al., 2018; Molnar et al., 2020).

Our work focuses on counterfactual explanations and automatically-discovered concepts. Counterfactual explanations are samples that change the model output, e.g., flip the output class (Wachter et al., 2018). We generated counterfactuals with invertible neural networks (INNs) (Jacobsen et al., 2018; Kingma & Dhariwal, 2018). This approach has recently gained momentum (Hvilshøj et al., 2021; Dombrowski et al., 2021; Mackowiak et al., 2020). Previous works have also used GANs and VAEs for counterfactual generation (Goyal et al., 2019; Mertes et al., 2020; Sauer & Geiger, 2021; Singla et al., 2020; Liu et al., 2019; Baumgartner et al., 2018; Chang et al., 2019). The main advantage of using INNs for counterfactuals is that the generative function is perfectly aligned with the forward function, as an analytic inverse exists.

Concepts represent abstract properties, which can be used to explain a model. For example, the classification of an image as "zebra" could be explained by a pronounced similarity to the "stripe" concept. This similarity is determined by the dot product of the network's internal activations with a concept vector. TCAV (Kim et al., 2018) required manually defined concepts. Recent works proposed to discover concepts automatically (Ghorbani et al., 2019; Zhang et al., 2021).

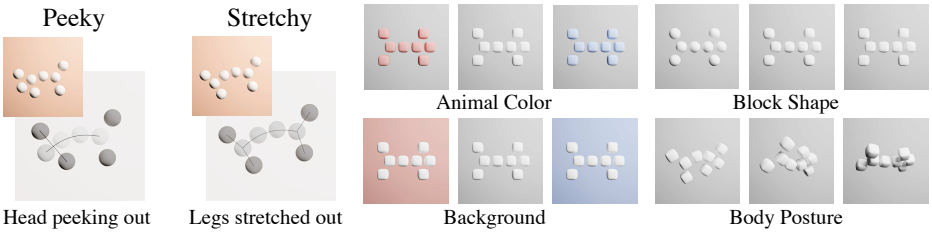


Figure 2: The left panel depicts the main difference between *Peeky* and *Stretchy*: the legs' position. While *Peeky* shows one pair of legs moved inwards, *Stretchy's* legs are moved outwards. TWO4TWO offers different attributes: animal color, background color, the shape of the blocks and the animal's body posture. All of which can be controlled and biased separately.

User Studies for Interpretability Previous works with the task of bias discovery have mainly evaluated saliency maps and used datasets with a single, simple bias, e.g. background Adebayo et al. (2020a); Ribeiro et al. (2016a) or image watermarks Kim et al. (2018). User studies for concept-based methods tested only the accessibility of the explanations by asking users to assign images to a concept (Zhang et al., 2021; Ghorbani et al., 2019). Counterfactual explanations have been evaluated by Mertes et al. (2020) on a forward-prediction task. We thus believe that we are the first to extensively test counterfactual-based and concept-based explanations on bias discovery using a challenging dataset. Recently, a study on exemplary-based explanations focused on understanding internal activations of a neural network (Borowski et al., 2020). It showed that for this task, examples could be more beneficial than complex feature visualizations (Olah et al., 2017). Similarly, there is evidence that participants often rely on model predictions rather than on explanations (Alqaraawi et al., 2020; Adebayo et al., 2020a).

Synthetic Datasets for Interpretable Vision Datasets with known ground-truth biases have been proposed before. BAM is an artificial dataset (Yang & Kim, 2019) where spurious background correlations are introduced by pasting segmented objects on different textures, e.g. dogs on bamboo forests. However, the resulting images are unsuitable for user studies as they look artificial and make it easy for participants to suspect that the background is important. Additionally, it would be difficult to introduce more than one bias. A limitation that also the synthetic dataset in (Chen et al., 2018) shares. The synthetic dataset created by Arras et al. (2021) created a dataset to technically evaluate saliency methods on a visual question answering task technically. TWO4TWO is the first dataset designed explicitly for human subject evaluations. To the best of our knowledge, we provide the first unified approach to evaluate interpretable vision on a bias-discovery task.

3 TWO4TWO: DATASETS WITH KNOWN FEATURE IMPORTANCE

Dataset Description Datasets generated with TWO4TWO consist of two abstract *animal* classes, called *Peeky* and *Stretchy*. Both consist of eight blocks: four for the spine and four for the legs. For both animals, one pair of legs is always at an extended position. The other pair moves parallel to the spine inward and outward. The attribute *legs' position*, a scalar in [0,1], controls the position. At a value of 0.5, the pair of legs are at the same vertical position as the last block of the spine. Peekies have a leg position ≤ 0.52 which means legs are moved mostly inwards to the body center. In the same fashion, Stretchies are extended outwards, legs' position ≥ 0.48. We added some ambiguity to ensure a model has an incentive to use possible biases. Therefore, Peekies and Stretchies are equally likely for a legs' position between 0.48 and 0.52. It is also difficult for humans to tell if the legs are outward or inwards in this range. Besides the legs' position, the dataset has the following parameters: *body posture* (bending and three rotation angles), *position*, *animal color* (from red to blue), *blocks' shape* (from cubes to spheres), and *background color* (from red to blue). Each can be changed arbitrarily and continuously (See Appendix Table 5).

When designing the dataset, we wanted to ensure that (1) participants can become experts within a few minutes of training, (2) it allows for the creation of multiple biases that are difficult to find, and (3) that it provides a challenge for existing interpretability methods. Goal (1) is met as participants can be instructed using only a few examples (see the tutorial video in Appendix C). The high number of controllable attributes achieve Goal (2). We biased the attributes such that they do not stand out, which we validated in the first user study. Goal (3) is met by spatially overlapping attributes and long-

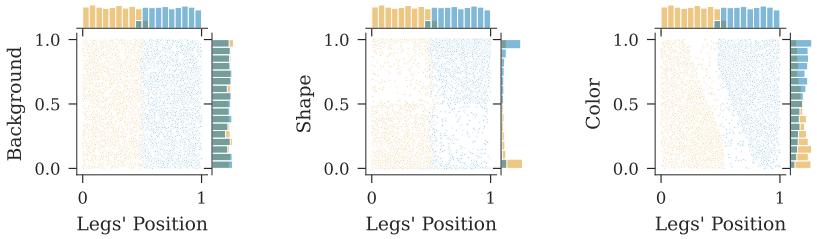


Figure 3: The joint distributions of legs' position and the attributes background (left), shape (middle), and color (right). Datapoints are yellow for Peekies and blue for Stretchies. The background is not biased. The shape is biased for legs' position lower than (0.45) or greater (0.55), but is uniform in the center. The color contains additional predictive information about the target class, as it allow to discriminate between Peeky and Stretchy where the legs' position overlaps. However, for more extreme arms' positions the color is uniform and not biased.

range image dependencies. Spatially overlapping attributes, like color and shape, directly challenge saliency map explanations. Long-range image dependencies, like the legs' positions relative to the spine, can not be explained when analyzing patches separately as done in (Chen et al., 2019; Brendel & Bethge, 2018). Both properties are common in real-world datasets: For example, race and gender in facial datasets are encoded by spatially overlapping features. Long-range image dependencies are particularly relevant for pose estimation and visual reasoning (Johnson et al., 2017).

Introducing Biases For our studies' dataset, we sampled the block's shape in a non-predictive biased fashion. This means that for legs' positions that clearly showed a Peeky [0, 0.45] most blocks were rather cubic, while for legs' positions that clearly showed a Stretchy [0.55, 1] most blocks were rather round. However, for the legs' positions between [0.45, 0.55] the blocks shape was uniformly distributed. In particular, in the even narrower interval [0.48, 0.52] where a classifier can only be as good as random guessing, the block's shape does not provide any additional information about the target class. In Figure 3, we show the joint distribution of the block's shape and legs' position.

We sampled the animals' color to be predictive for the target class. At the small interval where the legs overlap [0.48; 0.52], we distributed the animal color to provide additional class information. Stretchies were more likely to be red, and Peekies were more likely to be blue. Outside of this centered interval, the color gradually became uniformly distributed (see Figure 3). Hence, color was more equally distributed than the shape, making the color bias harder to detect visually. The remaining attributes, background color and body posture, were sampled independently of the class, and we expected our model to ignore them.

Measuring Ground-Truth Feature Importance Even if a dataset contains biases, it is unclear how relevant they will be to a neural network after training. Feature importance also depends on the network architecture, the optimization process, and even the weight initialization. As TWO4TWO allows us to change any parameter in isolation, we can directly compare the model prediction between two images that differ in only one parameter. For these two images, we measured both the median absolute logit change and also for how many samples the predicted class was flipped. Both measures quantify how influential each parameter is (see Table 1). As expected, the legs' position had a strong influence on the prediction. The model relied more on animal color than on the blocks' shape, which is expected as the color contains additional information about the class. Surprisingly, the prediction flip for unrelated attributes such as background was only slightly lower than for blocks' shape.

To analyze this further, we calculated a linear fit for each parameter change to the logit change. We reported the coefficient of determination R^2, which indicates how much of the variance in the prediction can be explained linearly by the analyzed property. While the unrelated properties sometimes flip a prediction, the direction of that flip is random ($R^2 \approx 0$). In contrast, the biased parameters influence predictions in a directed fashion, with animal color (R^2=0.751) being clearly more directed than blocks' shape (R^2=0.307).

4 MODEL AND EVALUATED METHODS

As discussed in section 3, TWO4TWO was designed to challenge existing interpretability methods, e.g., saliency map explanations and patch-based models. We selected two methods that might provide

Table 1: Importance of the data generating factors to the model's prediction. *Prediction Flip* quantifies how often the model's prediction changes the sign when changing the attribute. The *Mean Logit Change* reports the median of the absolute change in logit values. The R^2 score is calculated on an ordinary least squares from the changes of each factor to the changes in the model's logit. For more attributes, see Appendix Table 3.

Factor	Prediction Flip [%]	Median Logit Change	R^2
Legs' Position	41.680	2.493	0.933
Color	7.080	0.886	0.751
Shape	3.920	0.577	0.307
Background	2.640	0.523	0.006
Rotation Yaw	3.480	0.669	0.001
Bending	3.640	0.605	0.000

the user with the necessary information: counterfactuals generated with an invertible neural network (INN) and concept-based explanations (Zhang et al., 2021).

INN Counterfactuals We trained an INN using both a supervised and an unsupervised objective (Dinh et al., 2016; 2015). To predict the target class, the model first applies the forward function φ to map a data point $\boldsymbol{x}$ to a feature vector $\boldsymbol{z} = \varphi(\boldsymbol{x})$. Then, a linear classifier takes those features $\boldsymbol{z}$ and predicts the logit score $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{z} + b$. Any input can be reconstructed from the feature vector by applying the inverse function $\boldsymbol{x} = \varphi^{-1}(\boldsymbol{z})$. The model has a test accuracy of 96.7%. Further details can be found in Appendix A.2. The baseline and concept techniques are also applied to this model.

To create a counterfactual example $\tilde{\boldsymbol{x}}$ for a data point $\boldsymbol{x}$, we can exploit the linearity of the classifier. Moving along the weight vector $\boldsymbol{w}$, i.e., adding $\boldsymbol{w}$ to the features $\boldsymbol{z}$, changes the model's prediction. By controlling the step size with a scalar α, we can directly quantify the change in the logit value $\Delta y = \alpha\boldsymbol{w}^T\boldsymbol{w}$. The modified feature vector $\boldsymbol{z} + \alpha\boldsymbol{w}$ can be inverted back to the input domain, resulting in a counterfactual $\tilde{\boldsymbol{x}} = \varphi^{-1}(\boldsymbol{z} + \alpha\boldsymbol{w})$ which visualizes the changes introduced by a step $\alpha\boldsymbol{w}$ in $\boldsymbol{z}$-space. The INN's explanations are visualized in a grid where each row shows a single counterfactual interpolation (see Figure 1b).

Automatically-discovered Concepts We adapted the NMF approach of Zhang et al. (2021) to our specific network architecture. Because the network's internal representations also contain negative values, we used matrix factorization instead of NMF. We generated the concepts using layer 342 (from a total of 641 layers). The layer has a feature map resolution of 8x8. This choice represents a trade-off between enough spatial resolution and high-level information. We ran the matrix factorization with 10 components and selected the five components that correlated most with the logit score (r is in the range [0.21, 0.34]).

Our presentation of concept-based explanations was very similar to (Zhang et al., 2021): we visualized concepts with five exemplary images per row and highlighted regions corresponding to a concept. Since our classifier is binary, a negative contribution for Stretchy actually means a positive contribution for Peeky. Hence, we could have characterized a concept as *more Peeky* and *more Stretchy*, to make the design similar to the other two explanation techniques. However, as the concepts did not strongly correlate with the model's output, presenting them as class-related could confuse participants: a *more Peeky* column would have contained some images showing Stretchies and vice versa. Thus, we presented them separately in two consecutive rows (See Figure 1c). Presenting concepts in this fashion gives them a fair chance in the study because participants rated the relevance of each attribute for the model rather than for each class separately.

5 Human Subject Study

We share the view of Doshi-Velez & Kim (2017) and Wortman Vaughan & Wallach (2020) that user-testing of explanation techniques is a crucial but challenging endeavor. As our second main contribution, we propose and conduct a user study based on the Two4Two dataset which can act as a blue-print for future investigations. Our design has been iterated in over ten pilot studies and proposes solutions to common problems that arise when evaluating explanation techniques on crowd-sourcing platforms with lay participants.

5.1 DESIGN CONSIDERATIONS

Data without Prior Domain Knowlege We specifically designed the Two4Two dataset to avoid overburdening participants, as might be the case with other types of data. Within a few minutes, participants can easily become domain experts. Since the data is unknown to them prior to the study, we avoid introducing any prior domain knowledge as a confounding factor, which can be an issue (Alqaraawi et al., 2020).

Manageable but not Oversimplified Tasks We propose the task of *bias-discovery*: participants had to rate features as either *relevant* or *irrelevant* to a model. The task directly reflects users' perception of feature importance. Furthermore, bias-discovery has the advantage of being suitable for lay participants. At the same time, it is also grounded in the model's behavior. This is an advantage over tasks used in several previous studies, which only evaluated whether explanations were *accessible* to users, e.g. by identifying the target property *smiling* using image interpolations (Singla et al., 2020) or assigning images to a concept class (Zhang et al., 2021; Ghorbani et al., 2019). However, these tasks are an oversimplification and cannot measure any insights the users gained about the model. In contrast, Alqaraawi et al. (2020) employed the task of *forward prediction* of a neural network. This requires substantial model understanding and is very challenging, as reflected by the participants' low accuracy. Assessing trust in a *human-in-the-loop task*, despite its realistic setting, has the disadvantage that trust is influenced by many factors which are difficult to control for (Lee & See, 2004; Springer et al., 2017). Another approach is to asks participants to assess whether a model is fit for deployment (Ribeiro et al., 2016b; Adebayo et al., 2020b). However, in our own pilots studies, users deemed a model fit for deployment even if they knew it was biased.

Baseline Explanation Technique To quantify whether an explanation is beneficial for users, it must be compared to an alternative explanation. In this work, we argue that a very simple and reasonable alternative for users is to inspect the model's logits assigned to a set of input images. We designed such a baseline explanation as shown in Figure 1a. After several design iterations, we settled for a visually dense image grid with 5 columns sorted by the logit score, each column covering 20% of the logit values. The columns were labeled very certain for Peeky/Stretchy, certain for Peeky/Stretchy, and as unsure. Pilot studies showed that participants' attention is limited. We thus decided to display a total of 50 images, i.e. an image grid of 10 rows. The number of images was held constant between explanation techniques to ensure the same amount of visual information and a fair comparison. In this work, we focused on binary classifications. For a multi-class setting, one could adapt the baseline by contrasting one class verses another class.

High Response Quality We took extensive measures to ensure participants understood their task and the explanation techniques. Participants were required to watch three professionally-spoken tutorial videos, each under four minutes long. The videos explained, on a high level, the Two4Two dataset, machine learning and how to use an assigned explanation technique to discover relevant features. To avoid influencing participants, we prototyped idealized explanations using images from TWO4TWO. The explanations showed different biases than those in the study. Each video was followed by a written summary and set of multiple choice comprehension questions After failing such a test once, participants could study the video and summary again. When failing a test for a second time, participants were excluded from the study. We also excluded participants if their written answers reflected a serious misunderstanding of the task, indicated by very short answers copied for all attributes or reasoning that is very different from the tutorial. We recruited participants from Prolific who are fluent in English, hold an academic degree and have an approval rate of $\geq 90\%$. To ensure they are also motivated, we compensated them with an average hourly pay of £11.45 which included a bonus of £0.40 per correct answer.

5.2 EXPERIMENTAL DESIGN

We conducted two online user studies. Before starting the data collection, we formulated our hypotheses, chose appropriate statistical tests, and pre-registered our studies (see Appendix D). This way, we follow the gold-standard of defining the statistical analysis before the data collection, thus ensuring that our statistical results are reliable (Cockburn et al., 2018). The first study (N=50) analyzed whether the task was challenging enough that other methods could potentially improve over the baseline. We tested if at least one bias in our model (either the animal's color or the blocks' shape) was difficult to find using the baseline technique. Consequently, we used a within-subject design.

Table 2: The mean accuracy for each attribute by condition. $N_{\text{collected}}$ provide the number of participants collected and N_{filtered} the number of remaining participants after the filtering. Stars mark statistical significance.

Condition	$N_{\text{collected}}$	N_{filtered}	Overall	Legs	Color	Backgr.	Shape	Posture
Study 1 (Baseline)	50	43	73.4	86.0	48.8	86.0	74.4	72.1
Study 2	240	192	67.0	78.2	58.9	66.8	73.1	59.1
INN	80	62	**84.5**	***100.0**	*82.3	*79.0	90.3	**71.0**
Baseline	80	71	80.8	85.9	59.2	**95.8**	**93.0**	70.4
Concepts	80	59	32.2	45.8	32.2	18.6	32.2	32.2

In the second study (N=240), we evaluated the two explanation techniques described in Section 4 against the baseline using a between-subjects design. Participants were randomly, but equally assigned to one of the explanation techniques. We specified two directed hypotheses. We expected participants in the INN condition to perform better than those in baseline, because the baseline does not clearly highlight relevant features, whereas interpolations highlight features in isolation. We expected participants viewing concepts to perform worse than those in the baseline, due to their inability to highlight spatially overlapping features.

For both studies, participants completed a tutorial phase first. Using their assigned explanations, they then assessed the relevance of five attributes: legs' position relative to the spine, animal color, background, rotation or bending, and blocks' shape. The questions were formulated as: *"How relevant is <attribute> for the system?"*, and participants had to choose between *irrelevant* or *relevant*. The percentage of correct answers (accuracy) served as our primary metric. Participants also had to write a short, fully-sentenced justification for their answers. For links to the study, see Appendix C.

5.3 Results

Data Exclusions As stated in the preregistration, we automatically excluded all participants who withdrew their consent, failed one of the comprehension questions twice, skipped a video, or exceeded Prolific's time limit for completion. If a participant was excluded, a new participant's place was made available until the pre-registered number of completed responses was reached. We excluded 63 study respondents for the first study, and 145 for the second study in this fashion. We ensured that all participants were naive about the dataset. Once they participated in a study, they were blacklisted for future studies.

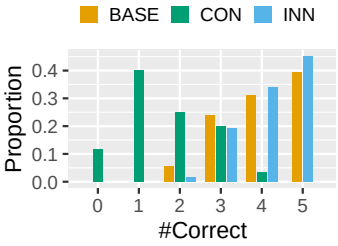


Figure 4: The proportion of correct answers for baseline (BASE), concepts (CON), and INN.

For completed studies, two annotators independently marked the participants' written answers and excluded those with copy and paste answers or indications of grave misunderstandings of the instructions. Participants were labeled as: *include*, *unsure*, or *exclude*. Both anotators had an agreement of $\kappa = 0.545$ for the first study and $\kappa = 0.643$ for the second (measured *include* vs. *unsure* and *exclude*). Disagreements were solved by discussion. In total, we excluded 7 participants from the first study (14%) and 48 participants from the second study (20%).

First Study For the accepted 43 participants, we used two-sided exact McNemar tests on their answers about the relevance of the *legs position* compared with *animal color* (first test) and *background* (second test). Participants found the color bias less often than the legs' positions (P <0.0001). The success rate for the color attribute was 49% vs. 86% for legs. The shape bias was not significantly harder to find than the legs' positions and was identified correctly with 74% accuracy (P=0.3036). Hence, we confirmed our hypothesis and concluded that other methods still have room for improvement over the baseline.

Second Study In the second study, we evaluated 192 valid participant responses (62 INN, 71 BASE, 59 CON). We expected data to be different from the normal distribution, and a Shapiro-Wilk test for all conditions confirmed this ($P < 0.001$). We depict the number of correct answers per condition in Figure 4. A Kruskal-Wallis test showed a significant differences in accuracy scores between conditions ($P < 0.001$). For focused comparisons, we used two Wilcoxon-rank-sum

tests with Bonferroni correction to correct for multiple comparisons. The accuracy scores differed significantly between the baseline and concept conditions ($P < 0.001$, r=0.778). The performance of participants using concepts was rather poor, with only 31.7% accuracy, considering that random answers would yield a score of 50%. For concepts, not a single attribute surpassed the 50% barrier. We found no significant difference when comparing the baseline and counterfactuals (P=0.441, r=0.091). Their mean accuracies are close, with 80.8% for baseline and 84.5% for counterfactuals. INN counterfactuals helped users to discover the main attribute, legs' position, (P <0.001) and color bias (P=0.033) more reliably.[1] However, counterfactuals performed significantly worse for the background attribute (P=0.033), while for blocks' shape and position we found no significant difference (for both, P=1).

Qualitative Results To understand how participants integrated the explanation techniques into their reasoning, we analyzed the textual answers of each feature qualitatively. Two annotators first applied open coding to the answers. They performed another pass of closed coding after agreeing on a subset of the relevant codes, on which the following analysis is based. Overall, the participants perceived the task as challenging, as they expressed being unsure about their answers (N=71).

We designed our image grid to show both possible classes and provide information about the model's certainty. We found that many participants integrated this additional source of information into their reasoning. This was especially prevalent in the baseline condition (N=51). Participants particularly focused on the columns 'very certain Peeky' and 'very certain Stretchy', as well as on the column 'unsure'. While this may have helped confirm or reject their own hypotheses, it sometimes led to confusion; for example, when an image that exhibited a pronounced leg position, and therefore could easily be identified as Peeky or Stretchy, was classified by the model as 'unsure' (N=14).

Across conditions, we also observed that participants expect that all images needed to support a hypothesis. *"The animals are in different colors, there are blue stretchy and also blue peeky animals, If the color was relevant peeky/stretchy would be in one color etc"* (P73, BASE). Across conditions, most participants that applied such deterministic reasoning failed to find the color bias. In contrast, other participants applied more probabilistic reasoning, which helped them deal with such contradictions: *"Peeky is more likely to be blue in colour, whereas Stretchy is more likely to be pink. This is not always true (e.g. the shapes can be white in colour at either ends of the spectrum) but it might be somewhat relevant to help the system decide"* (P197, INN).

Another observed strategy of participants was to reference how often they saw evidence for the relevance of a feature (N=35), which was very prevalent in the concepts condition (N=20). Especially concepts were rather difficult for participants to interpret. A common issue was that they expected a relevant feature to be highlighted completely and consistently (N=38). Several instances show that participants struggled to interpret how a highlighted region can explain the relevance of a feature, *"If this [the legs position] were relevant I would have expected the system to highlight only the portion of the image that contains the legs and spine. (e.g. only the legs and one block of the spine at each end). Instead, every image had minimally the entire animal highlighted"* (P82, CON). Furthermore, spatially overlaping features were another cause of confusion: *"there are rows in which the animal is highlighted but not the background so it could be because of color, shape or rotation"* (P157, CON)

Participants erred more often for the background in the INN condition than for the baseline. We conducted an analysis to investigate this issue. We found that 29 participants stated that they perceived no changes in the background of the counterfactuals and hence considered this feature irrelevant. Another 21 participants noted that they saw such a change, which let 12 of them to believe its a relevant feature. *"The background color changes in every case, it's also a little subtle but it does"* (P205). Another 9 participants decided that the changes were too subtle to be relevant. *"The background colour does not change an awful lot along each row, maybe in a couple of rows it changes slightly but I do not feel the change is significant enough that this is a relevant factor in the machine decision"* (P184).

Do Counterfactuals Highlight Irrelevant Features? Indeed, subtle perceptual changes in background color are present (Figure 1b). To quantify these changes, we decided to use an objective observer: a convolutional neural network. We trained a MobileNetV2 (Sandler et al., 2018) to predict

[1]The statistical analysis of the attributes for INN vs. baseline was not pre-registered. The reported p-values for the attributes were corrected for eight tests (including the pre-registered tests) using the Holm–Bonferroni method.

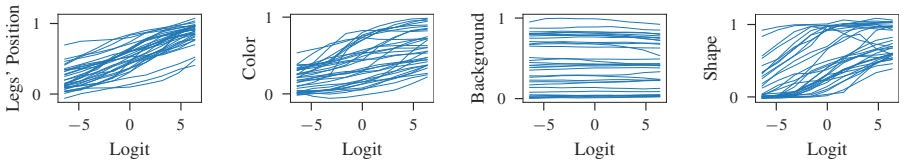


Figure 5: Attribute changes along counterfactual interpolations as measured by an observer convnet. Each line corresponds to a single sample whose logit score is modified through linear interpolations in the classifier space.

the parameter values of individual attributes of an image (e.g., background color, object color, etc.) using a completely unbiased version of TWO4TWO. After training, the model could predict the parameter values almost exactly (MSE < 0.0022, see Table 7). We then used this model to evaluate the parameter values of counterfactual INN interpolations, each spanning 99% of the logit distribution. We visualize the predictions of MobileNetV2 in Figure 5. All predictive properties (legs' position, body color, blocks' shape) are changed by the counterfactuals consistently. For the background, the changes are subtle but present. We also quantified the change in parameters using the difference between the maximum and minimum predicted value per individual interpolation (See Table 6). This supports the finding that relevant attributes change the most – legs' position: 0.662; shapes: 0.624; color: 0.440. The background changes less with 0.045, which seems enough to give some participants a false impression about its relevance.

6 CONCLUSION

Contributions We contribute a dataset with full control over the biases it contains and methods to verify the feature importances of a given model. The dataset was specifically designed for user studies and we contribute a carefully crafted study design as a template for future empirical evaluations of interpretable vision. Our design includes a simple, yet powerful baseline technique that relies on the model's outputs only. While interpretability methods had room to improve over the baseline, we showed that two state-of-the-art methods did not perform significantly better. Our results emphasize that any explanation technique needs to be evaluated in extensive user studies.

Limitations Due to budget constraints, we limited the number of factors in our experimental design (external vs. internal validity trade-off). Our study introduced a predictive bias for the animal's color and a non-predictive bias for the blocks' shape. It remains unclear how our results may have changed for a different dataset configuration: certain biases could exhibit different visual saliency. It remains also left for future work to determine which visual interface design is optimal for a given method. Furthermore, our study design restricted participants to make binary choices and provide textual justifications – limiting our understanding of the participants issues.

Take-Aways We were surprised by the performance of the two tested techniques. Users had problems interpreting the automatically-discovered concepts and could not identify the relevant attributes. As we expected, explaining spatially overlapping features by highlighting important regions limited the concepts' expressiveness. On the other hand, INN counterfactuals also did not perform significantly better than the baseline. Still, counterfactuals were more helpful to discover the strongest bias in the model. However, some participants rated the relevance of the background incorrectly, as slight changes in the interpolations were still salient enough. It is therefore important for future work to develop counterfactuals that alter only relevant attributes.

We presented a user study on a synthetic dataset. We believe that the results also have implications for natural image data. When we created Two4Two, our objective was to translate challenges faced on "real" computer vision data (like spatially overlapping features) into an abstract domain. Although some properties of photorealistic datasets are lost in this abstraction, a method performing poorly on TWO4TWO would likely not perform well on a natural dataset with spatially overlapping features.

Outlook The user study was a reality check for two interpretability methods. Such studies can guide technical innovations by identifying areas where users still struggle with current explanation methods. They are laborious and expensive, but at the same time, they are crucial for future interpretability research. Future work could focus on creating a more realistic, controllable dataset, e.g. using augmented reality (Alhaija et al., 2018). By open-sourcing our videos, study, and code we encourage the community to take on the challenge to beat the simple baseline.

7 ACKNOWLEDGMENTS

We thank Jonas Köhler and Benjamin Wild for their feedback on the manuscript. We also thank our reviewers for their time. LS was supported by the Elsa-von-Neumann Scholarship of the state of Berlin. MS and PW were funded by the German Federal Ministry of Education and Research (BMBF) - NR 16DII113. In addition, the work was partially funded by the the German Federal Ministry of Education and Research (BMBF), under grant number 16DHB4018. We thank the Center for Information Services and High Performance Computing (ZIH) at Dresden University of Technology and the HPC Service of ZEDAT, Freie Universität Berlin, for generous allocations of computation time (Bennett et al., 2020).

8 ETHICS STATEMENT

In our human subject evaluation, it was important for us to pay crowd workers £11.45 per hour which is above UK minimum wage of £8.91. The crowd workers consented to the use of their answers in the study. No personal-data was collected. Our organization does not require an approval of online user studies through an ethics review board.

9 REPRODUCIBILITY STATEMENT

Our work brings some additional challenges to reproducibility beyond compared to other machine learning research. The results of the human subject study depend on the dataset, model and even the presentations in the video tutorials. We decided to tackle this challenge by first open-sourcing our dataset, model, videos, code, and even the study itself. For now, we share the links to the study and videos in the Appendix. We will share an machine-readable export of the study, the source-code, and the model with our reviewers via OpenReview once the discussion period start and will make them public at a later point in time.

In total, we spent around 5000 GBP including prestudies, software licenses, and our two studies in the paper. We estimate the costs of reproducing the studies in our work at around 2500 GBP excluding software licenses. For a study comparing only one condition against the baseline, we would estimate the costs to be around 1400 GBP.

REFERENCES

J. Adebayo, Michael Muelly, I. Liccardi, and Been Kim. Debugging tests for model explanations. *ArXiv*, abs/2011.05429, 2020a.

Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. Debugging tests for model explanations, 2020b.

Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars M. Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126:961–972, 2018.

Ahmed Alqaraawi, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze. Evaluating saliency map explanations for convolutional neural networks: A user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, pp. 263–274, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.1145/3377325.3377519.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sy21R9JAW.

Leila Arras, Ahmed Osman, and Wojciech Samek. Clevr-xai: A benchmark dataset for the ground truth evaluation of neural network explanations. *Information Fusion*, 2021. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2021.11.008. URL https://www.sciencedirect.com/science/article/pii/S1566253521002335.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7), 2015. doi: 10.1371/journal.pone.0130140.

Christian F Baumgartner, Lisa M Koch, Kerem Can Tezcan, Jia Xi Ang, and Ender Konukoglu. Visual feature attribution using wasserstein gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8309–8319, 2018.

Loris Bennett, Bernd Melchers, and Boris Proppe. Curta: A general-purpose high-performance computer at zedat, freie universität berlin. http://dx.doi.org/10.17169/refubium-26754, 2020.

Judy Borowski, Roland S. Zimmermann, Judith Schepers, Robert Geirhos, Thomas S. A. Wallis, Matthias Bethge, and Wieland Brendel. Exemplary natural images explain cnn activations better than feature visualizations, 2020.

Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. In *International Conference on Learning Representations*, 2018.

Carrie J. Cai, Jonas Jongejan, and Jess Holbrook. The effects of example-based explanations in a machine learning interface. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, pp. 258–262, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362726. doi: 10.1145/3301275.3302289. URL https://doi.org/10.1145/3301275.3302289.

Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and D. Duvenaud. Explaining image classifiers by counterfactual generation. In *ICLR*, 2019.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. In *Advances in neural information processing systems*, pp. 8930–8941, 2019.

Yuxin Chen, Oisin Mac Aodha, Shihan Su, Pietro Perona, and Yisong Yue. Near-optimal machine teaching via explanatory teaching sets. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1970–1978. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/chen18g.html.

Eric Chu, Deb Roy, and Jacob Andreas. Are visual explanations useful? a case study in model-in-the-loop prediction, 2020.

Andy Cockburn, Carl Gutwin, and Alan Dix. *HARK No More: On the Preregistration of CHI Experiments*, pp. 1–12. Association for Computing Machinery, New York, NY, USA, 2018. ISBN 9781450356206. URL https://doi.org/10.1145/3173574.3173715.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *CoRR*, abs/1410.8516, 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Ann-Kathrin Dombrowski, Jan E Gerken, and Pan Kessel. Diffeomorphic explanations with normalizing flows. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021. URL https://openreview.net/forum?id=ZBR9EpEl6G4.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv: 1702.08608*, 2017.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, pp. 9277–9286, 2019.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pp. 80–89. IEEE, 2018.

Yash Goyal, Amir Feder, Uri Shalit, and Been Kim. Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165*, 2019.

Frederik Hvilshøj, Alexandros Iosifidis, and Ira Assent. Ecinn: Efficient counterfactuals from invertible neural networks, 2021.

Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJsjkMb0Z.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Vaughan Jennifer Wortman. Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pp. 1–14, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080. doi: 10.1145/3313831.3376219. URL https://doi.org/10.1145/3313831.3376219.

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

Diederik P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *ArXiv*, abs/1807.03039, 2018.

John D. Lee and Katrina A. See. Trust in Automation: Designing for Appropriate Reliance. *Human Factors*, 46(1):50–80, March 2004. ISSN 0018-7208. doi: 10.1518/hfes.46.1.50_30392.

Shusen Liu, Bhavya Kailkhura, Donald Loveland, and Yong Han. Generative counterfactual introspection for explainable deep learning, 2019.

Radek Mackowiak, Lynton Ardizzone, Ullrich Köthe, and Carsten Rother. Generative classifiers as a basis for trustworthy computer vision. *arXiv preprint arXiv:2007.15036*, 2020.

Silvan Mertes, Tobias Huber, Katharina Weitz, Alexander Heimerl, and Elisabeth André. Ganterfactual - counterfactual explanations for medical non-experts using generative adversarial learning, 2020.

Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Interpretable Machine Learning – A Brief History, State-of-the-Art and Challenges. In *ECML PKDD 2020 Workshops*, Communications in Computer and Information Science, pp. 417–431, Cham, 2020. Springer International Publishing. ISBN 978-3-030-65965-3. doi: 10.1007/978-3-030-65965-3_28.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "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*, 2016a.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016b.

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Axel Sauer and Andreas Geiger. Counterfactual generative networks, 2021.

Hua Shen and Ting-Hao Kenneth Huang. How Useful Are the Machine-Generated Interpretations to General Users? A Human Evaluation on Guessing the Incorrectly Predicted Labels. In *Proceedings of the Eighth AAAI Conference on Human Computation and Crowdsourcing (HCOMP-20)*, volume 8, pp. 168–172, Virtual, October 2020. AAAI Press. ISBN 978-1-57735-848-0.

Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. Explanation by progressive exaggeration. In *International Conference on Learning Representations*, 2020.

Aaron Springer, Victoria Hollis, and Steve Whittaker. Dice in the Black Box: User Experiences with an Inscrutable Algorithm. *AAAI Spring Symposium Series*, 2017.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328. JMLR.org, 2017.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2), 2018.

Jennifer Wortman Vaughan and Hanna Wallach. A human-centered agenda for intelligible machine learning. This is a draft version of a chapter in a book to be published in the 2020 - 21 timeframe., November 2020. URL `https://www.microsoft.com/en-us/research/publication/a-human-centered-agenda-for-intelligible-machine-learning/`.

Mengjiao Yang and Been Kim. Benchmarking attribution methods with relative feature importance, 2019.

Kyra Yee, U. Tantipongpipat, and Shubhanshu Mishra. Image cropping on twitter: Fairness metrics, their limitations, and the importance of representation, design, and agency. *ArXiv*, abs/2105.08667, 2021.

Ruihan Zhang, Prashan Madumal, Tim Miller, Krista A. Ehinger, and Benjamin I. P. Rubinstein. Invertible concept-based explanations for cnn models with non-negative concept activation vectors, 2021.

Table 3: Importance of the data generating factors to the model's prediction. For the R^2 score, we fitted an ordinary least squares from the factors' deltas to the deltas of the model's logits and then report the coefficient of determination (R^2). The *Mean Logit Change* reports the median of the absolute change in logit values. The *Prediction Flip* column quantifies how often the model's prediction changed the sign when changing the attribute.

Factor	Prediciton Flip [%]	Median Logit Change	R^2
Legs' Position	41.680	2.493	0.933
Color	7.080	0.886	0.751
Shape	3.920	0.577	0.307
Position Y	2.960	0.597	0.007
Background	2.640	0.523	0.006
Rotation Yaw	3.480	0.669	0.001
Rotation Roll	2.260	0.413	0.001
Bending	3.640	0.605	0.000
Rotation Pitch	3.500	0.627	0.000
Position X	3.380	0.581	0.000

Table 4: *Norm* quantifies the length of feature map changes ($\Delta \boldsymbol{h} = f(x) - f(\hat{x})$) after resampling the different data generative factors. *Angle w. Clas.* quantifies the mean angle (in degrees) between $\Delta \boldsymbol{h}$ and the classifier weight $\boldsymbol{w}$.

Factor	Norm	Norm Std.	Angle w. Clas.	Angle Std.
Legs' Position	70.8	2.6	79.9	1.9
Color	53.8	2.4	85.3	1.3
Shape	66.9	2.1	88.5	1.5
Position Y	68.1	2.8	89.7	1.7
Background	60.3	2.6	89.7	1.5
Bending	68.9	2.6	89.8	1.7
Rotation Yaw	68.5	2.7	89.8	1.7
Rotation Roll	55.2	2.9	89.9	1.2
Position X	68.4	3.0	89.9	1.6
Rotation Pitch	69.5	2.5	90.0	1.5

A Appendix: Technical Details

A.1 Two4Two Dataset Details

Factor	Range	Distribution	Biased	Additional Class Information
Legs' Position	[0, 1]	Uniform with overlap	Yes	-
Color	[0, 1]	See Figure 3	Yes	Yes
Shape	[0, 1]	See Figure 3	Yes	No
Position Y	[-0.8, 0]	Uniform	No	No
Position X	[-0.8, 0]	Uniform	No	No
Background	$[0.05, 0.95]$	Uniform	No	No
Rotation Yaw	$[0, 2\pi]$	Uniform	No	No
Rotation Roll	$[-\pi/4, \pi/4]$	Truncated Normal(0, $0.03\pi/4$)	No	No
Rotation Pitch	$[-\pi/6, \pi/6]$	Truncated Normal(0, π / 8)	No	No
Bending	$[-\pi/10, \pi/10]$	Truncated Normal(0, π / 20)	No	No

Table 5: Distribution of each attribute in the study's dataset. *Biased* denotes whether an attribute is unequally distributed for the two classes. *Additional Class Information* show if an attribute contains any additional information about the target class not already given by the legs' position.

Table 6: Two4Two: Effect of interpolating along the weight vector.

Attribute	Mean Maximal Change	Std.
Legs' Position	0.662	0.140
Color	0.440	0.190
Shapes	0.624	0.208
Bending	0.059	0.042
Background	0.045	0.044
Rotation Pitch	0.186	0.126
Rotation Yaw	0.102	0.182
Rotation Roll	0.003	0.001
Position X	0.105	0.078
Position Y	0.103	0.078

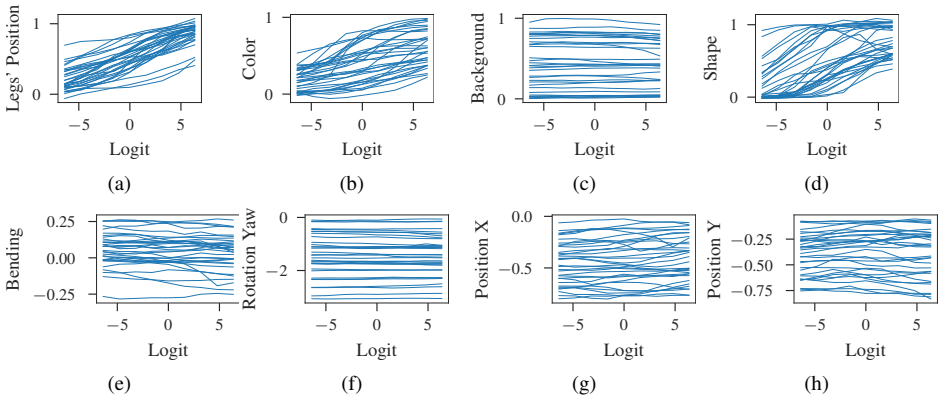


Figure 6: All attribute values as predicted by an observer convnet for sequences of counterfactual interpolations. Each line corresponds to a single sample whose logit score is modified through linear interpolations in classifier space.

```python
import dataclasses
import numpy as np
import matplotlib.pyplot as plt

from two4two.blender import render
from two4two.bias import Sampler, Continouos
from two4two.scene_parameters import SceneParameters

@dataclasses.dataclass
class RotationBiasSampler(Sampler):
    """A rotation-biased sampler.

    The rotation is sampled conditionally depending on the object type.
    Positive rotations for peaky and negative rotations for stretchy.
    """

    obj_rotation_yaw: Continouos = dataclasses.field(
        default_factory=lambda: {
            'peaky': np.random.uniform(-np.pi / 4, 0),
            'stretchy': np.random.uniform(0, np.pi / 4),
        })

# sample a 4 images
sampler = RotationBiasSampler()
params = [sampler.sample() for _ in range(4)]
for img, mask, param in render(params):
    plt.imshow(img)
    plt.title(f"{param.obj_name}: {param.obj_rotation_yaw}")
    plt.show()
```

Listing 1: Source code example to create a biased sampler. High positive rotations are predictive of Stretchy and low negative rotations of Peaky.

A.2 Architecture of the Invertible Neural Network

Our model is based on the Glow architecture (Kingma & Dhariwal, 2018) and contains 7 blocks. A block is a collection of 32 flow steps, followed by a down-sampling layer, and ends with a fade-out layer. A single flow step consists of *actnorm*, *invertible* 1×1 *convolution* and *affine coupling* layer. The down-sampling keeps all dimensions, e.g. a shape of (h, w, c) becomes $(h/2, w/2, 4c)$. The fade-out layer maps removes half of the channels. The out-faded channels are than mapped to a standard normal distribution to compute the unsupervised loss. For generating counterfactuals, the out-faded values are not thrown away but rather stored to be used when computing the inverse.

The model is trained using a supervised loss and an unsupervised objective. In total our model had 687 layers and 261 million parameters. The classifier used the output of layer 641. The remaining layers 642-687 were optimized using the standard unsupervised flow objective. For the first 641 layers, we also trained on the classifier's supervised loss.

Let φ denote the first 641 layers and $\mu : \mathbb{R}^n \mapsto \mathbb{R}^n$ the last. We train φ both on a supervised loss from the classifier $f(\boldsymbol{x})$ and an unsupervised loss from matching the prior distribution $\mathcal{N}(0, I)$ and the log determinante of the Jacobian. μ is only trained on the unsupervised loss:

$$\arg \min_{\theta_\varphi, \theta_\mu, \theta_f} L_{\text{un}}(\mu \circ \varphi(\boldsymbol{x})) + \beta \, L_{\text{sup}}(\boldsymbol{w}^T \varphi(\boldsymbol{x}) + b, \, y_{\text{true}}). \tag{1}$$

For the supervised loss L_{sup}, we use the binary cross entropy. As unsupervised loss L_{un}, we use the commonly used standard flow loss obtained from the change of variables trick Dinh et al. (2016). The unsupervised loss ensures that inverting the function results in realistic looking images and can also be seen as a regularization.

The layer 342 used for the concept explanations is an affine coupling layer.

Table 7: Test performance of the supervised trained model MobileNet-V2 measured using a mean squared error (MSE).

Attribute	Test MSE
Legs' Position	0.0008912
Bending	0.0001915
Background	0.0001128
Color	0.0005297
Rotation Pitch	0.0009235
Rotation Roll	0.0005622
Rotation Yaw	0.002243
Position X	0.0004451
Position Y	0.0003912
Shapes	0.001102

A.3 SUPERVISED MOBILENET-V2

We used a MobileNet-V2 to predict the attributes values. The models test mean squared errors are denoted in Table 7.

B LINKS TO MODEL, DATASET AND STUDY

Model export:
`https://f002.backblazeb2.com/file/iclr2022/do_users_benefit_from_interpretable_vision_model.tar.gz`

Unbiased dataset:
`https://f002.backblazeb2.com/file/iclr2022/two4two_obj_color_and_spherical_finer_search_spherical_uniform_0.33_uniform_0.15_unbiased.tar`

Biased dataset:
`https://f002.backblazeb2.com/file/iclr2022/two4two_obj_color_and_spherical_finer_search_spherical_uniform_0.33_uniform_0.15.tar`

Export of study:
`https://f002.backblazeb2.com/file/iclr2022/ICLR2022_Export_Do_Users_Benefit_From_Interpretable_Vision.qsf`

PDF print-out of the study:
`https://f002.backblazeb2.com/file/iclr2022/ICLR2022_Export_Do_Users_Benefit_From_Interpretable_Vision.pdf`

C USER-STUDY LINKS AND VIDEOS

The studies can be accessed on the Qualtrics platform (with anonymized consent form) under the following links:

Baseline condition:
`https://wznbm.qualtrics.com/jfe/form/SV_7Umdmdaq8EHVRm6`

INN condition:
`https://wznbm.qualtrics.com/jfe/form/SV_dneHADG7BxjVurc`

Concepts condition:
`https://wznbm.qualtrics.com/jfe/form/SV_0PWErBQmGL0lobk`

The tutorial videos can be viewed under the following links:

Introduction Tutorial for Peeky and Stretchy:
`https://f002.backblazeb2.com/file/iclr2022/Intro_Peeky_Stretchy.mp4`

Second Introduction Tutorial for ML and Biases:
`https://f002.backblazeb2.com/file/iclr2022/Second_Intro_ML.mp4`

Tutorial for baseline condition:
`https://f002.backblazeb2.com/file/iclr2022/condition_BASE.mp4`

Tutorial for concept condition:
`https://f002.backblazeb2.com/file/iclr2022/condition_CONCEPTS.mp4`

Tutorial for INN condition:
`https://f002.backblazeb2.com/file/iclr2022/condition_INN.mp4`

D USER-STUDY PREREGISTRATION AND HYPOTHESIS

The Preregistrations can also be viewed under the following URLs:

- Validation of TWO4TWO: `https://aspredicted.org/blind.php?x=/62X_15J`
- Study 2: Concepts vs. Baseline and INN vs. Baseline: `https://aspredicted.org/blind.php?x=/7XN_77P`

We also paid the participants in both studies the more lucrative tariff included in the preregistration of study 2, e.g. for third comprehension task: 3.50GBP and so on.

D.1 VALIDATION OF TWO4TWO

1) Have any data been collected for this study already?

No, no data have been collected for this study yet.

2) What's the main question being asked or hypothesis being tested in this study?

This study investigates whether users identify biases learned by a neural network. The neural networks task is to discriminate between two abstract animals (”Peeky” and ”Stretchy”). Each participant is presented with predictions of the system in a 10x5 image grid.

After an initial tutorial phase, the participants have to find biases in the model. They do this by scoring different characteristics as relevant or irrelevant. The characteristics are: ”legs position relative to the spine (LEGS)”, ”object color (COLOR)”, ”background (BACK)”, ”rounded or rectangular shape of the blocks (SHAPE)”, and ”rotation and bending (ROT)”.

The main research question is whether we succeeded in creating a model that contains at least one bias that is hard to detect, i.e. either COLOR or SHAPE should be harder to detect than LEGS.

HB: Participants can identify the biases in COLOR or SHAPE less frequently than LEGS.

3) Describe the key dependent variable(s) specifying how they will be measured.

Participant will answer the following questions:

- LEGS: How relevant is the legs position relative to the spine for the system?: Relevant / Irrelevant
- COLOR: How relevant is the color of the animal for the system? Relevant / Irrelevant
- BACK: How relevant is the background of the animal for the system? Relevant / Irrelevant
- SHAPE: How relevant is the rounded or rectangular shape of the animal's blocks for the system? Relevant / Irrelevant
- ROT: How relevant is the rotation and bending of the animal for the system? Relevant / Irrelevant

The ground truth answer is that LEGS, COLOR, SHAPE are relevant while BACK and ROT are irrelevant. Our first dependent variable is the number of times the head position was selected as relevant. Our second dependent variable is the number of times the color of the animal was selected as relevant. Our third dependent variable is the number of times the rounded or rectangular shape of the animal's blocks was selected as relevant.

4) How many and which conditions will participants be assigned to?

Our study follows a within-subject design and has only one condition. We first show the participants introductory videos about the two abstract animals, the machine learning system, and some guidance on how to interpret the predictions of the system. Each video is accompanied by a written summary. We then show the predictions of the system in a grid of images: 10 sorted rows of 5 images drawn from the validation set (50 original images). Each of the five columns represents the neural netwoks's logit range. Similarly rated images are assigned to the same column.

5) Specify exactly which analyses you will conduct to examine the main question/hypothesis.

We will conduct two exact one-sided McNemar-tests with LEGS acting as our control: one between SHAPE and LEGS and a second between COLOR and LEGS. We will use a one-sided test as we expect that SHAPE and COLOR are harder to identify. The significance level of both tests will be Bonferroni adjusted to $\alpha = 0.025$.

6) Describe exactly how outliers will be defined and handled, and your precise rule(s) for excluding observations.

We reject participants with low effort responses or who failed to understand the dataset, machine learning concept, or explanation method. We have implemented hard-coded exclusion criteria directly in the survey (implemented with Qulatrics and Prolific).

- did not finish experiment at all or in under 77 minutes
- did not watch tutorial videos completely (there are 3 videos) or failed a multiple-choice comprehension test twice (there are four such tests), unless participants explicitly ask us to retake the study
- using a device smaller than a tablet (min. 600 px in width or height)
- provided answers about relevant characteristics in under 30 seconds
- withdrawn data consent / returned task on Prolific
- circumvented Qualtrics protection against retaking the entire survey again (first complete submission will be counted)

We do not plan to exclude any participants who passed all of the above criteria unless the qualitative answers reveal a serious misunderstanding of the study instructions that the multiple choice tests did not cover. We will report such exclusions in detail in the Appendix.

7) How many observations will be collected or what will determine sample size?

No need to justify decision, but be precise about exactly how the number will be determined. 50 participants from Prolific with the background:

- Fluent in English
- Hold an academic degree
- Prolific approval rate of at least 90%
- Did not participate in pilot studies
- Passed hard coded exclusion criteria (see 8).

We pay participants max. 8.00 GBP (6.00 GBP base salary + 2.00 GBP max bonus). For those failing any comprehension questions or not watching the video, we pay:

- First comprehension task: no compensation
- Second comprehension task: 0.5 GBP
- Third comprehension task: 1.75 GBP
- Failed to watch first video: no compensation
- Failed to watch second video: 1 GBP
- Failed to watch third video: 2 GBP

8) Anything else you would like to pre-register? (e.g., secondary analyses, variables collected for exploratory purposes, unusual analyses planned?) We ask participants to answer three multiple choice comprehension tests in the form of true/false statements to ensure that they understood the task and the dataset. We also ask them to provide some free-text justification of why they chose a relevant / irrelevant rating to the questions in Section 3.

D.2 STUDY 2: CONCEPTS VS. BASELINE AND INN VS. BASELINE

1) Have any data been collected for this study already?

No, no data have been collected for this study yet.

2) What's the main question being asked or hypothesis being tested in this study?

This study investigates whether users identify biases learned by a neural network. The neural networks task is to discriminate between two abstract animals ("Peeky" and "Stretchy"). Each participant is presented one of three different explanation methods: baseline (B), counterfactuals obtained using invertible neural networks (CF) and prototypes (P).

Each participant is randomly assigned to a method. After an initial tutorial phase, the participants have to find biases in the model. They do this by scoring different characteristics as relevant or irrelevant. The characteristics are: "legs position relative to the spine (LEGS)", "object color (COLOR)", "background (BACK)", "rounded or rectangular shape of the blocks (SHAPE)", and "rotation and bending (ROT)".

The main question of our study is whether the participants can correctly identify relevant and irrelevant attributes using these explanation methods (B, CF, P). This is reflected by two hypotheses:

H1: *Participants identify relevant and irrelevant attributes with less accuracy using P compared to B.*

H2: *Participants identify relevant and irrelevant attributes with higher accuracy using CF compared to B.*

3) Describe the key dependent variable(s) specifying how they will be measured.

Participant will answer the following questions:

- LEGS: How relevant is the legs position relative to the spine for the system?: Relevant / Irrelevant
- COLOR: How relevant is the color of the animal for the system? Relevant / Irrelevant
- BACK: How relevant is the background of the animal for the system? Relevant / Irrelevant
- SHAPE: How relevant is the rounded or rectangular shape of the animal's blocks for the system? Relevant / Irrelevant
- ROT: How relevant is the rotation and bending of the animal for the system? Relevant / Irrelevant

The ground truth answer is that LEGS, COLOR, SHAPE are relevant while BACK and ROT are irrelevant. Our dependent variable is the percentage of correctly answered questions per participant (accuracy, which is computed as (true positives + true negatives)/number of total answers).

4) How many and which conditions will participants be assigned to?

We run a between-subject study, with randomly but equally assigned participants to 1 of 3 conditions. We first show introductory videos about the two abstract animals, the machine learning system, the explanation technique and some guidance on how to interpret the technique. Each video is accompanied by a written summary. We then show a grid of (10x5) images:

1. B: NN predictions explained with 10 sorted rows of 5 images drawn from the validation set (50 original images). Each of the five columns represents a score range. Similarly rated images are assigned to the same column.

2.CF: Same grid layout as B, but the NN is explained by counterfactual interpolations. Each row contains interpolations which change the prediction of the NN to fit the designated score. Original images are used as starting points but are not shown.

3.P: We found concepts based on the work by (Zhang et al., 2020). Each row shows a set of relevant concepts. We only used concepts correlated with at least r=0.2 with the model logit values. In total, we display 10 rows where each row contains a concept. Each row contains a set of 5 example images for which the concept is relevant.

(Zhang et al., 2020) https://arxiv.org/abs/2006.15417

5) Specify exactly which analyses you will conduct to examine the main question/hypothesis.

We will compute the accuracy scores for each participant and then compare the accuracy scores between the conditions. We expect the data to be non-normally distributed, and will test this assumption using a Shapiro-Wilk test with a significance level of $\alpha = 0.05$. If our assumption is true, we plan to conduct a Kruskal-Wallis test, followed by post-hoc analysis using Wilcoxon's-rank-sum tests for focused comparison between the groups CF and B (expecting higher accuracy in CF) and P and B (expecting lower accuracy in P).

If the data is normally distributed, we will conduct a one-way ANOVA with planned contrasts, if the following assumptions of ANOVAs are met:

- Homogeneity of the variance of the population (assessed with a Levene-Test with a significance level of $\alpha = 0.05$.)

If the homogeneity of variance assumption of ANOVA is violated (assessed with a Levene-Test with a significance level of $\alpha = 0.05$.), we plan to perform Welch's Anova.

6) Describe exactly how outliers will be defined and handled, and your precise rule(s) for excluding observations.

We reject participants with low effort responses or who failed to understand the dataset, machine learning concept, or explanation method. We have implemented hard-coded exclusion criteria directly in the survey (implemented with Qulatrics and Prolific).

- did not finish experiment at all or in under 77 minutes
- did not watch the tutorial videos completely (there are 3 videos) or failed a multiple-choice comprehension test twice (there are four such tests), unless participants explicitly ask us to retake the study
- using a device smaller than a tablet (min. 600 px in width or height)
- provided answers about relevant characteristics in under 30 seconds
- withdrawn data consent / returned task on Prolific
- circumvented Qualtrics protection against retaking the entire survey again (first complete submission will be counted)

We do not plan to exclude any participants who passed all of the above criteria unless the qualitative answers reveal a serious misunderstanding of the study instructions that the multiple choice tests did not cover. We will report such exclusions in detail in the Appendix.

7) How many observations will be collected or what will determine sample size? No need to justify decision, but be precise about exactly how the number will be determined.

240 (80 per condition) participants from Prolific with the background:

- Fluent in English
- First, we sample participants with an academic degree. If we do not reach the desired participant number, which is likely given the limited availability of such subjects, we will supplement with participants with an academic degree in other subjects. All participants will be randomly and equally split into the 4 conditions.
- Prolific approval rate of at least 90%
- Did not participate in pilot studies
- Passed hard coded exclusion criteria (see 8).

We pay participants max. 6.50 GBP (4.50 GBP base salary + 2.00 GBP max bonus). For those failing any comprehension questions or not watching the video, we pay:

- First comprehension task: 0.5 GBP
- Second comprehension task: 1.75 GBP

- Third comprehension task: 3.50GBP
- Failed to watch first video: no compensation
- Failed to watch second video: 1 GBP
- Failed to watch third video: 2 GBP

8) Anything else you would like to pre-register? (e.g., secondary analyses, variables collected for exploratory purposes, unusual analyses planned?)

In a previous study, we collected 50 responses for the baseline condition only (Preregistration #75056)). We do not plan to use the data for this study.

We ask participants to answer three multiple choice comprehension tests in the form of true/false statements to ensure that they understood the task and the dataset. We also ask them to provide some free-text justification of why they chose a relevant / irrelevant rating to the questions in Section 3.

Additionally, we ask the participants about their machine learning expertise level. Participants can rate their expertise as: complete novice, some expertise, or expert in the topic. We plan to use descriptive statistics to see how accuracies change per condition for each expertise level and how expertise was distributed within our sample.

We are also planning a qualitative thematic analysis of the open-text questions in our survey via open and axial coding, with the aim of understanding how participants integrated explanations in their reasoning about the relevance of attributes.

7 DNNR: Differential Nearest Neighbor Regression

In cases where deep neural networks and explanations for them do not work efficiently, it is worth exploring simpler methods with more interpretable outputs. This is especially true for tabular data, where the number of features is limited. Tabular data remains the most common data type for many practical applications.

With this motivation, (Nader et al. 2022) introduced a novel method for regression tasks called Differential Nearest Neighbor Regression (DNNR). DNNR extends the k-nearest neighbor regression method with higher-order derivatives, offering a more straightforward and interpretable approach compared to deep neural networks. In this chapter, we will discuss the development, theoretical bounds, and experimental results of DNNR, as well as its advantages and limitations.

DNNR's simplicity enables the derivation of theoretical bounds on generalization error. The theoretical bounds are an extension of the bounds for the k-nearest neighbor regression

Furthermore, the training points responsible for the prediction can be inspected directly. When using first-order derivatives, one can inspect also the corresponding linear coefficients. The advantage of DNNR is that this linear approximation is made by the prediction algorithm itself, and not by an explanation algorithm. Thus, these linear coefficients must reflect the importance of the features for the prediction.

While DNNR provides a simple and interpretable model, it does not apply to every problem. In particular, DNNR cannot work with datasets containing thousands of dimensions. In this case, deep neural networks are more suitable. However, for tabular data, DNNR is a promising alternative to deep neural networks and provides a more interpretable model than gradient boosting methods.

This work is relevant for my thesis because it shows how simple models allow us to derive theoretical guarantees.

DNNR: Differential Nearest Neighbors Regression

Youssef Nader [*1] **Leon Sixt** [*1] **Tim Landgraf** [1]

Abstract

K-nearest neighbors (KNN) is one of the earliest and most established algorithms in machine learning. For regression tasks, KNN averages the targets within a neighborhood which poses a number of challenges: the neighborhood definition is crucial for the predictive performance as neighbors might be selected based on uninformative features, and averaging does not account for how the function changes locally. We propose a novel method called Differential Nearest Neighbors Regression (DNNR) that addresses both issues simultaneously: during training, DNNR estimates local gradients to scale the features; during inference, it performs an n-th order Taylor approximation using estimated gradients. In a large-scale evaluation on over 250 datasets, we find that DNNR performs comparably to state-of-the-art gradient boosting methods and MLPs while maintaining the simplicity and transparency of KNN. This allows us to derive theoretical error bounds and inspect failures. In times that call for transparency of ML models, DNNR provides a good balance between performance and interpretability.[2]

1. Introduction

K-nearest neighbors (KNN) is an early machine learning algorithm (Cover & Hart, 1967) and a prototypical example for a transparent algorithm. Transparency means that a model's decision can be explained by inspecting of its parts. KNN's transparency follows from its simplicity: it can be expressed in simple terms as "the system averages the targets of the most similar points to the query". At the same time, the algorithm's simplicity makes it amenable for theoretical analysis, such as obtaining bounds on KNN's prediction error (Chaudhuri & Dasgupta, 2014).

However, KNN's predictive performance is limited. Most works aiming to improve KNN primarily focused on the selection of the neighbors, the distance metric, and the number of nearest neighbors k (Wettschereck & Dietterich, 1993; Weinberger & Tesauro, 2007; Cleary & Trigg, 1995). KNN's averaging scheme assumes that the target variable's changes are independent of those in the input features. Here, we introduce *Differential Nearest Neighbor Regression* (DNNR) to make use of that very gradient information. For each neighbor of a query point, we estimate the gradient of the function, and then – instead of averaging the targets – we average a Taylor approximation. KNN can then be seen as a zero-order Taylor approximation, while DNNR uses higher orders of the Taylor's theorem. A visual summary of the differences between KNN regression and DNNR can be found in Figure 1.

In a theoretical analysis, we derived a bound on the point-wise error of DNNR in relation to parameters such as the number of training points and the neighborhood size and found that the error bound favors DNNR over KNN. In an empirical evaluation on over 250 different datasets, we confirmed that DNNR outperforms KNN regression and performs on par with gradient boosting methods. An ablation study then confirmed that both the gradient-based prediction and the feature scaling contribute to the performance gains. Using a synthetic dataset generated with known underlying ground truth, we simulated the error bound and found that DNNR requires fewer training points than KNN. Furthermore, we present an investigation on a DNNR failure and showcase how the model's transparency can be used in such an analysis.

The regulation of Machine Learning algorithms in high-risk applications is being discussed globally or already under preparation (European Commission, 2021). We contribute to the overall goal of transparent and safer ML models in the following ways:

- We propose a new regression method (DNNR) that performs on par with state-of-the-art algorithms;
- DNNR is theoretically grounded: we provide a proof to bound DNNR's point-wise error (Theorem 1) and validate its usefulness empirically;

[*]Equal contribution [1]Department of Computer Science, Freie Universität Berlin, Germany. Correspondence to: Youssef Nader <youssef.nader@fu-berlin.de>, Leon Sixt <leon.sixt@fu-berlin.de>, Tim Landgraf <tim.landgraf@fu-berlin.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[2]For code, see https://github.com/younader/DNNR_paper_code

- An extensive evaluation against 11 methods on a set of 8 regression datasets, the PMLB benchmark (133 datasets), and Feynman symbolic regression (119 datasets);
- We provide detailed analyses to understand DNNR's performance (ablation study; impact of data properties) and transparency (inspection of failure cases).

2. Related Work

KNN is a non-parametric model based on a simple voting decision rule where the target of a given point is predicted by averaging the targets of neighboring samples (Cover & Hart, 1967). For an introduction to KNN, we refer the reader to (Chen & Shah, 2018).

Numerous methods have been proposed to improve this simple decision rule. (Kulkarni & Posner, 1995; Chaudhuri & Dasgupta, 2014) investigated the KNN convergence rate under different sampling conditions while (Balsubramani et al., 2019) and (Wettschereck & Dietterich, 1993) proposed different methods for an adaptive choice of k.

Although the choice of the number of neighbors is critical, it is not the only factor governing KNN performance. A large subset of the KNN literature proposed techniques for the choice of the distance metric that defines the neighborhood. (Cleary & Trigg, 1995) introduced an entropy-based distance metric, and (Wang et al., 2007) proposed an adaptive distance. Metric learning methods propose data-driven learning mechanisms for the distance function (Weinberger & Tesauro, 2007; Weinberger et al., 2006; Wang et al., 2018). A similar approach changes the data representation upon which the distance function operates via feature weighting or feature selection (Aha, 1998; Vivencio et al., 2007). (Bhatia & Vandana, 2010) provides a more comprehensive overview of the different KNN techniques. However, all these methods do not change how the prediction is being performed – all use an averaging scheme of the targets that effectively does not account for how the function changes within the local neighborhood.

A method that uses local changes is local linear regression (LL) (Fan, 1992). Similar to KNN, local linear regression selects k-nearest neighbors and then fits a hyperplane locally. This differs from our proposed method as we fit the gradient for all nearest neighbors separately. The single hyperplane of LL regression assumes an identical gradient for each neighbor. We show results for LL regression in the quantitative evaluation.

Gradient Approximation Estimating the gradient from data has been studied for various reasons, including variable selection and dimensionality reduction (Hristache et al., 2001; Mukherjee & Zhou, 2006). Several non-parametric methods exist to estimate the gradient from data (Fan & Gijbels, 1996; De Brabanter et al., 2013), and bounds of convergence already exist for some techniques (Berahas et al., 2021; Turner et al., 2010). An error bound on L1-penalized gradient approximation using KNN is derived in (Ben-Shabat & Gould, 2020) using local gradient approximation for 3D model reconstruction by fitting truncated jets in the KNN neighborhoods. (Ausset et al., 2021). The work also applied the estimated gradient to variable selection and gradient optimization problems, but did not utilize it to improve the prediction. As we will present in detail, our method combines non-parametric gradient estimation using KNN, Taylor expansion, and feature scaling for regression modeling. To our surprise, this combination was neither explored theoretically nor empirically before.

Algorithm 1 Pseudocode of DNNR's prediction for a query point X. The feature scaling is omitted in this pseudocode. The OLS function solves an ordinary least-squares problem.

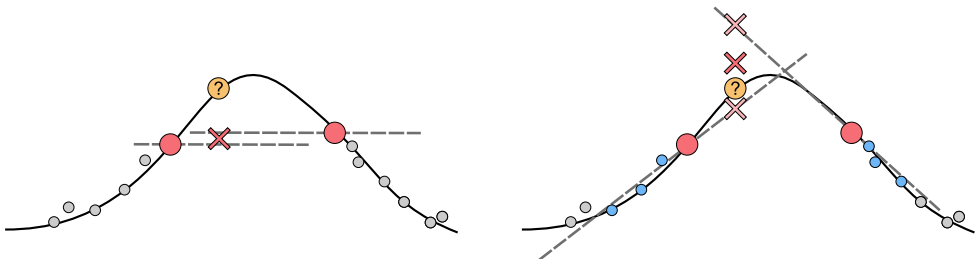
Require: a query point x, train data $\{(X_i, Y_i)\}$, nearest neighbors k, nearest neighbors for gradient estimation k', range for target value $y_{\min}, y_{\max}$:
$M \leftarrow$ nn(x, k)
\# M contains the indices of the k nearest neighbors
predictions = []
for each neighbor index m $\in M$ **do**
  A $\leftarrow$ nn(X_m, k')
  \# A contains indices of the k' neighbors for X_m
  $\Delta \boldsymbol{X} \leftarrow X_A - x_m$
  $\Delta \boldsymbol{Y} \leftarrow Y_A - y_m$
  $\hat{\gamma}_m \leftarrow$ OLS($\Delta \boldsymbol{X}, \Delta \boldsymbol{Y}$)
  \# $\hat{\gamma}_m$ approximates the gradient
  $\hat{y}_m \leftarrow y_m + \hat{\gamma}(x_m - x)$
  predictions.append($\hat{y}_m$)
end for
$\hat{y}$ = mean(predictions)
return clip($\hat{y}, y_{\min}, y_{\max}$)

3. Method

Notation We will consider the typical supervised regression problem. The training data is given as a set of n tuples of data points and target values $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$. We will denote the expected target value by $\eta(x) = \mathbb{E}[Y|X = x]$. A ball with radius r around x is given by $B_{x,r}$. For a ball around x with exactly k training points, we will use a $\#$ sign as in $B_{\boldsymbol{x},\#k}$. We summarize our notation in the Appendix Table 4.

DNNR Vanilla KNN predicts the target of a given datapoint x by averaging the targets of nearby training points:

$$\eta_{\text{KNN}}(x) = \frac{1}{k} \sum_{X_m \in B_{x,\#k}} Y_m. \tag{1}$$



(a) KNN (b) DNNR

Figure 1. **(a)** An illustration of KNN regression. To predict a value for a query (circle with question mark), the target values of the nearest points (red circles) are averaged. KNN's prediction is marked by the red cross. The other data points (gray circles) are not used for prediction. **(b)** Similar illustration of DNNR. The local gradient (gray dashed line) is estimated for each neighbor and a target value is interpolated linearly (light red crosses). The final prediction (red cross) is the average of these interpolated values.

This simple averaging scheme can be seen as a zeroth-term Taylor expansion around the inference point. If we would know the gradient of the function η, we could easily extend it to a first degree Taylor expansion:

$$\eta_{\text{known}\nabla}(x) = \frac{1}{k}\sum_{X_m \in B_{x,\#k}} \left(Y_m + \nabla\eta(X_m)(x - X_m)\right). \quad (2)$$

Of course, we only have access to the training data, not the underlying target function. Therefore, we approximate the gradient using nearby points. We can approximate $\nabla\eta(X_m)$ by solving a least-squares problem:

$$\hat{\gamma}_m = \arg\min_{\gamma_m} ||A\gamma_m - q||, \quad (3)$$

where $\hat{\gamma}_m \in \mathbb{R}^d$ is the estimated gradient at point X_m, $A \in \mathbb{R}^{k' \times d}$ contains the normalized differences $(X_i - X_m)/h_i$ as row vectors, $h_i = ||X_i - X_m||$, k' is the number of points used to approximate the gradient, i indexes these k' nearby points around X_m , and $q \in R^{k'}$ denotes the differences in the labels $q = (Y_i - Y_m)/h_i$. The result $\hat{\gamma}_m$ can then substitute the real gradient in equation (2) to yield the DNNR approximation:

$$\eta_{\text{DNNR}}(x) = \frac{1}{k}\sum_{X_m \in B_{x,\#k}} \left(Y_m + \hat{\gamma}_m(x - X_m)\right). \quad (4)$$

Using the approximate Taylor expansion, we aim to improve the prediction performance by also utilizing the local change in the function. The pseudocode of DNNR is listed in Algorithm 1. The algorithm can also be extended easily to higher-orders of a Taylor expansion. In the evaluation, we will report results using the diagonal of the Hessian, i.e. the elements corresponding to $\frac{\partial^2 \eta}{\partial^2 x_i}$.

Feature Weighting Using an isotropic distance weighs each dimension equally. This way, a neighbor may be picked based on irrelevant dimensions. A simple improvement is to scale the feature dimensions as done in previous work (Weinberger & Tesauro, 2007). We use a common approach for a distance metric d using a diagonal matrix W to scale each dimension:

$$d(x_i, x_j) = (x_i - x_j)^T W (x_i, x_j) \quad (5)$$

DNNR's predictions are differentiable w.r.t. the input dimensions, so a loss can be backpropagated to the scaling matrix W and optimized using gradient descent. Inspired by the Taylor theorem, we require that nearby points predict each other well while predictions of far points may come with a larger error. Therefore, the loss enforces a correlation between distance and the prediction error:

$$W^* = \arg\min_W \sum_{i,j \in I} \text{cossim}\big(d(X_i, X_j), \left|Y_i - \hat{\eta}_{\text{DNNR}}(X_{\text{nn}(i,k')})\right|\big), \quad (6)$$

where I is an index set of nearby points, and cossim denotes the cosine similarity.

At first sight, an alternative might have been minimizing the prediction's mean squared error (MSE). However, minimizing the MSE would not alter the scaling, as the prediction is scale-invariant, i.e. downscaling a dimension will increase the gradient, and the prediction will stay the same. Therefore, a spatial inductive bias in equation 6 is needed.

4. Theoretical Analysis

We focus on the point-wise error estimate of DNNR vs. KNN regression. The proof contains two parts: the approximation error of the gradient and the point-wise prediction error. In Appendix C.2, we show that:

Lemma 1 *Let $f : D \subset \mathbb{R}^d \to \mathbb{R}$ be of class C^μ, $a \in D$, and $\mathcal{B}(a) \subset D$ be some neighborhood of a. Suppose that around point a we have m neighboring points v_k, $k = 1, \ldots, m$ with $a, v_1, \ldots, v_m \in \mathcal{B}(a) \subset D$. Suppose further that all μ-th order derivatives are Lipschitz, $i \in 1 \ldots \mu$:*

$\frac{\partial^i f}{\partial a_1^{l_1} \ldots \partial a_d^{l_d}} \in Lip_{\vartheta_i}(\mathcal{B}(a))$ *where* $l_1 + \ldots + l_d = i$ *and we approximate the gradient locally at* a *by* $\hat{\gamma} = E_1 \hat{\omega}$ *via the least-squares solution* $\hat{\omega} = \arg\min_{\omega \in \mathbb{R}^d} ||A\omega - q||$, *where*

$$A = \begin{pmatrix} \nu_1^T & \nu_{1*}^T \\ \nu_2^T & \nu_{2*}^T \\ \vdots & \\ \nu_m^T & \nu_{m*}^T \end{pmatrix}, \quad q = \begin{pmatrix} \frac{f(a+h_1\nu_1)-f(a)}{h_1} \\ \frac{f(a+h_2\nu_2)-f(a)}{h_2} \\ \vdots \\ \frac{f(a+h_m\nu_m)-f(a)}{h_m} \end{pmatrix}, \quad (7)$$

$A \in \mathbb{R}^{m \times p}$, $q \in \mathbb{R}^m$, $E_1 = \begin{pmatrix} I_d & 0 \end{pmatrix} \in \mathbb{R}^{d \times p}$, $h_k = ||v_k - a||$ *with* $h_k \nu_k = v_k - a$; $\nu_k^T = (\nu_{1_k}, \ldots, \nu_{d_k})$, ν_{k*}^T *denotes higher-order terms* $\nu_{k*}^T = \left(\frac{h_k^{\mu'-1}}{\mu'!} \binom{\mu'}{l_1, \ldots, l_d} \prod_{i_1}^d \nu_i^{k_i} \right)_{2 \le \mu' \le \mu, l_1 + \ldots l_d = \mu'}$ *and* $p = \sum_{i=1}^{\mu} \frac{(d+\mu)!}{d!}$. *Then a bound on the error in the least-squares gradient estimate is given by:*

$$||\nabla f(a) - \hat{\gamma}||_2 \le \frac{\vartheta_{\max} h_{\max}^{\mu}}{\sigma_1 (\mu+1)!} \sqrt{\sum_{i=1}^{m} ||\nu_i||_1^{2\mu}}, \quad (8)$$

where σ_1 *is the smallest singular value of A, which is assumed to have rank*$(A) = p$, $\vartheta_{\max} = \max_{i \in 1 \ldots k} \vartheta_i$ *and* $h_{\max} = \max_{1 \le k \le m} h_k$.

This lemma extends a result from the two-dimensional case (Turner et al., 2010). The gradient approximation depends on the Lipschitz constant $\vartheta_{\max}$, the distance to the neighbors $h_{\max}$, and the smallest singular value σ_1 of the normed differences A. The lemma also shows that by accounting for higher-order terms, e.g. picking $\mu > 1$, the gradient approximation can be made more accurate.

The following theorem is based on Theorem 3.3.1 in (Chen & Shah, 2018). We built on the same assumptions except requiring Lipschitz instead of Hölder continuity.

Technical Assumptions ($A^{\text{technical}}_{\mathcal{X},\rho,\mathbb{P}_X}$): The feature space $\mathcal{X}$ and distance ρ form a separable metric space. The feature distribution $\mathbb{P}_X$ is a Borel probability measure.

Assumption Besicovitch($A^{\text{Besicovitch}}_{\eta}$): The regression function η satisfied the Besicovitch condition if $\lim_{r \downarrow 0} \mathbb{E}[Y|X \in B_{x,r}] = \eta(x)$ for x almost everywhere w.r.t. $\mathbb{P}_X$.

Assumption Lipschitz ($A^{\text{Lipschitz}(\vartheta_{\max})}_{\eta}$): The regression function η is Lipschitz continuous with parameter $\vartheta_{\max}$ if $|\eta(x) - \eta(x')| \le \vartheta_{\max} \rho(x, x')$ for all $x, x' \in \mathcal{X}$.

Using Lemma 1, we prove the following theorem in Appendix C.2:

Theorem 1 *(DNNR pointwise error) Under assumptions* $A^{technical}_{\mathcal{X},\rho,\mathbb{P}_X}$ *and* $A^{Besicovitch}_{\eta}$, *let* $x \in supp(\mathbb{P}_X)$ *be a feature vector,* $\varepsilon > 0$ *be an error tolerance in estimating the expected label* $\eta(x) = \mathbb{E}[Y|X = x]$, *and* $\delta \in (0, 1)$ *be a probability tolerance. Suppose that* $Y \in [y_{\min}, y_{\max}]$ *for some constants* $y_{\min}$ *and* $y_{\max}$. *There exists a threshold distance* $h^*_{DNNR} \in (0, \inf)$ *such that for any smaller distance* $h \in (0, h^*)$, *if the number of training points* n *satisfies:*

$$n \ge \frac{8}{\mathbb{P}_X(B_{x,h})} \log \frac{2}{\delta}, \quad (9)$$

and the number of nearest neighbors satisfies

$$\frac{2(y_{\max} - y_{\min})^2}{\varepsilon^2} \log \frac{4}{\delta} \le k \le \frac{1}{2} n \mathbb{P}_X(B_{x,h}), \quad (10)$$

then with probability at least $1 - \delta$ *over randomness in sampling the training data, DNNR regression at point* x *has error*

$$|\hat{\eta}_{DNNR}(x) - \eta(x)| \le \varepsilon. \quad (11)$$

If we know that $A^{Lipschitz(\vartheta_{\max})}_{\eta}$ *also holds, we can pick* h^*_{DNNR} *as:*

$$h^*_{DNNR} = \sqrt{\frac{\varepsilon}{\vartheta_{\max}(1+\tau)}}, \quad (12)$$

where $\tau = \mathbb{E}\left[\frac{\sqrt{\sum_{i=1}^m ||\nu_i||_1^{2\mu}}}{\sigma_1} \,\middle|\, X \in B_{x,h} \right]$. ν, σ_1, *and* $\vartheta_{\max}$ *are defined as in Lemma 1.*

The theorem states that we can bound the point-wise error locally with a high probability given that the conditions on n and k are fulfilled. Theorem 3.3.1 from (Chen & Shah, 2018) provides a KNN regression point-wise error bound (their theorem is in turn based on (Chaudhuri & Dasgupta, 2014)). For KNN, the restriction on the maximum distance $h^*_{\text{KNN}} = \frac{\varepsilon}{2\vartheta_{\max}}$, while the other conditions on sample size and probability are identical. To compare DNNR and KNN, it is beneficial to solve for the error tolerance ε.

$$\varepsilon_{\text{DNNR}} = h^2_{\text{DNNR}} \vartheta_{\max}(1+\tau), \quad (13)$$

and for KNN:

$$\varepsilon_{\text{KNN}} = 2\vartheta_{\max} h_{\text{KNN}}. \quad (14)$$

The influence of the different variables is as follows:

- Both depend on the Lipschitz constant $\vartheta_{\max}$ linearly.
- Distance to nearest neighbors: $h_{\max}$ vs $h^2_{\max}$. For DNNR, the error decreases quadratically in $h_{\max}$. When $h_{\max}$ becomes small, $h^2_{\max}$ will be even smaller.
- For DNNR, τ represents the error in estimating the gradient. As long $\tau < \frac{2}{h_{\max}} - 1$, the error tolerance of DNNR will be lower than for KNN. As $\tau \propto \frac{1}{\sigma_1}$, an ill-conditioned matrix A might increase DNNR's error.

5. Experiments

We compared DNNR against other methods, including state-of-the-art gradient boosting methods. First, we discuss

which baselines we compared against and the general experimental setup. Then, we present large-scale quantitative evaluations followed by an ablation study and further qualitative analyses.

5.1. Setup

We compared DNNR against state-of-the-art boosting methods (CatBoost, XGBoost, Gradient Boosted Trees (Dorogush et al., 2018; Chen & Guestrin, 2016)), classical methods such as (KNN, multi-layer perceptron (MLP), and TabNet, that is a deep learning approach for tabular data (Arik & Pfister, 2021). We also included KNN-Scaled, which uses DNNR's feature weighting but KNN's averaging scheme.

We used standard scaling for all datasets (zero mean, unit variance per dimension). Each model, except TabNet, was optimized using a grid search over multiple parameters, and the models were refit using their best parameters on the validation data before test inference. We ensured that each method had a comparable search space, which are listed in Appendix D). Due to its long training times, we had to skip the grid search for TabNet. Approximately 4.1k CPU hours were used to run the experiments. For the larger benchmarks (Feynman and PMLB), we followed the setup in (Cava et al., 2021). Our baselines report similar or slightly better performance than on SRBench[3].

5.2. Quantitative Experiments

Benchmark Datasets: The goal of this benchmark is to inspect DNNR's performance on eight real-world regression datasets: Yacht, California, Protein, Airfoil, Concrete, Sarcos, CO2 Emissions, and NOX emissions. The last two datasets are part of the Gas Emission dataset. All datasets were taken from the UCI repository (Dua & Graff, 2017), except California (Kelley Pace & Barry, 1997) and Sarcos (Rasmussen & Williams, 2006). These datasets were also used in previous work (Bui et al., 2016). Some datasets also have discrete features (Yacht, Airfoil), which challenges DNNR's assumption of continuity. All the datasets were evaluated using a 10-fold split, except for the Sarcos dataset which comes with a predefined test set. Additionally, we fixed the data leakage in the Sarcos dataset by removing all test data points from the training set.

In Table 1, we report the averaged mean squared error (MSE) over the 10-folds for each dataset and model. Overall, CatBoost is the best performing method. We find that DNNR achieves the best performance on the Sarcos and California datasets and the second-order achieves the best performance on Protein. For NOxEmissions, CO^2Emissions, and Protein, DNNR is within 5% percentage difference to the best performing method (see Table 3). Discrete features violate

[3]See results here: https://cavalab.org/srbench/blackbox/

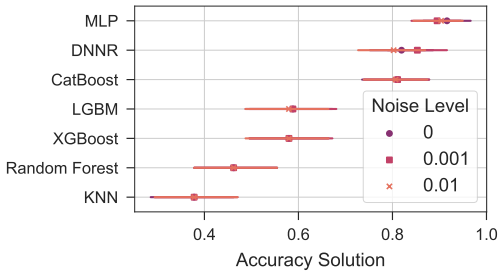


Figure 2. Accuracy on the Feynman Symbolic Regression Database under three levels of noise. The marks show the percentage of solutions with $R^2 > 0.999$. The bars denote 95% confidence intervals.

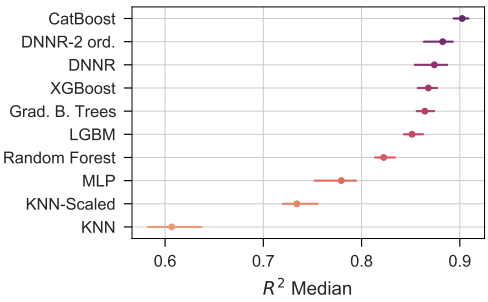


Figure 3. Results on the PMLB benchmark. The markers show the median R^2 performance over all datasets runs. Horizontal bars indicate the 95% bootstrapped confidence interval.

DNNR's assumptions, which can affect its performance, as the results on Airfoil and Yacht indicate. On these two datasets, DNNR is not under the best-performing methods (e.g. CatBoost: 1.25, XGBoost: 1.63, for Airfoil) but still performs better than vanilla KNN (2.30 vs. 4.25).

Discrete features can render the Taylor approximation meaningless, e.g. all neighbors may have the same value in one dimension rendering the gradient zero, or a linear approximation may not be sufficient when values exhibit large jumps. Interestingly, the second-order DNNR yields better results on the Airfoil and Concrete datasets, presumably because the second-order approximates sharp gradients better.

On the other datasets, DNNR delivers a significant improvement over KNN and also over KNN with DNNR feature scaling (KNN-Scaled).

Feynman Benchmark As the second benchmark, we selected the Feynman Symbolic Regression Database, which consists of 119 datasets sampled from classical and quantum physics equations (Udrescu & Tegmark, 2020). These equations are continuous differentiable functions. The difficulty can be increased by adding noise.

Table 1. The MSE on eight regression datasets averaged over 10-folds. The best-performing values are marked as bold. The standard deviations are given after the ± signs. For the Sarcos dataset, we only evaluate on the given test set.

	Protein	CO^2Emission	California	Yacht	Airfoil	Concrete	NOxEmission	Sarcos
CatBoost	11.82 ± 0.33	1.11 ± 0.40	**0.19 ± 0.01**	**0.25 ± 0.25**	**1.26 ± 0.24**	**14.00 ± 5.35**	14.65 ± 1.00	1.313
Grad. B. Trees	12.15 ± 0.41	1.25 ± 0.44	0.20 ± 0.01	0.33 ± 0.15	1.76 ± 0.51	16.03 ± 6.13	15.60 ± 1.00	1.813
LGBM	12.74 ± 0.35	1.17 ± 0.41	0.19 ± 0.01	6.86 ± 9.10	2.03 ± 0.38	16.10 ± 5.11	15.66 ± 1.00	1.613
MLP	14.34 ± 0.44	1.23 ± 0.47	0.35 ± 0.26	4.34 ± 10.21	10.17 ± 3.01	35.27 ± 12.17	19.91 ± 1.82	1.277
Rand. Forest	11.80 ± 0.25	1.12 ± 0.43	0.23 ± 0.02	1.13 ± 0.73	2.81 ± 0.66	22.75 ± 5.64	16.35 ± 1.03	2.264
XGBoost	12.01 ± 0.28	1.21 ± 0.44	0.20 ± 0.02	**0.26 ± 0.16**	1.63 ± 0.40	16.76 ± 6.68	14.99 ± 0.98	1.824
KNN	13.39 ± 0.37	1.17 ± 0.43	0.39 ± 0.03	68.46 ± 51.11	4.25 ± 0.96	60.76 ± 14.19	17.37 ± 1.26	1.752
KNN-Scaled	12.54 ± 0.57	1.20 ± 0.44	0.19 ± 0.02	2.66 ± 1.92	4.14 ± 0.94	40.25 ± 7.00	15.41 ± 1.21	1.770
LL	14.07 ± 0.18	1.20 ± 0.46	0.33 ± 0.07	50.83 ± 14.81	7.04 ± 1.39	51.40 ± 10.50	16.69 ± 1.10	0.792
LL-Scaled	12.90 ± 0.29	**1.10 ± 0.43**	0.21 ± 0.02	2.47 ± 1.70	3.53 ± 1.24	32.54 ± 6.33	14.57 ± 0.96	0.786
Tabnet	17.02 ± 2.68	1.22 ± 0.38	0.39 ± 0.04	2.83 ± 2.59	9.98 ± 3.04	43.73 ± 14.59	**12.97 ± 0.92**	1.304
DNNR	12.31 ± 0.35	1.12 ± 0.47	**0.19 ± 0.02**	1.05 ± 0.63	2.83 ± 0.60	36.52 ± 18.03	13.34 ± 0.96	**0.708**
DNNR-2 ord.	**11.64 ± 0.44**	1.24 ± 0.50	0.22 ± 0.02	0.48 ± 0.43	2.30 ± 0.48	28.35 ± 13.47	15.61 ± 1.94	0.727

The evaluation for the Feynman benchmark was executed with 10 different splits for each dataset and noise level (std=0, 0.001, 0.01) – similar to (Cava et al., 2021). For the first split, we divided the data into 70/5/25% train, validation, and test sets. The hyperparameter tuning was done with validation data of the first split. Then the models were refit using the best parameters and evaluated on the 25% test set. Subsequent splits (75/25% train/test) then used these hyperparameters.

For the Feynman benchmark, accuracy is defined as the percentage of datasets that were solved with a coefficient of determination $R^2 > 0.999$. We report this accuracy in Figure 2. DNNR is the second-best performing method after MLP. CatBoost's performance is also notable, with an accuracy of more than 80%. The different noise levels had minor effects on the methods.

PMLB Benchmark The PMLB benchmark contains real and synthetic datasets with categorical features, discrete targets, and noisy data in general. In total, we used 133 PMLB datasets. The evaluation setup for the PMLB datasets was similar to the Feynman benchmark. Figure 3 shows the median R^2 performance of the different models with a 95% confidence interval. CatBoost is the best performing method with an R^2 median > 0.9 with DNNR second-order closely in the second position. DNNR, XGBoost, and Gradient Boosted Trees perform similarly well. The worst-performing method is KNN regression. While adding feature weighting (KNN-Scaled) improves the R^2 median considerably by over 0.1, only DNNR's additional use of gradient information yields results comparable to gradient boosting methods.

5.3. Ablation

In the previous evaluations, we already included KNN-Scaled to measure the effect of the scaling versus the gradient information. We dissected DNNR even further and tested various design alternatives: such as scaling of the neighborhood (MLKR (Weinberger & Tesauro, 2007)) and regularization on the gradient estimation (Lasso (Ausset et al., 2021)). We based this analysis on the Airfoil, Concrete, and 5000 samples from Friedman-1 datasets (see Section 5.4). As before, we conducted a hyperparameters sweep for each model configuration and used a 10-fold validation for each dataset.

For the Concrete and Friedman-1 datasets, using only gradient information and no feature scaling (DNNR-Unsc.) already improved over KNN's performance. For the Airfoil dataset, which contains categorical features, using gradients without scaling (DNNR-Unsc.) leads to worse results. MLKR improves KNN's performance more than DNNR's scaling for KNN. However, when using gradient estimation, MLKR is less suitable as can be seen in the difference between DNNR and DNNR-MLKR on Airfoil and Concrete. MLKR draws apart local neighborhoods as points are scaled based on similarities in the target value. The inference might than be carried out by points that are drastically different from the query, both in L2-metric and a different gradient.

Furthermore, we would like to note that MLKR is also computationally more expensive than DNNR's scaling as they use Gaussian kernels resulting in a runtime quadratic in the number of samples. These results highlight that while the gradient information might be helpful for unscaled neighborhoods, scaling yields better gradients and results in better approximations.

Using Lasso regularization on the gradient estimation as done in (Ausset et al., 2021) did not perform well. We speculate that the regularization limits the gradient estimation. Future work might test if DNNR benefits from Lasso regularization in high-dimensional problems as motivated by the authors.

Table 2. Ablation study. We report the MSE for different variations of DNNR for three datasets.

	Airfoil	Concrete	Friedman-1
DNNR	2.83 ± 0.60	36.52 ± 18.03	**0.01 ± 0.00**
DNNR-Unsc.	4.82 ± 0.74	49.97 ± 13.59	1.03 ± 0.15
KNN-MLKR	3.32 ± 0.84	36.85 ± 9.89	0.40 ± 0.07
KNN-Scaled	4.14 ± 0.94	40.25 ± 7.00	7.27 ± 0.53
DNNR-MLKR	3.20 ± 1.08	1.5e13 ± 4e13	0.03 ± 0.00
KNN	4.25 ± 0.96	60.76 ± 14.19	3.93 ± 0.43
DNNR Rand.	24.38 ± 3.44	115.14 ± 20.52	6.07 ± 0.56
DNNR-Lasso	5.25 ± 1.05	40.24 ± 8.43	7.33 ± 0.52
DNNR-2 ord.	**2.30 ± 0.48**	**28.35 ± 13.47**	**0.01 ± 0.00**

5.4. Effect of noise, #samples, and #features

This analysis investigates how different data properties affect the model's performance. Such an analysis requires a controlled environment: we used the Friedman-1 dataset (Friedman, 1991). This dataset is based on the following equation:

$$y(x) = 10x_3 + 5x_4 + 20\,(x_2 - 0.5)^2 + 10\sin(\pi x_0 x_1) + s\epsilon, \tag{15}$$

where x_i is uniformly distributed in [0, 1], the noise ϵ is sampled from a standard normal distribution, and s controls the variance of the noise. Unimportant features can be added by simply sampling $x_j \sim U(0, 1)$. Friedman-1 allowed us to test the models under different sampling conditions: the number of samples, the magnitude of noise, and the number of unimportant features.

As defaults, we choose the number of samples = 5000, the number of features = 10, and the noise level = 0. Besides DNNR, we also evaluated Gradient Boosted Trees, CatBoost, Random Forest, MLP, and KNN. For each setting, we run a 5-fold evaluation (the hyperparameters of each method are fitted on the first fold and then fixed for the remaining 4).

We report the effect of each condition in Appendix Figure 8. For the number of samples, we note that DNNR's error declined rapidly. Second-best is CatBoost. For noise, we observed two groupings. While one group (MLP & KNN) performed poorly, their MSE did not increase when adding noise. For the better performing group (DNNR, Gradient Boosted Trees, CatBoost, Random Forest), the error did increase when adding noise. In this group, DNNR performed the best for low noise levels but was beaten by CatBoost slightly for higher levels of noise.

Increasing the number of unimportant features impacted KNN and MLP particularly. DNNR dropped from being the best method to sharing the second place with Gradient Boosted Trees as the feature scaling cannot entirely mitigate the effect of unimportant features. Tree-based methods were barely affected, as they are adept at handling unimportant features and operate on the information gain of each feature.

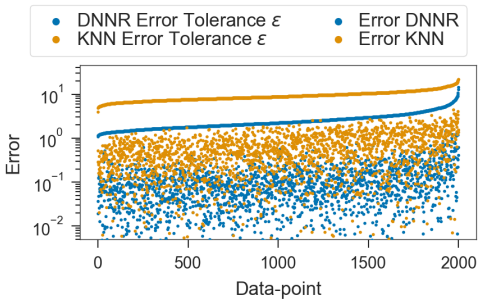


Figure 4. Comparison between the error bound of KNN (yellow) and DNNR (blue). On the x-axis, all test data points sorted by their error bounds are plotted. The DNNR performs better than KNN and the DNNR error bound is also lower.

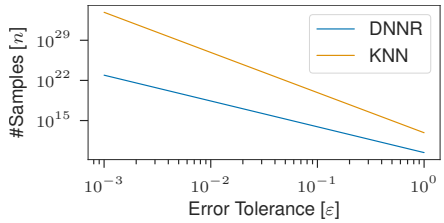


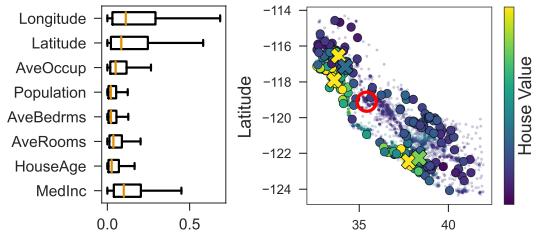
Figure 5. Depicts the error tolerance versus the number of training samples for the Friedman-1 dataset. KNN requires multiple orders more training points to guarantee the same error tolerance.

5.5. Application of the theoretical bound

In this evaluation, we analyze the error bounds from section 4. As dataset, we use the Friedman-1 dataset introduced before in section 5.4. The synthetic dataset allows the sampling of arbitrary points. Thereby we can also simulate very dense neighborhoods.

First, we apply Theorem 1 to the dataset. We pick a probability tolerance of 0.95 ($\delta = 0.05$), and a Lipschitz constant for equation (15) of $\vartheta_{\max} = 40$. For DNNR, we estimate a value of $\tau \approx 5.59$ from the data. We show the dependency between error tolerance and the number of samples required in Figure 5. In this exemplary calculation, KNN requires multiple orders of magnitude more training data than DNNR to achieve the same theoretical guarantee on the error tolerance. Still, even DNNR would require an unrealistic amount of samples, e.g. around 10^{15} for an error tolerance of $\varepsilon = 0.1$.

As a more practical application, we investigated how local conditions, e.g. distance to the neighbors and the Lipschitz constant, influence the local prediction error. Therefore, we sampled a realistically sized train and test set (10.000 and 2.000 samples) and then compared the error tolerance for KNN and DNNR according to equations (13) and (14). For both methods, we choose $k = 7$, and for the number of



(a) Feature relevances

(b) Failure (error of 4.85)

Figure 6. **(a)** Feature relevance of DNNR Unscaled on the California Housing dataset. **(b)** A failure of DNNR Unscaled. The red circle marks the query point. The prediction is done by approximating the gradient locally for each nearest neighbors (crosses). The circles visualize the points used for gradient approximation. As DNNR Unscaled weights each dimension equally, it does not use spatially nearby points, even though longitude and latitude are scored important.

neighbors to approximate the gradient, we used $k' = 32$. We purposely violate the conditions on k and n of Theorem 1, as we want to analyze how applicable the estimated error tolerances are in a more realistic setting. In Figure 4, we sorted the 2000 test points according to the error tolerances of KNN and DNNR respectively. For both methods, we see that the error tolerances strictly bound the actual errors by a gap of around one order of magnitude. We also observe that the error tolerances were correlated with the actual errors, e.g. as the error tolerances decrease, so does the actual error.

5.6. Feature importance & Inspecting a prediction

DNNR allows inspecting which neighbors were used for the prediction and how they contributed. For the following exemplary inspection, we use the California Housing dataset (Kelley Pace & Barry, 1997). The dataset's dependent variable is the median house value per block group (a block group is an aggregation of a local area). The eight observational variables represent the location, median income, and information about the houses, such as average rooms or occupation. For this study, we analyzed DNNR Unscaled, i.e. DNNR without the feature scaling. We omitted the feature scaling, as the feature relevance would be impacted by the scaling and the DNNR with scaling is also performing so well that the error would be minor to inspect.

First, we can provide a simple local feature relevance score by multiplying the estimated gradient with the difference in the input:

$$\boldsymbol{\xi}_m = |(\boldsymbol{x} - \boldsymbol{x}_m) \odot \hat{\gamma}_m|, \tag{16}$$

where x_m is the point where we estimate the gradient and $\hat{\gamma}_m$ the locally fitted gradient. This formulation of feature importance is analogous to a linear model where one would take $w \odot x$ (Molnar, 2019, sec. 5.1.). It is a known property that the gradient reflects the model's sensitivity and can be used for feature importance, and variable selection (Mukherjee & Zhou, 2006; Guyon & Elisseeff, 2003). We show the distributions of the local feature importance in Figure 6a.

The most important dimensions are longitude, latitude, and median income. We validate the local feature importance by applying it to variable selection. Using all dimensions, we get an MSE of around 0.34 (this is lower than in Table 1 as we do not use feature weighting). When deleting the three most important dimensions, the MSE increases to 0.99. However, keeping only the most important dimensions, the MSE slightly improves to 0.33. Therefore, we conclude that the feature importance has found the most important dimensions.

We now move on to inspect a failure case of DNNR. In Figure 6b, we show how the neighborhood of a poorly predicted point can be inspected. The prediction (red circle) is off by 4.85. From looking at the projection of the data to the longitude and latitude dimensions, we can see that the prediction is based on points (crosses) far away from the query. These points might have a similar number of bedrooms but differ in the location. As we found in the previous experiment, the latitude/longitude belong to the most important features. This inspection motivates the feature scaling once again, as DNNR Unscale weights all dimension equally, it selected the nearest neighbors using less relevant dimensions.

6. Conclusion, Limitations, and Future Work

Conclusion DNNR showed that local datapoint gradients carry valuable information for prediction and can be exploited using a simple Taylor expansion to provide a significant performance boost over KNN regression. In large-scale evaluations, DNNR achieved comparable results to state-of-the-art complex gradient boosting models. An advantage of DNNR's simplicity is that we can obtain error bounds by extending KNN's theory. Our theoretical analysis illustrates the benefits of using DNNR over KNN. DNNR strikes a good balance between performance and transparency and may therefore be the method of choice in problems with elevated requirements for the system's interpretability.

Limitations Our evaluation of DNNR points to a potential limitation on discrete data. When features or targets have the same value, the gradient is zero, and DNNR falls back to KNN's decision. On partially discrete datasets, DNNR always performs at least as well as KNN, e.g. the results of DNNR and KNN on the Yacht dataset (Table 1). DNNR also inherits some limitations from KNN, such that the L2-metric might not represent similarities optimally.

Future Work DNNR was designed for regression tasks but could also be adapted for classification, e.g. by fitting

the gradients of a logistic function as in (Mukherjee & Wu, 2006) or via label smoothing (Müller et al., 2019). An interesting future direction may be extending DNNR specifically to symbolic regression by utilizing the estimated gradient information. Future work could also explore the use of DNNR for data augmentation, where points could be sampled, and the label computed estimated with the local gradient. Another research direction would be to tighten the theoretical bounds or to study the effect of scaling from a theoretical perspective.

Acknowledgments

We thank Maximilian Granz, Oana-Iuliana Popescu, Benjamin Wild, Luis Herrmann, and David Dormagen for fruitful discussion and feedback on our manuscript. We are also grateful for our reviewers' feedback. LS was supported by the Elsa-Neumann Scholarship of the state of Berlin. We thank the HPC Service of ZEDAT, Freie Universität Berlin, for generous allocations of computation time (Bennett et al., 2020).

References

Aha, D. W. *Feature Weighting for Lazy Learning Algorithms*, pp. 13–32. Springer US, Boston, MA, 1998. ISBN 978-1-4615-5725-8. doi: 10.1007/978-1-4615-5725-8_2. URL https://doi.org/10.1007/978-1-4615-5725-8_2.

Arik, S. O. and Pfister, T. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16826.

Ausset, G., Clémen, S., et al. Nearest neighbour based estimates of gradients: Sharp nonasymptotic bounds and applications. In *International Conference on Artificial Intelligence and Statistics*, pp. 532–540. PMLR, 2021. URL http://proceedings.mlr.press/v130/ausset21a/ausset21a.pdf.

Balsubramani, A., Dasgupta, S., Freund, Y., and Moran, S. An adaptive nearest neighbor rule for classification. In *NeurIPS*, 2019. URL https://proceedings.neurips.cc/paper/2019/file/a6a767bbb2e3513233f942e0ff24272c-Paper.pdf.

Ben-Shabat, Y. and Gould, S. Deepfit: 3d surface fitting via neural network weighted least squares. In *Computer Vision – ECCV 2020*, pp. 20–34, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58452-8. URL https://link.springer.com/chapter/10.1007/978-3-030-58452-8_2.

Bennett, L., Melchers, B., and Proppe, B. Curta: A general-purpose high-performance computer at zedat, freie universität berlin. http://dx.doi.org/10.17169/refubium-26754, 2020.

Berahas, A., Cao, L., Choromanski, K., and Scheinberg, K. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Computational Mathematics*, 05 2021. doi: 10.1007/s10208-021-09513-z. URL https://doi.org/10.1007/s10208-021-09513-z.

Bhatia, N. and Vandana. Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security*, 8, 07 2010. URL https://doi.org/10.5120/16754-7073.

Bui, T., Hernandez-Lobato, D., Hernandez-Lobato, J., Li, Y., and Turner, R. Deep gaussian processes for regression using approximate expectation propagation. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1472–1481, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/bui16.html.

Cava, W. L., Orzechowski, P., Burlacu, B., de Franca, F. O., Virgolin, M., Jin, Y., Kommenda, M., and Moore, J. H. Contemporary symbolic regression methods and their relative performance. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=xVQMrDLyGst.

Chaudhuri, K. and Dasgupta, S. Rates of convergence for nearest neighbor classification. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/db957c626a8cd7a27231adfbf51e20eb-Paper.pdf.

Chen, G. H. and Shah, D. Explaining the success of nearest neighbor methods in prediction. *Foundations and Trends® in Machine Learning*, 10(5-6):337–588, 2018. ISSN 1935-8237. doi: 10.1561/2200000064. URL http://dx.doi.org/10.1561/2200000064.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. KDD '16, pp. 785–794,

New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

Cleary, J. G. and Trigg, L. E. K*: An instance-based learner using an entropic distance measure. In Prieditis, A. and Russell, S. (eds.), *Machine Learning Proceedings 1995*, pp. 108–114. Morgan Kaufmann, San Francisco (CA), 1995. ISBN 978-1-55860-377-6. doi: https://doi.org/10.1016/B978-1-55860-377-6.50022-0. URL https://www.sciencedirect.com/science/article/pii/B9781558603776500220.

Cover, T. and Hart, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964. URL https://doi.org/10.1109/TIT.1967.1053964.

De Brabanter, K., De Brabanter, J., De Moor, B., and Gijbels, I. Derivative estimation with local polynomial fitting. *J. Mach. Learn. Res.*, 14(1):281–301, jan 2013. ISSN 1532-4435. URL https://www.jmlr.org/papers/volume14/debrabanter13a-deleted/debrabanter13a.pdf.

Dorogush, A. V., Ershov, V., and Gulin, A. Catboost: gradient boosting with categorical features support. *ArXiv*, abs/1810.11363, 2018. URL https://arxiv.org/abs/1810.11363.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

European Commission. Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts, 2021. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206.

Fan, J. Design-adaptive nonparametric regression. *Journal of the American statistical Association*, 87(420):998–1004, 1992. URL https://doi.org/10.2307/2290637.

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*. Number 66 in Monographs on statistics and applied probability series. Chapman & Hall, 1996. ISBN 0412983214. URL http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+19282144X&sourceid=fbw_bibsonomy.

Friedman, J. H. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991. ISSN 00905364. URL http://www.jstor.org/stable/2241837.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003. URL https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf.

Hristache, M., Juditsky, A., Polzehl, J., and Spokoiny, V. Structure Adaptive Approach for Dimension Reduction. *The Annals of Statistics*, 29(6):1537 – 1566, 2001. doi: 10.1214/aos/1015345954. URL https://doi.org/10.1214/aos/1015345954.

Kelley Pace, R. and Barry, R. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997. ISSN 0167-7152. doi: https://doi.org/10.1016/S0167-7152(96)00140-X. URL https://www.sciencedirect.com/science/article/pii/S016771529600140X.

Kulkarni, S. and Posner, S. Rates of convergence of nearest neighbor estimation under arbitrary sampling. *IEEE Transactions on Information Theory*, 41(4):1028–1039, 1995. doi: 10.1109/18.391248. URL https://doi.org/10.1109/18.391248.

Molnar, C. *Interpretable Machine Learning*. 2019. URL https://christophm.github.io/interpretable-ml-book/.

Mukherjee, S. and Wu, Q. Estimation of gradients and coordinate covariation in classification. *Journal of Machine Learning Research*, 7:2481–2514, 12 2006. URL https://jmlr.org/papers/volume7/mukherjee06b/mukherjee06b.pdf.

Mukherjee, S. and Zhou, D.-X. Learning coordinate covariances via gradients. *Journal of Machine Learning Research*, 7(18):519–549, 2006. URL http://jmlr.org/papers/v7/mukherjee06a.html.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/f1748d6b0fd9d439f71450117eba2725-Paper.pdf.

Rasmussen, C. and Williams, C. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, January 2006. URL http://www.gaussianprocess.org/gpml/.

Turner, I. W., Belward, J. A., and Oqielat, M. N. Error bounds for least squares gradient estimates. *SIAM Journal on Scientific Computing*, 32:2146–2166, 2010. URL https://doi.org/10.1137/080744906.

Udrescu, S.-M. and Tegmark, M. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16), 2020. doi: 10.1126/sciadv.aay2631. URL https://www.science.org/doi/abs/10.1126/sciadv.aay2631.

Vivencio, D. P., Hruschka, E. R., do Carmo Nicoletti, M., dos Santos, E. B., and de O. Galvão, S. D. C. Feature-weighted k-nearest neighbor classifier. *2007 IEEE Symposium on Foundations of Computational Intelligence*, pp. 481–486, 2007. URL https://doi.org/10.1109/FOCI.2007.371516.

Wang, J., Neskovic, P., and Cooper, L. N. Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognition Letters*, 28(2):207–213, 2007. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2006.07.002. URL https://www.sciencedirect.com/science/article/pii/S0167865506001917.

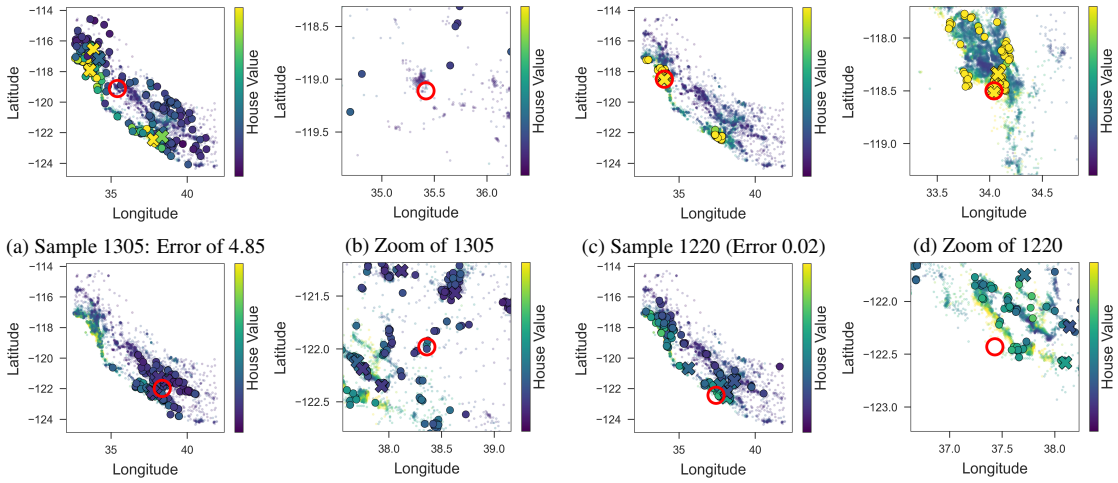
Wang, Q., Wan, J., and Yuan, Y. Deep metric learning for crowdedness regression. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2633–2643, 2018. doi: 10.1109/TCSVT.2017.2703920. URL https://doi.org/10.1109/TCSVT.2017.2703920.

Weinberger, K. Q. and Tesauro, G. Metric learning for kernel regression. In Meila, M. and Shen, X. (eds.), *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pp. 612–619, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR. URL https://proceedings.mlr.press/v2/weinberger07a.html.

Weinberger, K. Q., Blitzer, J., and Saul, L. Distance metric learning for large margin nearest neighbor classification. In Weiss, Y., Schölkopf, B., and Platt, J. (eds.), *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2006. URL https://proceedings.neurips.cc/paper/2005/file/a7f592cef8b130a6967a90617db5681b-Paper.pdf.

Wettschereck, D. and Dietterich, T. G. Locally adaptive nearest neighbor algorithms. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pp. 184–191, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. URL https://proceedings.neurips.cc/paper/1993/file/5f0f5e5f33945135b874349cfbed4fb9-Paper.pdf.

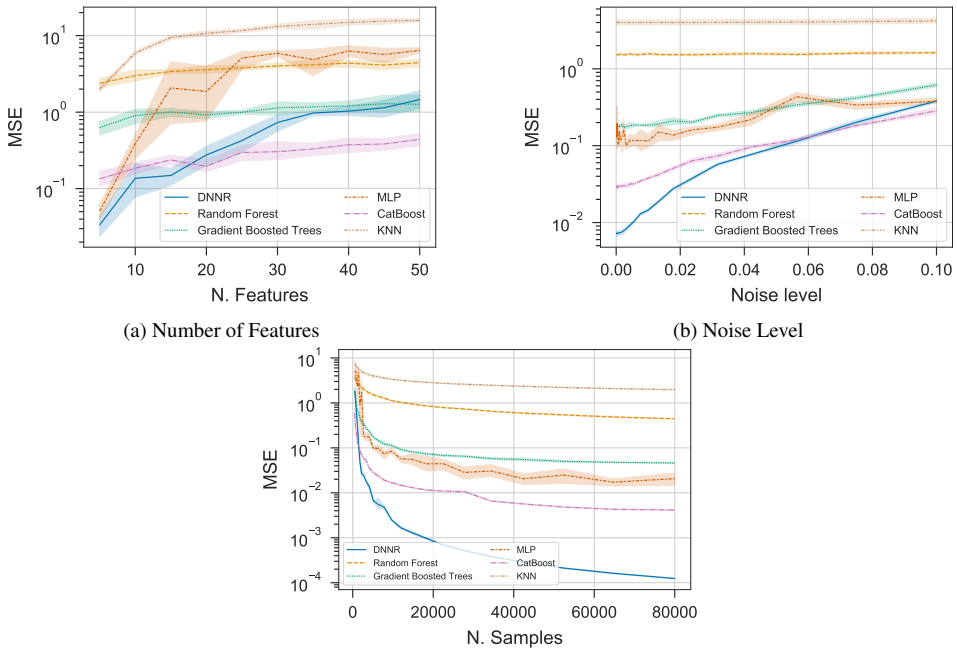
A. Figures



(a) Sample 1305: Error of 4.85 (b) Zoom of 1305 (c) Sample 1220 (Error 0.02) (d) Zoom of 1220

(e) Sample 1081 (Error 0.13) (f) Zoom of 1081 (g) Sample 393 (Error 1.01) (h) Zoom of 393

Figure 7. The selected neighbors for different queries on the California dataset. The red circle marks the query point. The prediction is done by approximating the gradient locally for each nearest neighbors (crosses). The filled circles visualize the points used for gradient approximation. The neighbors are further away for higher errors **(a) & (g)** while close to the query for lower errors **(c) & (e)**.



(a) Number of Features (b) Noise Level

(c) Number of Samples

Figure 8. The effect of different parameters of the dataset. The results are obtains on the Friedman-1 dataset. The confidence intervals denote the standard derivation over multiple folds.

B. Tables

Table 3. Mean percentage difference from the best performing model on each dataset.

	Protein	CO^2Emission	California	Yacht	Airfoil	Concrete	NOxEmission	Sarcos
CatBoost	1.54%	1.09%	0.00%	0.00%	0.00%	0.00%	12.91%	85.45%
Grad. B. Trees	4.42%	13.71%	7.94%	32.14%	39.32%	14.53%	20.26%	156.07%
KNN	15.03%	6.63%	108.47%	27065.48%	236.23%	334.12%	33.91%	147.46%
KNN-neigh	7.76%	8.90%	2.65%	953.97%	227.69%	187.56%	18.79%	150.00%
LGBM	9.51%	6.09%	3.17%	2623.41%	60.36%	15.05%	20.74%	127.82%
LL	20.93%	8.99%	72.49%	20072.22%	457.28%	267.28%	28.69%	11.86%
LL-neigh	10.83%	0.00%	12.70%	880.95%	179.35%	132.46%	12.35%	11.02%
MLP	23.22%	11.99%	84.13%	1623.02%	704.43%	151.99%	53.46%	80.37%
Random Forest	1.39%	1.54%	21.69%	348.81%	122.15%	62.53%	26.08%	219.77%
Tabnet	46.25%	10.63%	108.47%	1024.21%	689.79%	212.47%	0.00%	84.18%
XGBoost	3.17%	9.99%	6.35%	2.78%	28.72%	19.73%	15.54%	157.63%
DNNR-2 ord.	0.00%	12.35%	13.76%	88.89%	81.96%	102.52%	20.37%	2.68%
DNNR	5.79%	1.27%	0.00%	315.08%	123.58%	160.93%	2.86%	0.00%

C. Approximation Bounds

Notation	Meaning
x	An input vector
X	The random variable of the input
Y	The random variable of the output
k	The number of nearest neighbors
k'	The number of neighbors to estimate the gradient
$\eta(x)$	The expected target value: $\eta(x) = \mathbb{E}[Y\|X = x]$
$\nabla\eta(x)$	The gradient of $\eta(x)$ w.r.t. x
$\eta_{\text{KNN}}(x)$	Estimated regression function for KNN regression
$\eta_{\text{DNNR}}(x)$	Estimated regression function for DNNR
$\text{nn}(x, k)$	Returns the indices of the k nearest neighbors of x
$\hat{Y}_{m,x}$	Estimated regression value for x from point X_m
$\vartheta_{\max}$	Maximum Lipschitz constant
A	Matrix to estimate the gradient using OLS
σ_1	Smallest singular value of A
C^μ	Set of μ times differentiable continuous functions
$(v \cdot \nabla)^\mu f$	The μ-order directional derivative of f w.r.t v
$\mathbb{E}_n[.]$	Expectation over n training points $(X_i, Y_i)_{1\leq i\leq n}$
$\hat{\gamma}_m$	Locally estimated gradient for point m.
$\hat{\omega}_m$	Locally estimated gradient and higher order terms
ε	The error tolerance
δ	Probability tolerance
h_m	Distance between x and point X_m
$\rho(x, x')$	Distance metric

Table 4. Notation used in this work.

The proof of the approximation bounds of DANN extends the the proof of KNN approximation bounds given in (Chen & Shah, 2018, p. 68ff.) by a Taylor Approximation. For the approximation of the gradient, we will rely on results given in (Turner et al., 2010).

Notation Our notation follows the one given in (Chen & Shah, 2018):

C.1. General Properties

We list here some inequalities used in the later proof.

Jensen's Inequality Given a random variable X and a convex function $f(X)$ then:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]. \tag{17}$$

Hoeffding's Inequality Let $X_1, \ldots, X_k$ be independent random variables bounded between $[a, b]$. The empirical mean is given by: $\bar{X} = \frac{1}{k}(X_1 + \ldots + X_k)$. Then:

$$\mathbb{P}(\left|\bar{X} - \mathbb{E}[\bar{X}]\right| > t) \leq 2\exp\left(-\frac{2kt^2}{(b-a)^2}\right) \tag{18}$$

Chernoff Bound for Binomial distribution Let $X = \sum_{i=1}^{n} X_i$ be a sum of n independent binary random variable each $X_i = 1$ with probability p_i. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^{k} p_i = n\bar{p}$, where $\bar{p} = \frac{1}{n}\sum_{i=1}^{k} p_i$. Then

$$\mathbb{P}(X \leq (1-\delta)\mu) \leq \exp(-\mu\delta^2/2). \tag{19}$$

C.2. Gradient Approximation

Here, we first proof two lemmas for bounding the gradient. They generalize the proof in (Turner et al., 2010, section 3.1) from 2 to d-dimensions.

The first Lemma bounds terms of a Taylor series and is need for the proof of Lemma .

Lemma 2 *Let $f : D \subset \mathbb{R}^d \to \mathbb{R}$ be of class C^μ, $a \in D$, and $B(a) \subset D$ some neighborhood of a. Suppose that for $i = 0, 1, \ldots, n$ we have $\frac{\partial^\mu f}{\partial^\mu a} \in Lip_{\vartheta_i}(B(a))$ and $\vartheta_{\max} = \max_{i\in 1,\ldots\mu} \vartheta_i$. Then for any $a + h\nu \in B(a)$ with $||\nu|| = 1$*

$$\left|\sum_{k=1}^{\mu} \frac{h^{k-1}}{k!}(\nu\cdot\nabla)^k f(a) - \frac{f(a+h\nu) - f(a)}{h}\right| \leq \frac{h^\mu}{(\mu+1)!}\vartheta_{\max}||v||_1^\mu. \tag{20}$$

Proof Lemma 2. The proof starts with rearranging the Taylor Series which is given here:

$$f(a+h\nu) = f(a) + h\frac{(\nu\cdot\nabla)f(a)}{1!} + \ldots + h^\mu\frac{(\nu\cdot\nabla)^\mu f(a)}{\mu!} + R_\mu. \tag{21}$$

The remainder R_μ has the following form:

$$R_\mu = \frac{h^{\mu+1}}{\mu!}\int_0^1 (1-t)^\mu(\nu\cdot\nabla)^{\mu+1} f(a+th\nu)dt. \tag{22}$$

By dividing by h and rearanging some terms, we have:

$$\frac{f(a+h\nu) - f(a)}{h} - \sum_{k=1}^{\mu-1}\frac{h^{k-1}}{k!}(\nu\cdot\nabla)^k f(a) = \frac{R_{\mu-1}}{h}. \tag{23}$$

Now, we add the same quantity to both sides of the previous equation, using $\int_0^1 (1-t)^{\mu-1}dt = \frac{1}{\mu}$:

$$\begin{aligned}\frac{h^{\mu-1}}{\mu!}(\nu\cdot\nabla)^\mu f(a) - \frac{f(a+h\nu) - f(a)}{h} + \sum_{k=1}^{\mu-1}\frac{h^{k-1}}{k!}(\nu\cdot\nabla)^k f(a)\\ = \frac{h^{\mu-1}}{(\mu-1)!}\int_0^1 (1-t)^{\mu-1}\Big\{(\nu\cdot\nabla)^\mu f(a) - (\nu\cdot\nabla)^\mu f(a+th\nu)\Big\}dt.\end{aligned}$$

Therefore, we have:

$$
\begin{aligned}
&\left|\sum_{k=1}^{\mu-1}\frac{h^{k-1}}{k!}(\nu\cdot\nabla)^k f(a)-\frac{f(a+h\nu)-f(a)}{h}\right| \\
&=\left|\frac{h^{\mu-1}}{(\mu-1)!}\int_0^1(1-t)^{\mu-1}\Big\{(\nu\cdot\nabla)^\mu(f(a)-f(a+th\nu))\Big\}dt\right| \\
&=\left|\frac{h^{\mu-1}}{(\mu-1)!}\int_0^1(1-t)^{\mu-1}\Big\{(\nu\cdot\nabla)^\mu(f(a)-f(a+th\nu))\Big\}dt\right| \\
&\leq\frac{h^{\mu-1}}{(\mu-1)!}\int_0^1(1-t)^{\mu-1}\Big|\left(\nu\cdot\nabla\right)^\mu\left(f(a)-f(a+th\nu)\right)\Big|dt \\
&\leq\frac{h^{\mu+1}}{(\mu+1)!}\vartheta_{\max}\left|\nu\right|_1^\mu.
\end{aligned}
$$

In last step, we rewrote the μ-th directional derivative using partial derivative and bounded it using the Lipschitz-continuity as follows:

$$
\begin{aligned}
&|(\nu\cdot\nabla)^\mu(f(a)-f(a+th\nu))| \\
&=\left|\sum_{k_1+\ldots+k_d=\mu}\binom{\mu}{k_1,\ldots,k_d}\left(\prod_{i=1}^d\nu_i^{k_i}\right)\frac{\partial^\mu(f(a)-f(a+th\nu))}{\partial a_1^{k_1}\cdot\ldots\cdot\partial a_d^{k_d}}\right| \\
&=\sum_{k_1+\ldots+k_d=\mu}\binom{\mu}{k_1,\ldots,k_d}\left(\prod_{i=1}^d\left|\nu_i^{k_i}\right|\right)\frac{\partial^\mu\left|(f(a)-f(a+th\nu))\right|}{\partial a_1^{k_1}\cdot\ldots\cdot\partial a_d^{k_d}} \\
&\leq\sum_{k_1+\ldots+k_d=\mu}\binom{\mu}{k_1,\ldots,k_d}\left(\prod_{i=1}^d\left|\nu_i^{k_i}\right|\right)\vartheta_{\max}|a-a+th\nu|\quad\text{(Lipschitz continuity of order }\mu) \\
&=\vartheta_{\max}|th\nu|\left(|\nu_1|+\ldots+|\nu_d|\right)^\mu\qquad\text{(Multinomial theorem)} \\
&=\vartheta_{\max}|th|\left|\nu\right|_1^\mu.
\end{aligned}
$$

This finishes the proof of Lemma 2. □

Lemma 3 *Let $f:D\subset\mathbb{R}^d\to\mathbb{R}$ be of class C^μ, $a\in D$, and $B(a)\subset D$ be some neighborhood of a. Suppose around point a we have m neighboring points v_k, $k=1,\ldots,m$ with a, $v_1,\ldots,v_m\in B(a)\subset D$. Suppose further that $\frac{\partial^\mu f}{\partial a_1^{l_1}\ldots\partial a_d^{l_d}}\in Lip_{\vartheta_{\max}}(B(a))$ for $l_1+\ldots+l_d=\mu$, and we approximate the gradient locally at a by $E_1\hat{\omega}$ via the least-squares solution $\hat{\omega}=\arg\min_{\omega\in\mathbb{R}^d}||A\omega-q||$, where*

$$
A=\begin{pmatrix}\nu_1^T & \nu_{1*}^T\\ \nu_2^T & \nu_{2*}^T\\ \vdots & \\ \nu_m^T & \nu_{m*}^T\end{pmatrix}\in\mathbb{R}^{m\times d},\qquad q=\begin{pmatrix}\frac{f(a+h_1\nu_1)-f(a)}{h_1}\\ \frac{f(a+h_2\nu_2)-f(a)}{h_1}\\ \vdots\\ \frac{f(a+h_m\nu_m)-f(a)}{h_m}\end{pmatrix}\in\mathbb{R}^{mx1},\tag{24}
$$

$E_1=\left(I_d0\right)\in\mathbb{R}^{d\times p}$, $h_k=||v_k-a||$ with $h_k\nu_k=v_k-a$; $\nu_k^T=(\nu_{1_k},\ldots,\nu_{d_k})$,

$\nu_{k*}^T=\left(\frac{h_k^{\mu'-1}}{\mu'!}\binom{\mu'}{l_1,\ldots,l_d}\prod_{i_1}^d\nu_i^{k_i}\right)_{2\leq\mu'\leq\mu,l_1+\ldots l_d=\mu'}$

and $p=\sum_{i=1}^\mu\frac{(d+\mu)!}{d!}$. Then a bound on the error in the least-squares gradient estimate is given by:

$$
||\nabla f(a)-E_1\hat{\omega}||\leq\frac{\vartheta_{\max}h_{\max}^\mu}{\sigma_1(\mu+1)!}\sqrt{\sum_{i=1}^m||\nu_i||_1^{2\mu}},\tag{25}
$$

where σ_1 is the smallest singular value of A, which is assumed to have rank$(A) = p$, $\vartheta_{\max}$ is as defined in Lemma 2, and $h_{\max} = \max_{1 \leq k \leq m} h_k$.

Proof. Let $E_2 \in \mathbb{R}^{(p-d) \times p}$ be the last $p - d$ rows of the identity matrix I_p and

$$U = \left(\frac{\partial f}{\partial a_1}(a), \ldots, \frac{\partial f}{\partial a_d}(a), \frac{\partial^2 f}{\partial a_1^{l_{2_1}} \ldots \partial a_d^{l_{2_d}}}, \ldots, \frac{\partial^2 f}{\partial a_d^{l_{2_d}}}, \ldots, \frac{\partial^\mu f}{\partial a_1^{l_{n_1}}}, \ldots, \frac{\partial^\mu f}{\partial a_d^{l_{n_d}}} \right)^T \in \mathbb{R}^{p \times 1}. \tag{26}$$

Now $U - \hat{\omega}$ can be partitioned as $\begin{pmatrix} E_1(U - \hat{\omega}) \\ E_2(U - \hat{\omega}) \end{pmatrix}$, and hence:

$$||U - \hat{\omega}||^2 = ||E_1(U - \hat{\omega})||^2 + ||E_2(U - \hat{\omega})||^2 \geq ||E_1(U - \hat{\omega})||^2 = ||\nabla f(a) - E_1 \hat{\omega}||^2. \tag{27}$$

Next, with $\hat{\omega} = A^\dagger q$ and $||A^\dagger|| = \frac{1}{\sigma_1^2}$, we have

$$||U - \hat{\omega}||^2 = ||U - A^\dagger q||^2 = ||A^\dagger (AU - q)||^2 \leq ||A^\dagger||^2 ||AU - q||^2 = \frac{1}{\sigma_1^2} ||AU - q||^2. \tag{28}$$

Now using the result in Lemma 2, the following upper bound can be derived:

$$||AU - q||^2 = \sum_{i=1}^{m} \left|\left| \sum_{k=1}^{p} A_{i,k} U_k - q_k \right|\right|^2 \tag{29}$$

$$= \sum_{i=1}^{m} \left|\left| \sum_{k=1}^{p} A_{i,k} U_k - q_k \right|\right|^2 \tag{30}$$

$$= \sum_{i=1}^{m} \left|\left| \sum_{k=1}^{\mu} \frac{h_i^{k-1}}{k!} (\nu_i \cdot \nabla)^k f(a) - \frac{f(a + h\nu) - f(a)}{h} \right|\right|^2 \tag{31}$$

$$\leq \sum_{i=1}^{m} \left(\frac{h_i^\mu}{(\mu+1)!} \vartheta_{\max} ||\nu_i||_1^\mu \right)^2 \qquad \text{(using Lemma 2)} \tag{32}$$

$$= \left(\frac{\vartheta_{\max}}{(\mu+1)!} \right)^2 \sum_{i=1}^{m} (h_i^\mu ||\nu_i||_1^\mu)^2 \tag{33}$$

$$\leq \left(\frac{\vartheta_{\max}}{(\mu+1)!} \right)^2 (h_{\max}^\mu)^2 \sum_{i=1}^{m} ||\nu_i||_1^{2\mu} \tag{34}$$

The result follows from

$$||\nabla f(a) - E_1 \hat{\omega}|| \leq ||U - \hat{\omega}|| \leq \frac{\vartheta_{\max} h_{\max}^\mu}{\sigma_1 (\mu+1)!} \sqrt{\sum_{i=1}^{m} ||\nu_i||_1^{2\mu}}. \quad \square \tag{35}$$

C.3. DaNN pointwise error

The proof follows the one from (Chen & Shah, 2018). In (Chen & Shah, 2018), also a method to break ties, e.g. points with the same distance is proposed – a common problem when the data is discrete and not continuous. We will not discuss how to break ties, as the gradient estimation assumes continuous dimensions where ties should a.s. never happen.

Technical Assumptions ($A^{\text{technical}}_{\mathcal{X},\rho,\mathbb{P}_X}$):

- The feature space $\mathcal{X}$ and distance ρ form a separable metric space.
- The feature distribution $\mathbb{P}_X$ is a Borel probability measure.

Assumptions Besicovitch($A_\eta^{\text{Besicovitch}}$): The regression function η satisfied the Besicovitch condition if $\lim_{r\downarrow 0}\mathbb{E}[Y|X\in B_{x,r}]=\eta(x)$ for x almost everywhere w.r.t. $\mathbb{P}_X$..

Assumption Lipschitz ($A_\eta^{\text{Lipschitz}(\vartheta_{\max})}$): The regression function η is Lipschitz continuous with parameter $\vartheta_{\max}$ if $|\eta(x)-\eta(x')|\leq\vartheta_{\max}\rho(x,x')$ for all $x,x'\in\mathcal{X}$.

Lemma 4 *Under assumptions $A_{\mathcal{X},\rho,\mathbb{P}_X}^{\text{technical}}$, $A_\eta^{\text{Besicovitch}}$, let $x\in supp(\mathbb{P}_X)$ be a feature vector, and $\eta(x)=\mathbb{E}[Y|X=x]\in\mathbb{R}$, be the expected label value for x. Let $\varepsilon>0$ be an error tolerance in estimating expected label $\eta(x)$, and $\delta\in(0,1)$ be a probability tolerance. Suppose that $Y\in[y_{\min},y_{\max}]$ for some constants $y_{\min}$ and $y_{\max}$. Let $\xi\in(0,1)$. Then there exists a threshold distance $h^*\in(0,\inf)$ such that for any smaller distance $h\in(0,h^*)$ and with the number of nearest neighbors satisfying $k\leq(1-\xi)n\mathbb{P}_X(B_{x,h})$, then with probability at least*

$$1-2\exp\left(-\frac{k\varepsilon^2}{2(y_{\max}-y_{\min})^2}\right)-\exp\left(-\frac{\xi^2 n\mathbb{P}_X(B_{x,r})}{2}\right). \tag{36}$$

we have

$$|\hat{\eta}_{DNNR}(x)-\eta(x)|\leq\varepsilon. \tag{37}$$

Furthermore, if the function η satisfies assumptions $A_\eta^{\text{Lipschitz}(\vartheta_{\max})}$, then we can take

$$h_{DNNR}^*=\sqrt{\frac{\varepsilon}{\vartheta_{\max}(1+\tau)}}, \tag{38}$$

where $\tau=\mathbb{E}\left[\frac{\sqrt{\sum_{i=1}^m||\nu_i||_1^{2\mu}}}{\sigma_1}\;\middle|\;X\in B_{x,h}\right]$. ν, σ_1, and $\vartheta_{\max}$ are defined as in Lemma 1.

Proof of Lemma 4: Fix $x\in\text{supp}(\mathbb{P}_X)$. Let $\varepsilon>0$. We upper-bound the error $|\hat{\eta}-\eta(x)|$ with the triangle inequality:

$$\begin{aligned}|\hat{\eta}(x)-\eta(x)|&=|\hat{\eta}(x)-\mathbb{E}_n[\hat{\eta}(x)]+\mathbb{E}_n[\hat{\eta}(x)]-\eta(x)|\\&\leq\underbrace{|\hat{\eta}(x)-\mathbb{E}_n[\hat{\eta}(x)]|}_{①}+\underbrace{|\mathbb{E}_n[\hat{\eta}(x)]-\eta(x)|}_{②}\end{aligned}$$

The proof now continues by showing that both ① and ② are below $\varepsilon/2$ with high probability. The proof is adapted from the KNN pointwise regression proof in (Chen & Shah, 2018, p. 68ff.). The part ① is almost identical to the proof in (Chen & Shah, 2018, p. 68ff.). For part ②, we will use the Taylor Approximation and then bound the gradient using the results from Lemma 1.

Part ① $|\hat{\eta}(x)-\mathbb{E}_n[\hat{\eta}(x)]|\leq\frac{\varepsilon}{2}$:

Lemma 5 *Under same assumption of Theorem 1, we have:*

$$\mathbb{P}\left(|\hat{\eta}(x)-\mathbb{E}_n[\hat{\eta}(x)]|\geq\frac{\varepsilon}{2}\right)\leq 2\exp\left(-\frac{k\varepsilon^2}{2(y_{\max}-y_{\min})^2}\right). \tag{39}$$

Proof Lemma 5: As we want to apply the Hoeffding's inequality, we have to show that the $\hat{Y}$ are independent

Probability Model: The randomness of the $\hat{\eta}_{\text{DNNR}}(x)$ can be described as:

1. Sample a feature vector $\tilde{X}\in\mathcal{X}$ from the marginal distribution of the $(k+1)$-st nearest neighbor of x, let $h_{k+1}=\rho(x,\tilde{X})$ denote the distance between x and $\tilde{X}$.
2. Sample k feature vectors i.i.d. from $\mathbb{P}_X$ conditioned on landing in the ball $B^o_{x,\rho(x,\tilde{X})}$,
3. Sample $n-k-1$ feature vectors i.i.d. from $\mathbb{P}_X$ conditoned on landing in $\mathcal{X}\setminus B^o_{x,h_{k+1}}$,
4. Randomly permute the n feature vectors sampled,

5. For each feature vector X_i generated, sample its label Y_i based on the conditional distribution $\mathbb{P}_{Y|X=X_i}$.

Therefore, we can write the expectation over the n training points as:

$$\mathbb{E}_n[\hat{\eta}(x)] = \mathbb{E}_{h_{k+1}(x)}[\mathbb{E}[\hat{Y}|X \in B^o_{x,h_{k+1}}], \tag{40}$$

where $\hat{Y} = Y + \nabla Y(X - x)$ denotes the prediction for point x from a data point X with label Y and gradient ∇Y.

The points samples in step 2 are precisely the k nearest neighbor of x, and their $\hat{Y}$ values i.i.d. with expectation $\mathbb{E}_n[\hat{\eta}(x)] = \mathbb{E}_{\tilde{X}}[\mathbb{E}[\hat{Y}|X \in B^o_{x,h_{k+1}}]$. As they are bounded between $y_{\min}$ and $y_{\max}$ (this can be enforced by clipping), Hoeffding's inequality yields:

$$\mathbb{P}\left(|\hat{\eta}(x) - \mathbb{E}_n[\hat{\eta}(x)]| \geq \frac{\varepsilon}{2}\right) \leq 2\exp\left(-\frac{k\varepsilon^2}{2(y_{\max} - y_{\min})^2}\right) \tag{41}$$

This finishes the proof of Lemma 5. □

Part ② $|\mathbb{E}_n[\hat{\eta}(x)] - \eta(x)| \leq \frac{\varepsilon}{2}$:

As discussed above (40) the expectation of $\hat{\eta}(x)$ is

$$\mathbb{E}_n[\hat{\eta}(x)] = \mathbb{E}_{h_{k+1}(x)}[E[Y|X \in B^o_{x,h_{k+1}}] \tag{42}$$

Suppose that we could show that there exists home $h > 0$ such that

$$|\mathbb{E}[Y|X \in B^o_{x,r}] - \eta(x)| \leq \frac{\varepsilon}{2} \qquad \text{for all } r \in (0, h]. \tag{43}$$

Then provided that $h_{k+1} \leq h$:

$$\begin{aligned}
|\mathbb{E}_n[\hat{\eta}(x) - \eta(x)]| &= |\mathbb{E}_{h_{k+1}(x)}[E[Y|X \in B^o_{x,h_{k+1}}] - \eta(x)]| && (44)\\
&\leq \mathbb{E}_{h_{k+1}(x)}[|E[Y|X \in B^o_{x,h_{k+1}}] - \eta(x)|] && \text{(Jensen's ineq.)} \quad (45)\\
&\leq \frac{\varepsilon}{2} && \text{(inequality (43))} \quad (46)
\end{aligned}$$

Before establishing the existence of h, we first show that for any distance $r > 0$, with high probability we can ensure that $h_{k+1} \leq r$. Thus, once we show that h exists, we also know that we can ensure that $h_{k+1} \leq h$ with high probability.

Lemma 6 . *Let $r > 0$ and $\xi \in (0, 1)$. For positive integer $k \leq (1 - \xi)n\mathbb{P}_X(B_{x,r})$,*

$$\mathbb{P}_X(h_{k+1}(x) \geq r) \leq \exp\left(-\frac{\xi^2 n\mathbb{P}_X(B_{x,r}}{2}\right). \tag{47}$$

Thus, we have $h_{k+1}(x) \leq r$ with probability at least $1 - \exp\left(-0.5\xi^2 n\mathbb{P}_X(B_{x,r})\right)$. *Proof of Lemma 6.* Fix $r > 0$ and $\xi \in (0, 1)$. Let $N_{x,r}$ be the number of training points that land in the closed ball $B_{x,r}$. Note that $N_{x,r} \sim \text{Binomial}(n, \mathbb{P}_X(B_{x,r}))$. Then by a Chernoff bound for the binomial distribution, for any integer $k \leq (1 - \xi)n\mathbb{P}_X(B_{x,r})$, we have

$$\begin{aligned}
\mathbb{P}(N_{x,r} \leq k) &\leq \exp\left(-\frac{(n\mathbb{P}_X(B_{x,r} - k)^2}{2n\mathbb{P}_X(B_{x,r})}\right) && (48)\\
&\leq \exp\left(-\frac{(n\mathbb{P}_X(B_{x,r} - (1-\xi)n\mathbb{P}_X(B_{x,r}))^2}{2n\mathbb{P}_X(B_{x,r})}\right) && (49)\\
&= \exp\left(-\frac{\xi^2 n\mathbb{P}_X(B_{x,r}))}{2}\right). && (50)
\end{aligned}$$

If $N_{x,r} \leq k$, then also $h_{k+1}(x) \geq r$. Therefore, the event $\{h_{k+1}(x) \geq r\} \subset \{N_{x,r} \leq k\}$ and we have:

$$\mathbb{P}_X(h_{k+1}(x) \geq r) \leq \mathbb{P}(N_{x,r} \leq k) \leq \exp\left(-\frac{\xi^2 n\mathbb{P}_X(B_{x,r}))}{2}\right). \quad \square \tag{51}$$

Now, we show which distance h ensures that inequality (43) holds. When we only know that

$$\lim_{r \downarrow 0} \mathbb{E}[Y|X \in B^o_{x,r}] = \eta(x), \tag{52}$$

then the definition of a limit implies that there exists $h^* > 0$ (that depends on x and ε) such that

$$|\mathbb{E}[Y|X \in B^o_{x,h}] - \eta(x)| \leq \frac{\varepsilon}{2} \quad \text{for all } h \in (0, h^*), \tag{53}$$

i.e., inequality (43) holds, and so we have $|\mathbb{E}_n[\hat{\eta}(x)] - \eta(x)| \leq \frac{\varepsilon}{2}$ as shown earlier in inequality (46).

In the following derivation, we will assume η to be Lipschitz continuous with parameters $\vartheta_{\max}$. Further, we move $\eta(x)$ inside the expectation and use it's first term of Taylor series with X as root point: $\eta(\boldsymbol{x}) = \eta(X) + \eta'(X)(\boldsymbol{x} - X) + o(|\boldsymbol{x} - X|)$, where $o(|\boldsymbol{x} - X|)$ bounds the higher-order terms.

$$\begin{aligned}
&\left|\mathbb{E}\left[\hat{\eta}(\boldsymbol{x}) \mid X \in B_{x,h}\right] - \eta(\boldsymbol{x})\right| = \\
&\quad = \left|\mathbb{E}\left[Y + \nabla Y(\boldsymbol{x} - X) \mid X \in B_{x,h}\right] - \eta(\boldsymbol{x})\right| \\
&\quad = \left|\mathbb{E}\left[Y + \nabla Y(\boldsymbol{x} - X) - \eta(X) - \eta'(X)(\boldsymbol{x} - X) + o(|x - X|) \mid X \in B_{x,h}\right]\right| \\
&\quad = \left|\mathbb{E}\left[Y - \eta(X) + (\nabla Y - \eta'(X))\,(\boldsymbol{x} - X) + o(|x - X|) \mid X \in B_{x,h}\right]\right|
\end{aligned}$$

Now, we know that $\mathbb{E}[|Y - \eta(X)| \mid X \in B_{x,h}] = 0$, as the noise term has zero mean.

$$\begin{aligned}
&= \left|\mathbb{E}\left[(\nabla Y - \eta'(X))\,(\boldsymbol{x} - X) + o(|x - X|) \mid X \in B_{x,h}\right]\right| \\
&\leq \mathbb{E}\left[\left|(\nabla Y - \eta'(X))\,(\boldsymbol{x} - X)\right| + \left|o(|x - X|)\right| \;\middle|\; X \in B_{x,h}\right]
\end{aligned}$$

We can bound $|x - X| < h_{\max}$ and $|\nabla Y - \eta'(X)|$ by using the results from Lemma 2. The higher order terms $o(|x - X|)$ can be bound by the remainder of the Talyor series: $|R_\mu| \leq \frac{\vartheta_\mu |X - x|^\mu}{(\mu+1)!}$, where $\vartheta_\mu \leq \vartheta_{\max}$.

$$\mathbb{E}\left[\left|(\nabla Y - \eta'(X))\,(\boldsymbol{x} - X)\right| + \left|o(|x - X|)\right| \;\middle|\; X \in B_{x,h}\right] \tag{54}$$

$$\leq \mathbb{E}\left[\frac{\vartheta_{\max} h^{\mu+1}_{\max}}{\sigma_1(\mu+1)!}\sqrt{\sum_{i=1}^{m} ||\nu_i||_1^{2\mu}} + \vartheta_{\max}\frac{h^2_{\max}}{2} \;\middle|\; X \in B_{x,h}\right] \tag{55}$$

$$\leq \frac{\vartheta_{\max} h^{\mu+1}_{\max}}{(\mu+1)!}\,\mathbb{E}\left[\frac{\sqrt{\sum_{i=1}^{m} ||\nu_i||_1^{2\mu}}}{\sigma_1} \;\middle|\; X \in B_{x,h}\right] + \vartheta_{\max}\frac{h^2_{\max}}{2} \tag{56}$$

$$\leq \frac{\tau\vartheta_{\max} h^{\mu+1}_{\max}}{(\mu+1)!} + \vartheta_{\max}\frac{h^2_{\max}}{2} \leq \frac{\varepsilon}{2} \tag{57}$$

We used in the last step $\tau = \mathbb{E}\left[\frac{\sqrt{\sum_{i=1}^{m} ||\nu_i||_1^{2\mu}}}{\sigma_1} \;\middle|\; X \in B_{x,h}\right]$. As this proof only concerns first-order approximations, we use $\mu = 1$:

$$\frac{\tau\vartheta_{\max} h^2_{\max}}{2} + \vartheta_{\max}\frac{h^2_{\max}}{2} \leq \frac{\varepsilon}{2} \tag{58}$$

$$\Leftrightarrow h_{\max} \leq \sqrt{\frac{\varepsilon}{\vartheta_{\max}(1+\tau)}} \tag{59}$$

This finishes the proof of Lemma 4 □.

Theorem 1 follows from selecting $\xi = \frac{1}{2}$ and observing that:

$$n \geq \frac{8}{\mathbb{P}_X(B_{x,h})} \log \frac{2}{\delta} \Rightarrow \exp\left(-\frac{\xi^2 n \mathbb{P}_X(B_{x,h})}{2}\right) \leq \frac{\delta}{2}, \tag{60}$$

$$k \geq \frac{2(y_{\max} - y_{\min})^2}{\varepsilon^2} \log \frac{4}{\delta} \Rightarrow \exp\left(-\frac{k\varepsilon^2}{2(y_{\max} - y_{\min})^2}\right) \leq \frac{\delta}{2}. \tag{61}$$

D. Hyperparameters

Table 5. Number of Hyperparameters tuned for each model on the benchmark datasets. For DNNR, KNN and MLP, we take small datasets to be $n < 2000$, and medium datasets to be $n < 50000$. The same applies for PMLB and Feynman. We were unable to tune TabNet model due to computation constraints and used their Tabnet-L configuration for larger datasets (Sarcos, Protein, CO^2 and NOx Emissions) and Tabnet-S (n_d and n_a = 16) for smaller datasets.

	Airfoil	CO^2Emission	California	Concrete	NOxEmission	Protein	Yacht
DNNR	150	40	40	150	40	40	150
LL	50	50	50	50	50	50	50
Random Forest	12	12	12	12	12	12	12
Grad. B. Trees	48	48	48	48	48	48	48
MLP	594	72	72	594	72	72	594
CatBoost	48	48	48	48	48	48	48
XGBoost	48	48	48	48	48	48	48
LGBM	48	48	48	48	48	48	48
KNN	384	64	64	384	64	64	384
Tabnet	1	1	1	1	1	1	1

Table 6. XGBoost Hyperparameters

learning_rate	max_depth	n_estimators
[0.001,0.01,0.1,0.3]	[3,5,10]	[50,100,500,1000]

Table 7. LightGBM Hyperparameters

learning_rate	max_depth	n_estimators
[0.001,0.01,0.1,0.3]	[3,5,10]	[50,100,500,1000]

Table 8. CatBoost Hyperparameters

verbose	learning_rate	max_depth	n_estimators
[False]	[0.001,0.01,0.1,0.3]	[3,5,10]	[50,100,500,1000]

Table 9. Gradient Boosting Hyperparameters

learning_rate	max_depth	n_estimators
[0.001,0.01,0.1,0.3]	[3,5,10]	[50,100,500,1000]

Table 10. Random Forests Hyper Parameters

criterion	n_estimators	max_features
[mse]	[50,100,500,1000]	[auto,sqrt,log2]

Table 11. MLP Hyperparameters for small datasets

hidden_layer_sizes
[(25,), (50,), (100,), (250,), (25, 25), (50, 50), (100, 100), (250, 250), (25, 25, 25), (50, 50, 50), (100, 100, 100)

alpha	batch_size	learning_rate	learning_rate_init	early_stopping
[0,0.01,1]	[64,128]	[constant,invscaling,adaptive]	[0.001,0.01,0.1]	[True]

Table 12. MLP Hyperparameters for medium datasets

hidden_layer_sizes	alpha
[(128,),(128, 128),(128, 128, 128)]	[0,0.01]

batch_size	learning_rate	learning_rate_init	early_stopping
[512]	[constant,invscaling,adaptive]	[0.0001,0.001,0.01,0.1]	[True]

Table 13. MLP Hyperparameters for large datasets

hidden_layer_sizes	alpha
[(128,),(128, 128),(128, 128, 128)]	[0]

batch_size	learning_rate	learning_rate_init	early_stopping
[512]	[constant,adaptive]	[0.0001,0.001,0.01]	[True]

Table 14. KNN Hyperparameters for small datasets

n_neighbors	weights
[2,5,7,10,20,30,40,50]	[uniform,distance]

algorithm	leaf_size	p
[ball_tree,kd_tree,brute]	[10,30,50,100]	[1,2]

Table 15. KNN Hyperparameters for medium datasets

n_neighbors	weights	leaf_size	p
[2,5,7,10,25,50,100,250]	[distance,uniform]	[10,30,50,100]	[2]

Table 16. KNN Hyperparameters for large datasets

n_neighbors	weights	leaf_size	p
[2,3,5,7,10,12,15,20,25]	[distance]	[10,30,50,100]	[2]

Table 17. Tabnet Hyperparameters for large datasets

n_d	verbose	n_a	lambda_sparse	batch_size	virtual_batch_size
[128]	[False]	[128]	[0.0001]	[4096]	[128]

momentum	n_steps	gamma	optimizer_params	scheduler_params	max_epochs	patience
[0.8]	[5]	[1.5]	[{'lr': 0.02}]	[{'step_size': 8000, 'gamma': 0.9}]	[3000]	[100]

Table 18. Tabnet Hyperparameters for small datasets

n_d	n_a	verbose	lambda_sparse	batch_size	virtual_batch_size
[8,16]	[8,16]	[False]	[0.0001]	[64,128]	[8,16]

momentum	n_steps	gamma	optimizer_params	scheduler_params	max_epochs	patience
[0.02]	[3]	[1.3]	[{'lr': 0.02}]	[{'step_size': 10, 'gamma': 0.95}]	[3000]	[100]

Table 19. DNNR Hyperparameters for small datasets, $n\_neighbohrs$ corresponds to the k. For the k' (number of neighbors used in approximating the gradient) we use n values sampled between $lower\_bound \times d$ and $lower\_bound \times d$

n_neighbhors	upper_bound	lower_bound	n
[1, 2, 3, 5, 7]	[15]	[2]	[30]

Table 20. DNNR Hyperparameters for medium datasets , $n\_neighbohrs$ corresponds to the k. For the k' (number of neighbors used in approximating the gradient) we use n values sampled between $lower\_bound \times d$ and $lower\_bound \times d$

n_neighbhors	upper_bound	lower_bound	n
[3,4]	[18]	[2]	[20]

Table 21. DNNR Hyperparameters for large datasets , $n\_neighbohrs$ corresponds to the k. For the k' (number of neighbors used in approximating the gradient) we use n values sampled between $lower\_bound \times d$ and $lower\_bound \times d$

n_neighbhors	upper_bound	lower_bound	n
[3]	[12]	[2]	[14]

Table 22. LL Hyperparameters $n\_neighbohrs$ corresponds to the k. For the size of the neighborhood we use n values sampled between $lower\_bound \times d$ and $lower\_bound \times d$

upper_bound	lower_bound	n
[25]	[2]	[50]

8 Conclusion

8.1 Discussion

In the introduction, the challenges of explainability were outlined, including simplifications and intransparent assumptions made by explanation methods, as well as the difficulty in evaluating these methods.

In (Sixt et al. 2020; Sixt et al. 2022a), the theoretical limitations of modified backpropagation methods were examined. Our findings revealed that popular methods failed the weight-randomization sanity check, and that the theoretical foundation of these methods is incomplete. Specifically, the Deep Taylor Decomposition (DTD) (Sixt et al. 2022a) was found to not satisfy the assumption of the Taylor Theorem, and to be either under constraint or reduced to the gradient x input method. Given the application of explanation methods in validating model decisions and gaining insight into model behavior, it is crucial to validate them theoretically and empirically.

Besides theoretical limitations, it is also important to evaluate if explanations are useful to users. For this purpose, a study design using the bias-detection task was created in (Sixt et al. 2022b). In the user study, it was found that a powerful generative method had little to no benefit for users compared to a simple baseline.

I also contributed to the development of new methods for explainability. In (Schulz et al. 2020), a method was developed that provides absolute importance scores for each input feature in bits per pixel. This absolute frame of reference allows us to compare the importance across different models and datasets. Moreoever, we provided a thorough technical evaluation.

In (Nader et al. 2022), we extended the classical k-nearest neighbor regression using higher-order information. We showed that this method outperforms the classical kNN regression and performs similarly to the state-of-the-art tree-based methods.

In general, I would conclude that explainability is a complex problem and is likely to remain so for the foreseeable future. Heuristic assumptions, as well

as the complexity of the models, and the difficulty of evaluating explainability methods make it a challenging field.

Overall, it is clear that explainability is a complex problem that will continue to be a challenge. While I would assess that the field is currently moving away from heuristic assumptions and towards more rigorous methods, the models have also become more complex than ever. Especially for large language models (Brown et al. 2020), it is unlikely to provide a comprehensive explanation of the model's behavior.

8.2 Future Prospects

The trade-off between model complexity and interpretability could be analyzed in more detail. In particular, it would be interesting to find a quantitative measure for interpretability and then to characterize the connection to the model complexity.

Current theoretical analyzes of explainability methods are often focused on specific methods, e.g., (Lundstrom et al. 2022) for integrated gradients or (Sixt et al. 2022a) for Deep Taylor Decomposition. Future work could create more general theoretical frameworks for explainability methods. Similar to the ACID tests to check CSS conformality, one could develop a set of theoretical and empirical tests to validate explainability methods.

Bibliography

Adebayo, J., J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim (2018). "Sanity checks for saliency maps". In: *Advances in Neural Information Processing Systems*, pp. 9505–9515.

Alqaraawi, A., M. Schuessler, P. Weiß, E. Costanza, and N. Berthouze (2020). "Evaluating Saliency Map Explanations for Convolutional Neural Networks: A User Study". In: *Proceedings of the 25th International Conference on Intelligent User Interfaces*. IUI '20. Cagliari, Italy: Association for Computing Machinery, pp. 275–285. ISBN: 9781450371186. DOI: 10.1145/3377325.3377519. URL: https://doi.org/10.1145/3377325.3377519.

Ancona, M., E. Ceolini, C. Öztireli, and M. Gross (2017). "A unified view of gradient-based attribution methods for deep neural networks". In: *NIPS 2017-Workshop on Interpreting, Explaining and Visualizing Deep Learning*. ETH Zurich.

Arras, L., A. Osman, and W. Samek (2022). "CLEVR-XAI: A benchmark dataset for the ground truth evaluation of neural network explanations". In: *Information Fusion* 81, pp. 14–40. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2021.11.008. URL: https://www.sciencedirect.com/science/article/pii/S1566253521002335.

Bach, S., A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek (2015). "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLoS ONE* 10.7.

Baehrens, D., T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller (2010). "How to explain individual classification decisions". In: *Journal of Machine Learning Research* 11.Jun, pp. 1803–1831.

Bauer, K., O. Hinz, W. van der Aalst, and C. Weinhardt (2021). *Expl (AI) n it to me–explainable AI and information systems research*.

Berzuini, C. (1988). "Combining Symbolic Learning Techniques and Statistical Regression Analysis." In: *AAAI*, pp. 612–617.

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter,

C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Cao, N. D., M. Schlichtkrull, W. Aziz, and I. Titov (2020). *How do Decisions Emerge across Layers in Neural Models? Interpretation with Differentiable Masking*. arXiv: 2004.14992 [cs.CL].

Chen, C., O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su (2019a). "This looks like that: deep learning for interpretable image recognition". In: *Advances in neural information processing systems*, pp. 8930–8941.

Chen, J., L. Song, M. J. Wainwright, and M. I. Jordan (2019b). "L-Shapley and C-Shapley: Efficient Model Interpretation for Structured Data". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=S1E3Ko09F7.

Craven, M. W. and J. W. Shavlik (1993). "Learning Symbolic Rules Using Artificial Neural Networks". en. In: *Machine Learning Proceedings 1993*. Elsevier, pp. 73–80. ISBN: 978-1-55860-307-3. DOI: 10.1016/B978-1-55860-307-3.50016-2. URL: https://linkinghub.elsevier.com/retrieve/pii/B9781558603073500162.

DeJong, G. (Aug. 1981). "Generalizations based on explanations". In: *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 1*. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 67–69.

Dejong, G. and R. Mooney (1986). "Explanation-based learning: An alternative view". en. In: *Machine Learning* 1.2, pp. 145–176. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00114116. URL: http://link.springer.com/10.1007/BF00114116.

Dimopoulos, I., J. Chronopoulos, A. Chronopoulou-Sereli, and S. Lek (Aug. 1999). "Neural network models to study relationships between lead concentration in grasses and permanent urban descriptors in Athens city (Greece)". en. In: *Ecological Modelling* 120.2-3, pp. 157–165. ISSN: 03043800. DOI: 10.1016/S0304-3800(99)00099-X. URL: https://linkinghub.elsevier.com/retrieve/pii/S030438009900099X.

Dimopoulos, Y., P. Bourret, and S. Lek (Dec. 1995). "Use of some sensitivity criteria for choosing networks with good generalization ability". en. In:

Neural Processing Letters 2.6, pp. 1–4. ISSN: 1573-773X. DOI: 10.1007/BF02309007. URL: https://doi.org/10.1007/BF02309007.

Ellman, T. (June 1989). "Explanation-based learning: a survey of programs and perspectives". en. In: *ACM Computing Surveys* 21.2, pp. 163–221. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/66443.66445. URL: https://dl.acm.org/doi/10.1145/66443.66445.

European Commission (2021). *LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS*. URL: https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence.

Fikes, R. E., P. E. Hart, and N. J. Nilsson (Jan. 1972). "Learning and executing generalized robot plans". en. In: *Artificial Intelligence* 3, pp. 251–288. ISSN: 00043702. DOI: 10.1016/0004-3702(72)90051-3. URL: https://linkinghub.elsevier.com/retrieve/pii/0004370272900513.

Friedman, E. J. (2004). "Paths and Consistency in Additive Cost Sharing". In: *Int. J. Game Theory* 32.4, pp. 501–518. ISSN: 0020-7276. DOI: 10.1007/s001820400173. URL: https://doi.org/10.1007/s001820400173.

Fu, L. and T. Chen (1993). "Sensitivity analysis for input vector in multilayer feedforward neural networks". en. In: *IEEE International Conference on Neural Networks*. San Francisco, CA, USA: IEEE, pp. 215–218. ISBN: 978-0-7803-0999-9. DOI: 10.1109/ICNN.1993.298559. URL: http://ieeexplore.ieee.org/document/298559/.

Garson, G. D. (Apr. 1991). "Interpreting neural-network connection weights". In: *AI Expert* 6.4, pp. 46–51. ISSN: 0888-3785.

Gedeon, T. D. (Apr. 1997). "Data Mining of Inputs: Analysing Magnitude and Functional Measures". en. In: *International Journal of Neural Systems* 08.02, pp. 209–218. ISSN: 0129-0657, 1793-6462. DOI: 10.1142/S0129065797000227. URL: https://www.worldscientific.com/doi/abs/10.1142/S0129065797000227.

Gevrey, M., I. Dimopoulos, and S. Lek (Feb. 2003). "Review and comparison of methods to study the contribution of variables in artificial neural network models". en. In: *Ecological Modelling* 160.3, pp. 249–264. ISSN: 03043800. DOI: 10.1016/S0304-3800(02)00257-0. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304380002002570.

– (May 2006). "Two-way interaction of input variables in the sensitivity analysis of neural network models". en. In: *Ecological Modelling*. Selected Papers

from the Third Conference of the International Society for Ecological Informatics (ISEI), August 26–30, 2002, Grottaferrata, Rome, Italy 195.1, pp. 43–50. ISSN: 0304-3800. DOI: 10.1016/j.ecolmodel.2005.11.008. URL: https://www.sciencedirect.com/science/article/pii/S0304380005005752.

Ghorbani, A., J. Wexler, J. Y. Zou, and B. Kim (2019). "Towards automatic concept-based explanations". In: *Advances in Neural Information Processing Systems*, pp. 9277–9286.

Giles, C. and C. Omlin (1993). "Rule refinement with recurrent neural networks". en. In: *IEEE International Conference on Neural Networks*. San Francisco, CA, USA: IEEE, pp. 801–806. ISBN: 978-0-7803-0999-9. DOI: 10.1109/ICNN.1993.298658. URL: http://ieeexplore.ieee.org/document/298658/.

Goh, A. (Jan. 1995). "Back-propagation neural networks for modeling complex systems". en. In: *Artificial Intelligence in Engineering* 9.3, pp. 143–151. ISSN: 09541810. DOI: 10.1016/0954-1810(94)00011-S. URL: https://linkinghub.elsevier.com/retrieve/pii/095418109400011S.

Gorry, G. A. (1967). "A System for Computer-Aided Diagnosis". Technical Report. USA: Massachusetts Institute of Technology.

Gorry, G. A. (1973). "Computer-Assisted Clinical Decision Making". In: *Methods of Information in Medicine* 12, pp. 45–51.

Hase, P. and M. Bansal (July 2020). "Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior?" In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 5540–5552. DOI: 10.18653/v1/2020.acl-main.491. URL: https://aclanthology.org/2020.acl-main.491.

Hauser, K., A. Kurz, S. Haggenmüller, R. C. Maron, C. von Kalle, J. S. Utikal, F. Meier, S. Hobelsberger, F. F. Gellrich, M. Sergon, et al. (2022). "Explainable artificial intelligence in skin cancer recognition: A systematic review". In: *European Journal of Cancer* 167, pp. 54–69.

Holzinger, A., A. Saranti, C. Molnar, P. Biecek, and W. Samek (2022). "Explainable AI Methods - A Brief Overview". In: *xxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*. Cham: Springer International Publishing, pp. 13–38. ISBN: 978-3-031-04083-2. DOI: 10.1007/978-3-031-04083-2_2. URL: https://doi.org/10.1007/978-3-031-04083-2_2.

Jiang, Z., R. Tang, J. Xin, and J. Lin (Nov. 2020). "Inserting Information Bottlenecks for Attribution in Transformers". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 3850–3857. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.343.

Kim, B., M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. (2018a). "Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)". In: *International Conference on Machine Learning*, pp. 2668–2677.

Kim, B., J. Seo, S. Jeon, J. Koo, J. Choe, and T. Jeon (2019). "Why are Saliency Maps Noisy? Cause of and Solution to Noisy Saliency Maps". In: *arXiv: 1902.04893*. eprint: 1902.04893 (cs.LG).

Kim, J., A. Rohrbach, T. Darrell, J. Canny, and Z. Akata (2018b). "Textual Explanations for Self-Driving Vehicles". en. In: *Computer Vision – ECCV 2018*. Ed. by V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss. Vol. 11206. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 577–593. ISBN: 978-3-030-01215-1 978-3-030-01216-8. DOI: 10.1007/978-3-030-01216-8_35. URL: https://link.springer.com/10.1007/978-3-030-01216-8_35.

Kindermans, P.-J., K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, and S. Dähne (Feb. 2018a). "Learning how to explain neural networks: PatternNet and PatternAttribution". en. In: URL: https://openreview.net/forum?id=Hkn7CBaTW.

– (2018b). "Learning how to explain neural networks: PatternNet and PatternAttribution". In: *International Conference on Learning Representations*.

Kingma, D. P. and M. Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Kumar, I. E., S. Venkatasubramanian, C. Scheidegger, and S. Friedler (2020). "Problems with Shapley-value-based explanations as feature importance measures". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 5491–5500. URL: https://proceedings.mlr.press/v119/kumar20e.html.

LeCun, Y., J. Denker, and S. Solla (1989). "Optimal Brain Damage". In: *Advances in Neural Information Processing Systems*. Vol. 2. Morgan-Kaufmann. URL: https://papers.nips.cc/paper/1989/hash/6c9882bbac1c7093bd250418812
Abstract.html.

Lek, S., A. Belaud, P. Baran, I. Dimopoulos, and M. Delacoste (Jan. 1996). "Role of some environmental variables in trout abundance models using neural networks". en. In: *Aquatic Living Resources* 9.1, pp. 23–29. ISSN: 0990-7440. DOI: 10.1051/alr:1996004. URL: http://www.alr-journal.org/10.1051/alr:1996004.

Lundstrom, D. D., T. Huang, and M. Razaviyayn (2022). "A Rigorous Study of Integrated Gradients Method and Extensions to Internal Neuron Attributions". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 14485–14508. URL: https://proceedings.mlr.press/v162/lundstrom22a.html.

Mahoney, J. and R. Mooney (1992). "Combining Neural and Symbolic Learning to Revise Probabilistic Rule Bases". In: *Advances in Neural Information Processing Systems*. Vol. 5. Morgan-Kaufmann. URL: https://proceedings.neurips.cc/paper/1992/hash/f76a89f0cb91bc419542ce9fa43902dc-Abstract.html.

Milne, L. (1995). "Feature selection using neural networks with contribution measures". English. In: DOI: 10.26190/unsworks/378. URL: http://hdl.handle.net/1959.4/37628.

Mitchell, T. M. (Mar. 1982). "Generalization as search". en. In: *Artificial Intelligence* 18.2, pp. 203–226. ISSN: 0004-3702. DOI: 10.1016/0004-3702(82)90040-6. URL: https://www.sciencedirect.com/science/article/pii/0004370282900406.

Mitchell, T. M., R. M. Keller, and S. T. Kedar-Cabelli (Mar. 1986). "Explanation-Based Generalization: A Unifying View". en. In: *Machine Learning* 1.1, pp. 47–80. ISSN: 1573-0565. DOI: 10.1023/A:1022691120807. URL: https://doi.org/10.1023/A:1022691120807.

Montavon, G., S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller (2017). "Explaining nonlinear classification decisions with deep taylor decomposition". In: *Pattern Recognition* 65, pp. 211–222.

Mozer, M. C. and P. Smolensky (1988). "Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment". In: *Advances in Neural Information Processing Systems*. Vol. 1. Morgan-Kaufmann. URL: https://proceedings.neurips.cc/paper/1988/hash/07e1cd7dca89a1678042477183b7ac3f-Abstract.html.

Nader, Y., L. Sixt, and T. Landgraf (2022). "DNNR: Differential Nearest Neighbors Regression". In: *Proceedings of the 39th International Conference on*

Machine Learning. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 16296–16317. URL: https://proceedings.mlr.press/v162/nader22a.html.

Olden, J. (May 2004). "An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data". In: *Ecological Modelling* 178. DOI: 10.1016/S0304-3800(04)00156-5.

Olden, J. D. and D. A. Jackson (Aug. 2002). "Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks". en. In: *Ecological Modelling* 154.1-2, pp. 135–150. ISSN: 03043800. DOI: 10.1016/S0304-3800(02)00064-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304380002000649.

Pastor-Bárcenas, O., E. Soria-Olivas, J. Martín-Guerrero, G. Camps-Valls, J. Carrasco-Rodríguez, and S. d. Valle-Tascón (Mar. 2005). "Unbiased sensitivity analysis and pruning techniques in neural networks for surface ozone modelling". en. In: *Ecological Modelling* 182.2, pp. 149–158. ISSN: 03043800. DOI: 10.1016/j.ecolmodel.2004.07.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304380004004107.

Ribeiro, M. T., S. Singh, and C. Guestrin (2016). ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, pp. 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778.

Roscher, R., B. Bohn, M. F. Duarte, and J. Garcke (2020). "Explainable machine learning for scientific insights and discoveries". In: *Ieee Access* 8, pp. 42200–42216.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Samek, W., A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller (2016). "Evaluating the visualization of what a deep neural network has learned". In: *IEEE transactions on neural networks and learning systems* 28.11, pp. 2660–2673.

Sanger, D. (Jan. 1989). "Contribution Analysis: A Technique for Assigning Responsibilities to Hidden Units in Connectionist Networks". In: *Connection Science* 1.2, pp. 115–138. ISSN: 0954-0091. DOI: 10.1080/09540098908915632. URL: https://doi.org/10.1080/09540098908915632.

Sarle, W. S. (2000). "How to measure importance of inputs?" en. In: p. 24. URL: ftp://ftp.sas.com/pub/neural/importance.html.

Scardi, M. and L. W. Harding (Aug. 1999). “Developing an empirical model of phytoplankton primary production: a neural network case study”. en. In: *Ecological Modelling* 120.2-3, pp. 213–223. ISSN: 03043800. DOI: 10.1016/S0304-3800(99)00103-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304380099001039.

Schulz, K. (2019). “Attribution by Information Measurement”. MA thesis. TU München.

Schulz, K., L. Sixt, F. Tombari, and T. Landgraf (2020). “Restricting the Flow: Information Bottlenecks for Attribution”. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=HkxJzAEtvS.

Sejnowski, T. J. and C. R. Rosenberg (1987). “Parallel networks that learn to pronounce English text”. In: *Complex systems* 1.1, pp. 145–168.

Setiono, R. and H. Liu (1996). “Symbolic representation of neural networks”. In: *Computer* 29.3. Publisher: IEEE, pp. 71–77.

Shapley, L. S. (1951). *Notes on the N-Person Game – II: The Value of an N-Person Game*. Santa Monica, CA: RAND Corporation. DOI: 10.7249/RM0670.

Shortliffe, E. H. (1976). *Computer-based medical consultations, MYCIN*. en. Artificial intelligence series 2. New York: Elsevier. ISBN: 978-0-444-00179-5.

Shrikumar, A., P. Greenside, and A. Kundaje (2017a). “Learning Important Features Through Propagating Activation Differences”. In: *ICML*.

Shrikumar, A., P. Greenside, and A. Kundaje (2017b). “Learning important features through propagating activation differences”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org, pp. 3145–3153.

Silver, B. (1983). “Learning equation solving methods from examples”. In: *IJCAI*, pp. 429–431.

Simonyan, K., A. Vedaldi, and A. Zisserman (2013). “Deep inside convolutional networks: Visualising image classification models and saliency maps”. In: *arXiv preprint arXiv:1312.6034*.

Sixt, L., M. Granz, and T. Landgraf (2020). “When Explanations Lie: Why Many Modified BP Attributions Fail”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 9046–9057. URL: https://proceedings.mlr.press/v119/sixt20a.html.

Sixt, L. and T. Landgraf (2022a). "A Rigorous Study Of The Deep Taylor Decomposition". In: *Transactions on Machine Learning Research*. URL: https://openreview.net/forum?id=Y4mgmw9OgV.

Sixt, L., M. Schuessler, O.-I. Popescu, P. Weiß, and T. Landgraf (2022b). "Do Users Benefit From Interpretable Vision? A User Study, Baseline, And Dataset". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=v6s3HVjPerv.

Springenberg, J. T., A. Dosovitskiy, T. Brox, and M. Riedmiller (2014). "Striving for Simplicity: The All Convolutional Net". In: *arXiv: 1412.6806*. eprint: 1412.6806 (cs.LG).

Springenberg, J. T., A. Dosovitskiy, T. Brox, and M. Riedmiller (2014). "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806*.

Sundararajan, M., A. Taly, and Q. Yan (2017). "Axiomatic attribution for deep networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 3319–3328.

Sung, A. H. (Oct. 1998). "Ranking importance of input parameters of neural networks". en. In: *Expert Systems with Applications* 15.3, pp. 405–411. ISSN: 0957-4174. DOI: 10.1016/S0957-4174(98)00041-4. URL: https://www.sciencedirect.com/science/article/pii/S0957417498000414.

Teach, R. L. and E. H. Shortliffe (Dec. 1981). "An analysis of physician attitudes regarding computer-based clinical consultation systems". en. In: *Computers and Biomedical Research* 14.6, pp. 542–558. ISSN: 0010-4809. DOI: 10.1016/0010-4809(81)90012-4. URL: https://www.sciencedirect.com/science/article/pii/0010480981900124.

Tishby, N., F. C. Pereira, and W. Bialek (2000). "The information bottleneck method". In: *arXiv preprint physics/0004057*.

Towell, G. and J. Shavlik (1991). "Interpretation of Artificial Neural Networks: Mapping Knowledge-Based Neural Networks into Rules". In: *Advances in Neural Information Processing Systems*. Vol. 4. Morgan-Kaufmann. URL: https://proceedings.neurips.cc/paper/1991/hash/ed265bc903a5a097f61d3ec064d96d2e-Abstract.html.

Towell, G. G. (1993). "Symbolic knowledge and neural networks: Insertion, refinement and extraction." PhD thesis.

Tresp, V., J. Hollatz, and S. Ahmad (1992). "Network Structuring and Training Using Rule-based Knowledge". In: *Advances in Neural Information Processing Systems*. Vol. 5. Morgan-Kaufmann. URL: https://proceedings.

neurips.cc/paper/1992/hash/4c27cea8526af8cfee3be5e183ac9605-Abstract.html.

Weizenbaum, J. (Jan. 1966). "ELIZA—a computer program for the study of natural language communication between man and machine". In: *Communications of the ACM* 9.1, pp. 36–45. ISSN: 0001-0782. DOI: 10.1145/365153.365168. URL: https://doi.org/10.1145/365153.365168.

Winograd, T. (1972). "Understanding natural language". In: *Cognitive psychology* 3.1, pp. 1–191.

Zeiler, M. D. and R. Fergus (2014). "Visualizing and Understanding Convolutional Networks". In: *European conference on computer vision*. Springer, pp. 818–833.

Zhang, J., S. A. Bargal, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff (2018). "Top-down neural attention by excitation backprop". In: *International Journal of Computer Vision* 126.10, pp. 1084–1102.

Zhang, R., P. Madumal, T. Miller, K. A. Ehinger, and B. I. Rubinstein (2021a). "Invertible concept-based explanations for cnn models with non-negative concept activation vectors". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 13, pp. 11682–11690.

Zhang, Y., A. Khakzar, Y. Li, A. Farshad, S. T. Kim, and N. Navab (2021b). "Fine-Grained Neural Network Explanation by Identifying Input Features with Predictive Information". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. URL: https://openreview.net/forum?id=HglgPZAYhcG.

Zusammenfassung Auf Deutsch

In meiner Dissertation habe ich mich mit der Interpretierbarkeit von künstlichen neuronalen Netzwerken beschäftigt. Interpretierbarkeit bedeutet die Fähigkeit eines Menschen deren Funktionsweise und Verhalten zu verstehen.

Durch insgesamt fünf Publikationen habe ich mich intensiv mit dem Thema beschäftigt. In der ersten Publikation (Schulz u. a. 2020) haben wir analysiert, inwiefern mit Hilfe eines informationstheoretischen Flaschenhalses, der die Informationsmenge beschränkt, neuronale Netzwerke besser verstanden werden können. Hierbei haben wir über mehrere Metriken eine Verbesserung gegenüber anderen Methoden festgestellt. In den Experimenten haben wir auch beobachtet, dass bestimmte LRP-Methoden (Bach u. a. 2015) invariant zu einer vollständig zufälligen Änderung der Gewichten in den hinteren Netzwerkschichten ist.

Dieses habe ich dann in der zweiten Publikation (Sixt u. a. 2022b) sowohl theoretisch als auch experimentell zeigen können, dass durch eine unabsichtliche Verwendung von positiven Gewichten die Methode zu einer Rank-1 Matrix konvergiert.

In meiner neuesten Arbeit (Sixt u. a. 2022a) habe ich mich mit dem theoretischen Überbau der LRP Methode beschäftigt, der sogenannten Deep Taylor Decomposition. Hier konnte ich grundlegende Mängel in dem theoretischen Überbau aufzeigen.

Des Weiteren haben wir untersucht, wie man Interpretierbarkeit in einer Online-Benutzerstudie messen kann. Hier konnten wir zeigen, dass auch eine einfache Erklärungsmethode, die nur auf der Netzwerkausgabe basiert, ein ähnliches Level von Interpretierbarkeit erreichen kann, wie im Vergleich komplexere Methoden, wie invertierbare neuronalen Netze.

In der Publikation (Nader u. a. 2022), haben wir eine einfache Erweiterung der kNN-Regression für höhere Ordnungen vorgestellt. Hierbei konnten wir eine bessere Generalisierung erreichen als die klassische kNN-Regression und durchaus auch mit Gradient-Boosting Methoden mithalten konnten.

Insgesamt hat meine Dissertation gezeigt, dass die Interpretierbarkeit von neuronalen Netzen ein schwieriges und oft unintuitives Thema ist, bei dem man auch schnell zu falschen Schlüssen kommen kann.