A Geometrical Perspective on Explanations for Deep Neural Networks

vorgelegt von

Ann-Kathrin Dombrowski, M.Sc.

an der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –
genehmigte Dissertation

Promotionsausschuss:

Vorsitz:	Prof. Dr. Benjamin Blankertz
Gutachter:	Prof. Dr. Klaus-Robert Müller
	Prof. Dr. Alexander Binder
	Prof. Dr. Wojciech Samek

Tag der wissenschaftlichen Aussprache: 09.12.2022

Berlin 2023

Abstract

In the past decade, artificial neural networks have seen unprecedented gains in capabilities and applications. With their increased popularity, the need for a more detailed understanding of how these models reach certain decisions emerged. The field of explainable artificial intelligence has therefore attracted significant attention in recent years, promising to provide insight into the decision-making process of deep neural networks.

Of course, explainability has its own caveats, and results produced by explanation methods are not always well understood. This curtails acceptance and effective application of explanation methods.

In this thesis we therefore work towards a unified geometrical understanding of explainability. We analyse undesired properties of explanation methods using concepts from differential geometry, and find countermeasures that improve their robustness and interpretability.

In the first part, we show that many popular gradient- and propagation-based explanations can be arbitrarily manipulated to fit an attacker's desired output. We analyse this surprising behavior theoretically and connect the explanation's susceptibility to manipulation to the high curvature of the network's output manifold. Based on these insights, we propose β-smoothing, a novel explanation method that is more robust against adversarial perturbations. Furthermore, we investigate how a changed training regime can reduce the curvature of a neural network and derive different regularizers which boost the robustness of explanations.

In the second part, we focus on another popular field of explainability, namely counterfactual explanations. These can be interpreted very intuitively and are therefore of tremendous value in medicine, finance, law, and other areas where user-friendly explanations are paramount. However, finding counterfactuals with structural differences to the query input, which stand in contrast to mere adversarial examples, can be difficult. Investigating this challenge from a geometrical point of view leads us to the insight that finding a suitable coordinate system for the search process reduces the generation of counterfactuals to a simple gradient ascent optimization. We then introduce an elegant, yet effective algorithm that makes use of the latent space of a generative model to produce high-quality counterfactuals which lie on the data manifold.

Zusammenfassung

Künstliche neuronale Netze haben in den letzten zehn Jahren eine beispiellose Renaissance erlebt, doch es ist noch immer schwierig den Entscheidungsprozess solcher neuronalen Netze nachzuvollziehen. So haben kürzlich Methoden der erklärbaren künstlichen Intelligenz viel Aufmerksamkeit auf sich gezogen, da diese dabei helfen können die Entscheidungsprozesse besser zu verstehen.

Ergebnisse von Erklärmethoden sind allerdings selbst nicht immer gut verständlich. Dies schränkt die Akzeptanz und den effektiven Einsatz von Erklärungsmethoden ein. In dieser Arbeit arbeiten wir daher auf ein einheitliches geometrisches Verständnis von Erklärbarkeit hin.

Mit Konzepten aus der Differentialgeometrie analysieren wir unerwünschte Eigenschaften von Erklärungsmethoden und finden Gegenmaßnahmen, welche die Robustheit und Interpretierbarkeit von Erklärungen verbessern.

Im ersten Teil zeigen wir, dass viele populäre gradienten- und propagationsbasierte Erklärungen willkürlich manipuliert werden können. Dieses überraschende Verhalten analysieren wir theoretisch und finden Parallelen zwischen der Manipulierbarkeit der Erklärung und der Krümmung der Ergebnismannigfaltigkeit des neuronalen Netzes. Basierend auf diesen Einblicken präsentieren wir β-smoothing, eine neue Erklärmethode, die robuster gegen adverserielle Störungen ist. Darüber hinaus untersuchen wir, wie ein verändertes Trainingsregime die Krümmung eines neuronalen Netzes reduzieren kann und leiten verschiedene Regularisierer her, welche die Robustheit von Erklärungen verbessern.

Im zweiten Teil konzentrieren wir uns auf einen anderen populären Bereich der Erklärbarkeit, nämlich kontrafaktische Erklärungen. Diese lassen sich sehr intuitiv interpretieren und sind daher wichtig in der Medizin, im Finanzwesen, im Recht und in anderen Bereichen, in denen benutzerfreundliche Erklärungen von größter Bedeutung sind. Die Suche nach kontrafaktischen Erklärungen mit strukturellen Unterschieden zur Eingabe, die im Gegensatz zu reinen adverseriellen Beispielen stehen, kann allerdings schwierig sein. Die Untersuchung dieser Herausforderung aus geometrischer Sicht führt uns zu der Erkenntnis, dass der Gebrauch eines geeigneten Koordinatensystems für den Suchprozess die Generierung von kontrafaktischen Erklärungen auf eine einfache Gradientenanstiegs Optimierung reduziert. Wir stellen einen effektiven Algorithmus vor, der den latenten Raum eines generativen Modells nutzt, um hochwertige kontrafaktische Erklärungen zu erzeugen, die auf der Datenmannigfaltigkeit liegen.

Acknowledgements

First of all, I want to thank Klaus for giving me the opportunity to work at IDA and for creating this unique atmosphere where so many skilled people with different backgrounds can come together, discuss, collaborate and help each other with enthusiasm. Thank you also for your guidance, your patience, and your confidence in my ability to figure out my path.

A huge thanks goes to Pan, who joined the lab a few months after I started and has been a supervisor and friend to me ever since. Thank you for your great ideas that turned into great papers, your positive attitude, your enthusiasm for differential geometry and machine learning, your explanations, and your help with mathematical concepts and scientific writing. I couldn't have done this without you.

I'm also grateful to Christopher for joint programming sessions and early discussions about explainability. Thank you so much for answering all the questions I had, it was always fun and productive working with you.

Thanks also to Jan for the many Zoom calls during Covid, for the invitation to visit Chalmers in Sweden, for the shared meals, laughs and the close collaboration on two research papers. I also highly appreciate you proofreading so much of my thesis and giving me constructive feedback.

Besides the people I closely collaborated with, I also wanna thank Malte, Nico, Bettina, Adrian, Jacob, Thomas, Miriam, Steffi, Daniel, Alex, Max, Philip, Kim, Nikolas, Mihail, Binh, Sergej, Marina, Oliver, Stefan, Michael, David, Hannah, Rob, Shin, Kristof and all other lovely people at IDA for sharing breaks, discussions, offices, canoes, parties, company runs and conference apartments with me.

Thanks to everyone that shared my time at the DAEDALUS research training group. Most of all thanks to Wolf for enlightening discussions about turbulence. Thanks to Ingo, Benjamin, Jiahan, Raquel, Philipp, Jonas, Ines, Nils, Oleksandr, Paul, Leon, and Qiao for sharing laughs and conversations at retreats.

I also highly appreciate the administrative work done by Andrea, Kerstin, Imke, and Cicilia and their help whenever I had an organizational problem. Thanks to Dominik for the cluster care and IT help.

Of course, the time of my PhD was not solely spent at the (home) office, so I wanna thank all the people that were not directly involved with my work but sure helped with having a healthy work-life balance.

I'm so happy to have Roman, Jan, Hüseyin, Michelle, Melanie, Charlotte, Sasha, Vitalí, and all other Forrózeiras e Forrózeiros in my life who share this love for dancing with me.

Besides my dancing friends, I'm super lucky to have met Johanna and all the other aspiring aerialists that courageously hang themselves again and again at bridges outside and beams in Katapult.

Thanks to the EA and rationalist communities for challenging my presumptions on social norms and ethics and providing me with a huge pool of awesome people to get to know.

A shout-out to all my lovely flatmates Josi, Jonas, Lennart, Eva, Daniel, Ina, Lisa, Fela, and Anne who shared living space, coffee, meals, workouts, meetups, and parties with me.

Thanks to my parents, Eva and Tilman, for their never-ending interest in whatever I present them with and to my sister, Olga, who makes for the best company whenever we're together and shares an appropriately high level of energy with me. I couldn't imagine a better family to have in my life.

Finally thanks to Jay, who has been with me for most of my PhD. I love you for being my best friend, my partner, my rubber duck for life- and work-debugging, my adventure-time-water-carrier, my oracle of reason, my snowboard buddy, and everything else that comes with being the weird human-crypto-token hybrid that you are...

Contents

1. Introduction

Machine learning models have made impressive progress at many different tasks, such as generating high-quality text and images [1–5], predicting the three-dimensional structure of proteins [6] and surpassing human performance in numerous games [7, 8] and classification tasks [9, 10].

Nearly all of these gains in ability can be attributed to deep neural networks which are described by millions or billions of parameters, whose values are determined during a training phase.

Although well-trained neural networks can produce highly accurate predictions, the decision-making processes behind them lack transparency, rendering them incomprehensible to humans. This reduces confidence in decisions made by these models and is problematic in safety-critical areas where we want to ensure that the prediction was based on valid input features, for example in autonomous driving, or in areas where a prediction alone is insufficient, for example in medical diagnostics.

Therefore, interest in visualizing, explaining, and interpreting deep neural networks has soared in recent years and the field of explainable artificial intelligence (XAI) [11] emerged. Explanation methods allow us to gain insights into the decision processes of machine learning models, and understand how they infer certain predictions. Identifying the patterns and strategies used for such predictions can promote new scientific discoveries [12–14]. Furthermore, explainability could become an integral part of the training and validation process of machine learning systems as explanation methods can facilitate the detection—and subsequent correction—of failure modes in deep learning. A recent example is the application of explanation methods to discover that some classifiers base their predictions on spurious correlations in the data set [15, 16]. The model's accuracy is then high on training and test data stemming from the same distribution but quickly degrades when the data distribution changes slightly.

Although explanation methods have provided valuable information about model behavior, some aspects of explainability are poorly understood [17–20] which hinders the potential acceptance and broad application of explanation methods. Especially theoretical inspection of explanation methods is, with few exceptions [21, 22], deficient in explainability research.

In this thesis we address current limitations of explanation methods and establish a precise understanding by analyzing the respective origins from a geometrical

perspective. Awareness of potential shortcomings of explanations is clearly essential in scenarios where important decisions are based on the explanation. It is also crucial for overcoming those limitations. We demonstrate this by using our theoretical insights to derive efficient methods that boost the robustness and interpretability of explanations. The research presented in this thesis is therefore an important contribution towards the development of trustworthy, coherent, and intelligible explanations for deep neural networks.

We focus on two popular sub-fields of explainability: attribution methods and counterfactual explanations.

Attribution methods [23–28] provide explanation maps which highlight the areas of the input deemed most important for a classifier's decision. These methods are widely used because of their straightforward implementation and quick results.

We demonstrate that many common attribution methods are not as reliable as hoped as their explanation maps can be manipulated to match arbitrary targets by adding adversarial perturbations to the input. This suggests gaps in our understanding of attribution methods.

A geometrical perspective can reveal why such manipulations are possible. For example, many attribution methods rely on the gradient of the neural network's output with respect to the input. This gradient can be thought of as a high-dimensional vector which is perpendicular to the hypersurface of equal network output. Our observation that explanation maps can be arbitrarily manipulated while the output stays constant indicates a high curvature of the output manifold.

Within this thesis, we make the above statements precise using tools from differential geometry and consequently accomplish a fundamental understanding of how explanation methods respond to input modifications. We then use our insights to develop effective counter measures. The resulting explanations are provably more robust against input manipulation, which we confirm through numerous experiments.

Attribution methods have been proven to be useful in various applications [15, 29, 30] but we sometimes desire explanations that are specific to selected targets or easier to understand by laypeople. In these cases, counterfactual explanations [31–35] can be beneficial. While attribution methods highlight relevant features in the query input, counterfactual explanations provide hypothetical alternative instances for selected targets and therefore offer more specific information. As counterfactuals are contrastive to the query input and usually focus on a small number of input features, they are regarded as easily interpretable, and thus, human-friendly explanations.

Counterfactuals are usually achieved by following the gradient of a specified target classification with respect to the original query input [31]. While this approach works reasonably well for low-dimensional (tabular) data, we run into complications when applying it to high-dimensional data, such as images.

Performing simple gradient updates on image data usually leads to adversarial

examples [36], which appear indistinguishable from the original image but cause a switch in classification by the neural network. The added perturbations are highly engineered and do not occur naturally. Consequently, we cannot hope to gain much information about what features from the input data were important. Structural alterations in the counterfactual, on the other hand, would improve our understanding of what naturally occurring input would cause a change in classification.

The following geometrical considerations are central to comprehending the reasons behind this problem. Natural images lie on a very low dimensional manifold (the data manifold) embedded in high-dimensional space. When training a classifier, the training images stem from this data manifold. However, the classifier can be evaluated on any input with the correct dimensions and might give arbitrary results on these inputs. The gradient of the classifier output with respect to the input points in the direction that would locally cause the biggest change in the prediction. This direction is generally not aligned with the data manifold. Following the gradient thus leads off manifold and to inputs which would not occur naturally but cause a large change in classifier output.

This leads us to question the choice of coordinate system in which the gradient updates—to find counterfactuals—are most commonly performed. Ideally we would perform such updates in a coordinate system in which the gradient is aligned with the data manifold and we therefore avoid running into adversarial examples.

We prove that a suitable coordinate system can be found by leveraging the latent space of a well trained generative model. This greatly facilitates the search for counterfactuals on the data manifold. On the basis of these different coordinate systems we investigate the connection between adversarial examples and counterfactuals theoretically and propose an elegant method to find counterfactuals from which we can gain valuable information about the decision process.

In the remainder of this chapter we will describe our contributions towards a unified geometrical understanding of explanation methods in more detail.

1.1. Contributions

As outlined above, we need methods from explainable artificial intelligence for reliable insight into the decision-making processes of deep neural networks in order to promote scientific discovery and examine if these models work as intended. Some behaviors of explanation methods are not properly understood, which limits trust in these methods and therefore reduces their usefulness.

This thesis investigates properties of explanations from a geometrical perspective. We are especially interested in how to assure that explanations are robust and easily interpretable as these characteristics are desirable for all applications of explainability. The research results were published at machine learning conferences and journals in

the form of four papers (see Section 1.1.1 for details).

Our robustness analysis focuses on attribution methods. We research the vulnerability of explanations when manipulating the input using concepts from differential geometry and propose measures to overcome this unwanted behavior.

In [37], we show an alarming property of attribution methods: they can be arbitrarily manipulated by adding small perturbations to the input, which, crucially, do not alter the prediction. We find these perturbations by minimizing the mean squared error between the explanation map and a given target map, iteratively updating the input. The approach is analogous to conventional adversarial attacks, which aim to change the neural network's prediction. A non-trivial difference is that the target for adversarial attacks on the explanation is usually orders of magnitude larger than for conventional adversarial attacks, as the dimension of the explanation map normally matches the input dimension.

We theoretically analyze the surprising susceptibility of explanations to adversarial attacks using tools from differential geometry and derive an upper bound on the maximal change in the explanation map. Based on these insights, we propose a new explanation method—β-smoothing—that produces crisp attribution maps and provably reduces the upper bound on the change in the explanation map. We confirm experimentally that the newly found explanations are indeed more robust against adversarial attacks.

While in [37] we change the explanation procedure for pre-trained neural networks, in [38], we investigate techniques that can be incorporated into the training phase so that applying methods from XAI to the newly trained networks results in robust explanation maps. Our theoretical analysis is based on the findings from our previous paper. The upper bound for the change in explanation, i.e., the upper bound on the Hessian of the network, derived in [37] depends on the first and second derivatives of the (smooth) activation function and the network's weights.

We generalize this upper bound to the non-smooth ReLU activation and identify three different interventions we can incorporate into the training, which all reduce the upper bound. The first method uses weight decay, a regularization technique usually applied to avoid overfitting and to improve accuracy. Our theoretical results suggest that it can also boost robustness. As a second measure, we propose to substitute ReLU activations with softplus activations during training and application, as softplus activations do not have the typical kinks of ReLU activations that lead to rapidly changing gradients when slightly perturbing the input. The third approach aims to minimize the Hessian directly by penalizing its Frobenius norm, which we estimate in each training step using a Monte Carlo sampling approach.

We perform extensive experiments, testing various combinations of hyperparameters for our three proposed interventions and acquire best results when combining them. The explanations of networks trained using our proposed methods are consequently more robust against random input perturbations and targeted adversarial attacks.

As described in the introduction to this chapter, we can gain additional, more specific insights into the decision process of the machine learning model by studying counterfactual explanations besides attribution methods. However, interpretable counterfactuals can be hard to find and attempts can result in adversarial examples.

In the remaining two papers included in this thesis, we research the similarities and differences between counterfactuals and adversarial examples and provide an effective method to generate counterfactual explanations.

To this end, we leverage the recent advances by generative models to accurately approximate the data manifolds of natural images.

In [39], we show how to use normalizing flows to create high-quality counterfactuals that are classified as a target class with great confidence. We use the fact that the flow induces a coordinate transformation more suitable for finding semantically meaningful counterfactuals. We demonstrate theoretically how applying gradient ascent in the base space of a well-trained flow corresponds to walking along the data manifold while gradient ascent in image space quickly leads off the data manifold and thus results in adversarial examples. Experiments on three different data sets confirm that our counterfactuals exhibit structural changes and look like naturally occurring images from the target class.

In [40], we refine our theoretical analysis and extend it to non-bijective generative models such as variational autoencoders and generative adversarial networks. These models come with fewer theoretical guarantees and might lead to some information loss but can scale to very high-dimensional data. Furthermore, we considerably broaden our experiments to include a regression task and high-resolution images. Our modular approach is easy to apply and works well with independently trained classifiers and generative models.

These contributions represent significant progress towards robust attribution methods and interpretable, semantically meaningful counterfactuals. Differential geometry is crucial in this research as it provides theoretical guarantees for our claims and equips us with the necessary insights to find practical solutions to the current challenges in explainable artificial intelligence.

1.1.1. Publications

The contents of this thesis have been published (or are about to be published) at ML conferences and journals in the form of four peer-reviewed papers listed below. A detailed description of the contributions of each author is added below the respective publication.

- **Explanations can be manipulated and geometry is to blame**
  by Ann-Kathrin Dombrowski, Maximillian Alber, Christopher J. Anders, Marcel Ackermann, Klaus-Robert Müller and Pan Kessel, published in Advances in

Neural Information Processing Systems (NeurIPS) 2019, (poster)

PK had the initial idea. AKD and PK further developed the project with input from all authors. Experiments were planned, conducted and evaluated for the most part by AKD with contibutions from PK, CJA, and MA. PK performed the mathematical proofs. PK and AKD wrote the paper. KRM and CJA were involved in paper revisions.

- **Towards robust explanations for deep neural networks**
  by Ann-Kathrin Dombrowski, Christopher J. Anders, Klaus-Rober Müller and Pan Kessel, published in Pattern Recognition Journal, 2022

  AKD and PK developed the paper idea. AKD planned, performed and evaluated experiments. CJA helped with initial code development. PK performed mathematical proofs. PK and AKD wrote the paper with revisions from KRM.

- **Diffeomorphic explanations with normalizing flows**
  by Ann-Kathrin Dombrowski*, Jan E. Gerken* and Pan Kessel, published at the Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models, ICML 2021, (contributed talk)

  PK had the initial idea. All authors were involved in further developing this idea. AKD planned, performed and evaluated experiments. PK and JEG performed mathematical proofs. All authors were involved in writing the paper.

- **Diffeomorphic counterfactuals with generative models**
  by Ann-Kathrin Dombrowski*, Jan E. Gerken*, Klaus-Rober Müller and Pan Kessel, currently under review at IEEE Pattern Analysis and Machine Intelligence Journal, 2022

  All authors were involved in developing the paper idea. AKD planned, performed and evaluated experiments. PK and JEG performed mathematical proofs with contributions from AKD. All authors were involved in writing the paper.

Much of this thesis's text and graphics were directly adopted from these papers, with permission from all co-authors. Clarifications and additional experimental results were added where appropriate and the related work sections were updated to include the most recent relevant literature. As the explainability of deep neural networks and the use of tools from differential geometry for the theoretical analyses are core to all our contibutions, the topic-specific introductions for each paper concerning these two areas were unified and extended to serve as two introductory chapters on explainable AI (see Chapter 2) and differential geometry (see Chapter 3), respectively.

Additional papers by the author that are not included in this thesis are listed below.

- **Fairwashing explanations with off-manifold detergent**
  by Christopher J. Anders, Plamen Pasliev, Ann-Kathrin Dombrowski and

* equal contribution

Klaus-Robert Müller, published at the International Conference on Machine Learning (ICML) 2020

- **Automated dissipation control for turbulence simulation with shell models**
  by Ann-Kathrin Dombrowski, Klaus-Robert Müller and Wolf-Christian Müller, ArXiv preprint, 2021

1.2. Structure of the thesis

The thesis is structured into seven parts. Chapter 1 (**Introduction**) serves as a preface describing organization and contributions of this thesis.

Chapter 2 (**ML background**) provides a general introduction to explainable artificial intelligence, covering the goals of explanation methods and recent developments in the field. We then give an overview of the different types of explanation methods. This is followed by more specific introductions to the two subfields of explainability on which this thesis focuses: attribution methods and counterfactual explanations. We present several popular attribution methods and the motivation behind them. For counterfactual explanations, we describe the general approach and delve into connected concepts such as contrastive explanations, adversarial examples and feature visualization. The chapter furthermore gives an overview of the image data sets and similarity measures used throughout this thesis.

This is followed by Chapter 3 (**Mathematical background**), where we give an introduction to the fundamental concepts of differential geometry, which we require for theoretical derivations in later chapters. We start by reviewing the essential basics of linear algebra before discussing more advanced topics of differential geometry such as manifolds, the Riemannian metric, the Weingarten map, pushforward and pullback.

The thesis's central part is divided into three chapters based on four previously published papers (see Section 1.1).

Chapter 4 (**Manipulating explanations**) shows that many popular explanation methods can be arbitrarily manipulated. We introduce an algorithm that perturbs the input with imperceptible noise so that the classifier's prediction is unchanged, but the explanation map reproduces an arbitrary target map. We analyse this unexpected behavior theoretically and derive an upper bound on the maximal change in the explanation map. Based on this upper bound, we propose β-smoothing, a novel explanation method that is more robust against adversarial perturbations. Our theoretical results are backed by experiments on various gradient- and propagation-based explanation methods, different network architectures and two data sets.

We then extend our analysis of explanation robustness and investigate how to incorporate robustness measures into the training regime in Chapter 5 (**Towards**

networks with robust explanations). We expand on the theoretical results of the previous chapter, generalizing the upper bound on the maximal change in the explanation map. We then derive three different effective regularizers that can be incorporated into the training processes of deep neural networks and provably lead to more robust explanations. We test our regularizers thoroughly for random input perturbations, training a vast number of neural networks with different configurations. We achieve the best results when combining our proposed regularizers. Furthermore, we show that our methods generalize to protection against adversarial attacks on the explanation and are applicable to different network architectures, data sets, and explanation methods.

While Chapter 4 and 5 focus on attribution methods, Chapter 6 (**Counterfactual explanations**) discusses counterfactual explanations. As explained above, the search process for counterfactuals can sometimes result in adversarial examples. We establish a rigorous theoretical framework for applying gradient ascent in a more suitable coordinate system induced by the latent space of a generative model. More specifically, we prove that, for well-trained generative models, the induced metric effectively rescales the gradient in image space so that directions perpendicular to the data manifold are suppressed, and only gradient directions along the data manifold remain. Update steps then move along the manifold and ultimately lead to structural changes in the input, which are easy to interpret for a human. Depending on the bijectivity of the generative model, we term the resulting explanations diffeomorphic and approximately diffeomorphic counterfactuals. We apply our algorithm to normalizing flows on four image data sets, spanning classification, and regression tasks. We achieve visually striking results and conduct a thorough quantitative analysis. Furthermore, we show how our algorithm can scale to high-dimensional data sets by leveraging variational autoencoders and generative adversarial networks. Finally, we discuss the advantages and drawbacks of the different generative models used.

To complete the thesis, Chapter 7 (**Conclusion and outlook**) provides a unified conclusion of our work, summarizing the different contributions presented in this dissertation, and discusses future research directions.

2. ML background

This chapter serves as an introduction to the machine learning (ML) topics covered in this thesis. We start with an overview of explainable artificial intelligence (XAI) followed by more specific introductions to the subfields of attribution methods and counterfactual explanations.

Furthermore we present the image data sets and similarity measures used throughout this thesis.

2.1. Explainable AI

In recent years, deep neural networks have revolutionized many different areas in research, medicine and industry, and triggered a renaissance of artificial intelligence (AI), and machine learning (ML) in particular. The main drivers of this success were the increased complexity—and therefore expressiveness—of these ML models together with efficient learning algorithms for deep learning [41, 42], the collection of large data sets [43], and the development of hardware [44, 45] and software [46, 47] that make the training scalable.

Despite their impressive performance, the reasoning behind the decision processes of deep neural networks remains difficult to grasp for humans. This opaqueness can limit their usefulness in applications that require transparency. Transparency of the decision process can be relevant for fostering trust in the system, confirming that the system is working as intended, or identifying potential problems that would not be apparent when looking merely at the model's outputs. It can also enable us to gain new insights into the application domain or deepen our understanding of how neural networks work in general.

Consequently, the field of eXplainable AI (XAI) [11, 48] emerged, aiming to open the black box of deep neural networks by making their decision processes more interpretable. However, defining what it means for a model to be interpretable is challenging. As an intuition, we can use the (non-mathematical) definition by Miller et al. [49]:

"Interpretability is the degree
to which a human can understand the cause of a decision.".

To that end, researchers have developed a vast and diverse collection of explanation methods. Explanation methods have received considerable interest from the

international ML communities, practitioners, researchers and policy makers, as interpretability is crucial for debugging, scientific understanding, and safety critical applications, such as medicine, autonomic driving, or security. Development and evaluation [50, 51] of explanation methods as well as refinement of what we hope to gain from them [52] remain active areas of research.

Many recent surveys [53–60] outline modern approaches to explainable AI in great detail. We briefly introduce the most commonly discussed aspects in this section.

Goals and purpose of XAI Explanation methods provide an interface between the human and the machine learning model. XAI aims to give accurate explanations in a human understandable manner. Therefore, explanations may look very different depending on their intended audience. A developer might want to use XAI to debug and improve their ML model. People affected by automated model decisions, like loan applicants, on the other hand, might want to understand their situation and which aspects they need to change to reach a more desirable outcome. Policy makers and regulatory entities need XAI to certify a model and ensure its decision processes are lawful, i.e., not based on discriminatory features. Finally, other users and domain experts, like medical doctors and analysts, intend to apply methods of XAI to find causal connections to further scientific knowledge. Improving model interpretability via XAI is thus an instrumental goal to achieve the improvement and expansive application of trusted, safe, fair, and accurate ML models that facilitate our daily lives.

Recent developments and applications of XAI With the introduction of various explanation methods and libraries that allow easy implementation [61–64], the application of XAI methods in various areas has seen a remarkable increase.

XAI facilitates the debugging and subsequent improvement of ML models in research settings as explanation methods can help detect biases or other flaws in the data collection and pre-processing steps that result in the ML model basing its predictions on undesired features [15, 30, 65]. Furthermore, explanation methods can be used to guide the training process [66] or prune a network after training [67].

Explanation methods show great results in medical applications [68–70]. Especially in diagnostics [71], explanations are crucial for doctors and patients alike when relying on the predictions of machine learning systems.

Another exciting field for XAI applications is finance [72–75] where one can use explanation methods to gain insights into systems from market forecasting and investment strategies to credit scoring and fraud detection.

Political debates about applications of ML systems and implications concerning fairness [32, 76, 77] further increase the relevance of XAI. For example, the European Union introduced a right to explanation in the General Data Protection Right

(GDPR) [31, 78–81]. Therefore, explainability is relevant for automated decision-making in insurance, hiring, loan assessment and predictive policing [82].

XAI is also becoming influential in industrial applications [83] as explanation methods can be used for failure diagnosis [84, 85] and predictive maintenance [86]. This trend is likely to grow further since machine learning systems offer valuable improvements but must meet many requirements to ensure the stability and robustness necessary for industrial production processes [87].

2.1.1. Explanation methods

Interpretable models vs post-hoc explainability Some models are interpretable by design, meaning elaborate methods from XAI are not necessarily needed to understand their decision-making processes. Linear and logistic regression fall into this category. Both base their prediction on a weighted sum of the feature inputs. This linear relationship makes them interpretable. Extensions to these models include generalized linear models (GLMs) [88, 89] and generalized additive models (GAMs) [90].

Another example of a transparent model is the decision tree, which uses a hierarchical structure of if/else statements to split the data repeatedly according to learned cutoff values in the features [91]. Other rule-based machine learning systems [92–94] are also considered interpretable by construction since humans easily understand the learned rules.

Furthermore, there is the Naive Bayes classifier [95], which produces probabilities for a class depending on the value of the features, assuming their conditional independence. This naive assumption makes the model interpretable but also limited.

An interpretable non-parametric approach is the k-nearest neighbour (kNN) [96] method, which uses the closest neighbouring training data points of a given data point for classification or regression. These exemplary instances from the data set can provide insights if the samples do not have too many variables.

In summary, the above models are usually considered interpretable since they relate input features and predictions relatively simply. If the input features grow numerous or convoluted, we might require additional support from XAI methods or more traditional means of visualization to understand even these simple models.

On the other hand, we have ML models that are inherently non-linear and lack explicit rules to quantify their decision processes. Such models are sometimes referred to as opaque models or black boxes. Deep neural networks fall into this category. To make their decision processes comprehensible, one needs to apply XAI methods post-hoc, that is, after training the model.

Types of explanations Explanations come in many forms. Attribution or feature importance methods assign a score to each input feature and thus usually produce a relevance score map of the same dimensionality as the input. We discuss some variants of attribution methods in Section 2.1.2.

Counterfactual explanations produce alternative inputs that look similar to the original query input but are assigned a different prediction. We give a detailed introduction to counterfactual explanations in Section 2.1.3.

Other approaches rely on prototypes or other instances from the training data [97,98].

Furthermore, there are concept-based explanations [99–102] that can take quite different forms than the original model inputs. For example, textual explanations can be given for image classifiers.

Local vs. global explanations Global explainability provides insights into global model behavior based on a holistic view of input features, model structure, and model parameters [103], i.e., how features contribute to a model's decision on average.

An example for a global explanation method is the partial dependence plot (PDP) [104, 105]. It averages over individual conditional expectation (ICE) plots which in turn show how the prediction changes when the feature of interest is changed, assuming independent features.

To measure interdependence between input features concerning the models' prediction, the Friedman's H-statistic [106] was introduced. Accumulated local effects (ALE) [107] take another approach to deal with correlation between features, calculating differences in predictions, based on the conditional distribution of the features, instead of averages.

It is also possible to train an interpretable surrogate model which mimics the behavior of the more complex opaque model. Furthermore, the Maximum Mean Discrepancy (MMD)-critic [97] finds prototypes from high-density areas of the data distribution and criticism from regions that are not well explained by the former, which are then evaluated together with their model predictions.

Another quite different approach that does not depend on individual inputs or predictions is feature visualization [108, 109], usually used for convolutional neural networks. This approach finds inputs that maximize the activations of a single (hidden-layer) neuron in a network by iteratively updating an input starting from random noise. We provide an example and discuss the connection to counterfactual explanations in Section 2.1.3.

Furthermore, there are approaches with a strong focus on the training data. For example, one can learn about model behavior by identifying influential training instances [110, 111] via deletion diagnostics and influence functions. An influential instance is a data point whose removal strongly affects the behavior of the retrained

model. The approach can also be applied to parameters or predictions of machine learning models.

Yet another approach are concept-based explanations [99, 101, 112]. A concept is any form of abstraction that helps the user understand the ML model and does not necessarily depend on the training process for a specific task. For example "stripes" can be a human-defined concept that might help to understand differences in the classification of pictures of horses and zebras. To understand global model behavior, one can then measure the influence of a concept on the prediction for a certain class.

The vastness of the input space, feature interdependence, and highly complex non-linear models can make global interpretability hard to achieve or computationally expensive. In many cases, practitioners are primarily interested in model behavior for a particular instance, which is why a large part of explainability focuses on local explanations which aim to explain predictions for individual inputs. The remainder of this section will discuss such local explanations.

White-box and black-box explanations One can categorize explanations depending on how much access to the model is required. Model agnostic explanations treat the model as a black box and only require access to a model's decision. This makes them readily applicable to any model and thus very flexible.

Some model agnostic approaches rely on perturbation or partial occlusion of the input and the resulting change in the prediction [113–116].

Occlusion-based approaches "remove" input regions by setting them to zero and then evaluate the model on the modified input. If the model output for the partially occluded input differs considerably from the original model output, the occluded region is deemed relevant. Gradually moving the occluded region can determine a relevance map for the complete input. Usually, regions (rather than individual dimensions) are set to zero, which means the resulting explanation maps are often coarser than the input. One can consider occlusion techniques as a particular case of input perturbation. Other input perturbations can include blurring or adding random noise.

One popular perturbation-based method is the Local Interpretable Model-Agnostic Explanation (LIME) [65]. LIME trains a simple interpretable model on perturbed instances of the query input so that the prediction of the simple model locally matches the prediction of the more complex model whose behavior we want to understand. If there are many input features, it can be advantageous to cluster them into regions (for example, superpixels for images), which are then jointly perturbed. An additional perturbation-based approach is the anchors method [117], which uses reinforcement learning techniques to distill features that are almost exclusively responsible for the predicted outcome.

Another well-known technique called SHAP (SHapley Additive exPlanations) [22] is

based on a concept from game theory, the Shapley value [118], which is the average marginal contribution of a selected input feature across all possible combinations.

In contrast to black box or model agnostic explanations, white box explanations require full access to the model and are often faster to compute. Those methods usually fall under the category of feature attribution methods and are applied in the context of image classification. For example Layerwise Relevance Propagation (LRP) [21, 26] is an explanation method for which specific propagation rules need to be implemented for hidden layers, for Pattern Attribution [119] one requires access to the networks weights to learn specific patterns, Guided Backpropagation [120], DeepLIFT [25], and DeconvNet [113] need access to individual activations of hidden units, and Grad-CAM [121] uses the gradient with respect to the last convolutional layer.

Explanations beyond attribution methods, for example, concept-based explanations [99–102, 112, 122] may also require access to the hidden layers of a neural network.

Other methods lie somewhere between model agnostic and white box explanations. They do not require access to the complete model structure but are based on the gradient of a prediction with respect to the input. Examples in this category include explanation methods like Saliency and Gradient maps [23–25], SmoothGrad [28], and Integrated Gradients [27]. Some methods [123] iteratively find regions relevant for the classification or train an additional model to localize important features [124]. The gradient is also needed for some counterfactual explanation methods [39, 125, 126].

In the remainder of this thesis, we focus on post-hoc, local, white-box explanations for neural networks trained on image data.

2.1.2. Attribution methods

In Chapter 4 and 5 we investigate the robustness of some of the most common attribution methods, which we introduce in this section. Example images with explanation maps for all considered methods are shown in Figure 2.1.

To define the explanation maps we consider a neural network $f : \mathbb{R}^N \to \mathbb{R}^K$ with ReLU non-linearities which classifies an image $x \in \mathbb{R}^N$ into one of K categories with the predicted class given by $k = \arg\max_i f(x)_i$. The explanation map is denoted by $h : \mathbb{R}^N \to \mathbb{R}^N$ and associates an image with a vector of the same dimension whose components encode the relevance score of each pixel for the neural network's prediction.

Gradient The Gradient explanation or Saliency map [23, 24] is the most basic explanation. It quantifies how infinitesimal perturbations in each pixel change the prediction $f(x)$. To that end, the gradient of the class of interest (usually the class of

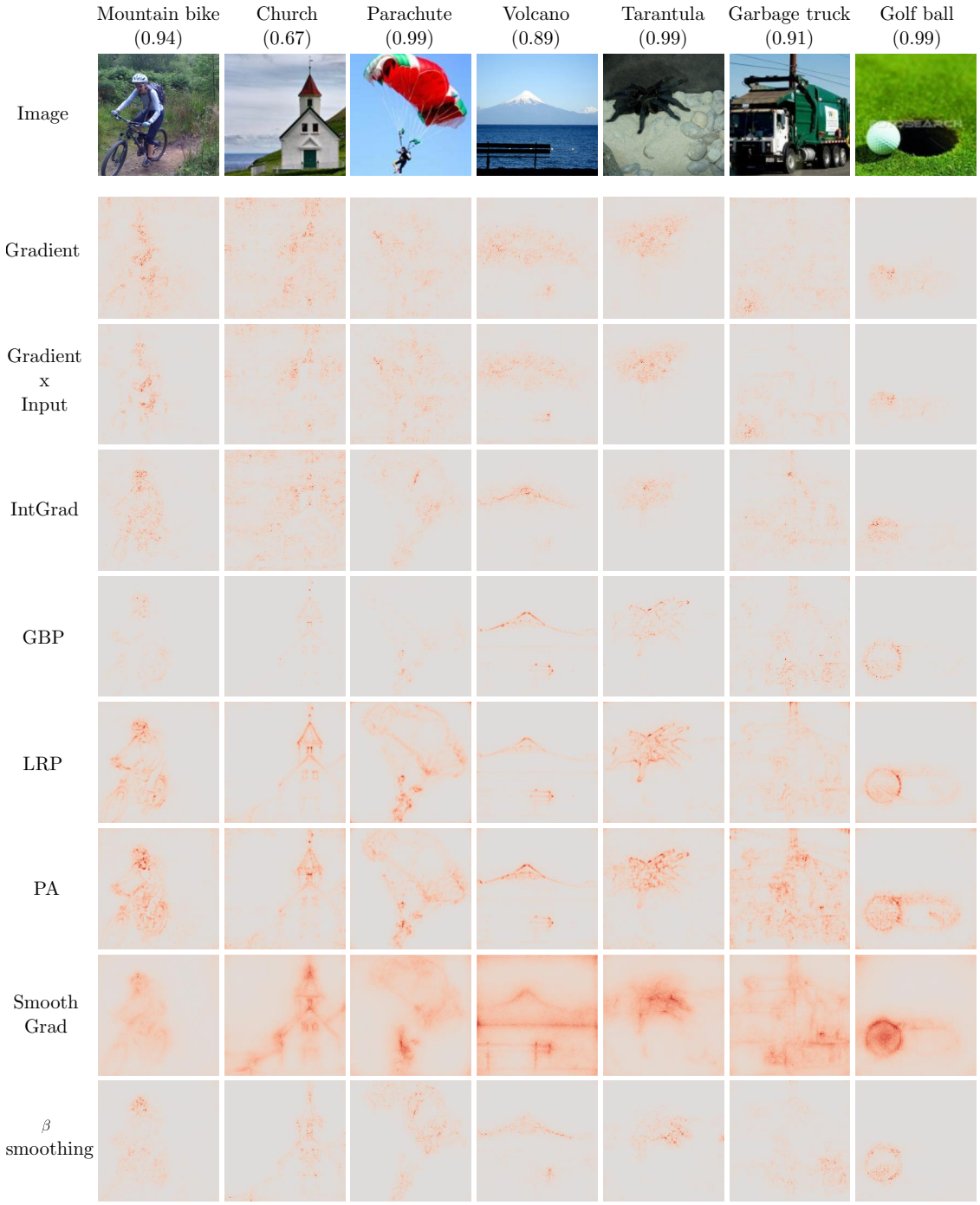


Fig. 2.1.: Explanation maps for different methods applied to a pre-trained VGG16 network. The first row shows the input images (from ImageNet [43]) with class and prediction confidence.

highest probability k) is calculated with respect to the input pixels. The explanation map is then defined as

$$h(x) = \frac{\partial f_k}{\partial x}(x). \tag{2.1}$$

Gradient $\times$ Input The explanation given by Gradient $\times$ Input [25] is defined as the element-wise multiplication of the gradient of the neural network with the respective input image. The explanation map is given by

$$h(x) = x \odot \frac{\partial f}{\partial x}(x)\,. \tag{2.2}$$

For linear models, this measure gives the exact contribution of each pixel to the prediction.

Integrated Gradients (IntGrad) Integrated Gradients [27] is based on two fundamental axioms: sensitivity and implementation invariance. Sensitivity means that if input and baseline differ in one feature and have different prediction, the differing feature should have a non-zero attribution. Implementation invariance means that if two networks are functionally equivalent (the outputs are equivalent for all inputs), the attributions should also be identical.

The explanation map is defined as

$$h(x) = (x - \bar{x}) \odot \int_0^1 \frac{\partial f(\bar{x} + t(x - \bar{x}))}{\partial x} \mathrm{d}t\,, \tag{2.3}$$

where $\bar{x}$ is a suitable baseline. The integration over the difference between the baseline and the image avoids problems with local gradients being saturated. The baseline aims to represent the absence of features. For our purposes we use the zero baseline, as proposed in the original paper. Using any constant color baseline can be problematic since pixels that have the same color as the baseline will always have an attribution of zero although they might belong to the object of interest [114, 127, 128]. Nonetheless, as we focus on analysing robustness of attribution methods, this simple baseline is sufficient for our purposes.

Guided Backpropagation (GBP) This method is a variation of the gradient explanation for which negative components of the gradient are set to zero while backpropagating through the non-linearities [120]. The motivation for this procedure is that explanation maps achieved by pure backpropagation [23, 24] or by *deconvolution* [113] do not produce sharp, recognizable image structures for higher layers in a neural network. By only allowing positive values to be backpropagated, guided backpropagation reduces the influence of neurons which decrease the activation of the higher layer unit for which we desire an explanation. The resulting explanation

map is then sharper and more focused on features that contributed positively to the classification. For the output layer, the relevance, i.e. how much a feature contributes to the prediction, is defined by

$$R_i^L = \delta_{i,k} f(x)_i \,, \tag{2.4}$$

where we use the Kronecker symbol

$$\delta_{i,k} = \begin{cases} 1, & \text{for } i = k \\ 0, & \text{for } i \neq k \end{cases} . \tag{2.5}$$

The relevance for a convolutional or dense layer l and neuron i can then be computed as

$$R_i^l = \sum_j W_{ji}^l R_j^{l+1} \tag{2.6}$$

while the rule for propagating through a ReLU activation unit is defined as

$$R_i^l = \begin{cases} R_i^{l+1}, & \text{if } R_i^{l+1} > 0 \text{ and } x_i^{l+1} > 0 \\ 0, & \text{otherwise} \end{cases} . \tag{2.7}$$

Layer-wise Relevance Propagation (LRP) Layer-wise Relevance Propagation [26] propagates the relevance backwards through the network. The approach is related to the layer-wise application of the Taylor series [21], for which different choices of root points correspond to different relevance propagation rules of LRP.

For the output layer, relevance is defined by

$$R_i^L = \delta_{i,k} \,. \tag{2.8}$$

The relevance of the final layer is then propagated backwards through all layers using specific propagation rules defined in [129]. For our purposes we use the z^+ rule for intermediate layers

$$R_i^l = \sum_j \frac{x_i^l (W^l)_{ji}^+}{\sum_i x_i^l (W^l)_{ji}^+} R_j^{l+1} \,, \tag{2.9}$$

where $(W^l)^+$ denotes the positive weights of the l-th layer and x^l is the activation vector of the l-th layer[1]. For the first layer, we use the $z^{\mathcal{B}}$ rule to account for the bounded input domain

$$R_i^0 = \sum_j \frac{x_j^0 W_{ji}^0 - l_j (W^0)_{ji}^+ - u_j (W^0)_{ji}^-}{\sum_i (x_j^0 W_{ji}^0 - l_j (W^0)_{ji}^+ - u_j (W^0)_{ji}^-)} R_j^1 \,, \tag{2.10}$$

where l_j and u_j are the lower and upper bounds of the input domain respectively.

[1] Note that the z^+ rule is only defined for positive inputs and ignores biases. We only apply it to simple CNNs. For more generally applicable rules see LRP-$\alpha\beta$ [26].

PatternAttribution (PA) This explanation method [119] was developed starting from analysing the behavior of linear models on noisy data. More specifically the authors construct an input $x = s + d$ by adding a *signal* s and a *distractor* d component together. The optimal model completely filters out the distracting noise and bases the prediction solely on the signal. For a linear model this means that the weight vector will be perpendicular to the noise and thus will not contain much information about the signal by itself [130].

The authors aim to filter out the noise component of the data so that only the signal remains (PatternNet) or relevance is only attributed to the signal (PatternAttribution). An optimal signal would be a minimal selection of input features which correlate with the output in the same way as the input itself. The authors then propose a two-component signal estimator, which recovers the signal optimally in the linear case and depends on learned patterns $A^l_{+,ij}$ and $A^l_{-,ij}$. The patterns A_+ (A_- analogously) for layer l are defined as

$$A^l_{+,ij} = \frac{\mathbb{E}_+[x^l_j y^l_i] - \mathbb{E}_+[x^l_j]\mathbb{E}[y^l_i]}{\sum_k \left(W^l_{ik}\mathbb{E}_+[x^l_k y^l_i] - W^l_{ik}\mathbb{E}_+[x^l_k]\mathbb{E}[y^l_i]\right)}\,, \tag{2.11}$$

where $\mathbb{E}_+[\bullet]$ is the expected value of the argument within the positive regime (negative values are set to zero), x^l is the input, W^l the respective weight matrix and y^l is the output (pre-activation).

PatternAttribution is then analogous to standard backpropagation upon element-wise multiplication of the weights of each neuron with the respective patterns A^l_+.

These patterns must be learned based on training data, after training of the neural network and before the explanation method can be applied.

SmoothGrad The idea behind SmoothGrad [28] is to remove the noise from the Gradient explanation by averaging over multiple Gradient explanations of slightly perturbed inputs. Therefore, SmoothGrad is not a stand-alone explanation method but is computed as the average

$$h(x) = \frac{1}{N}\sum_i^N \tilde{h}(x + \epsilon_i)\,, \tag{2.12}$$

where N is the number of samples over which we average, $\tilde{h}$ is the original explanation map and $\epsilon_i \sim \mathcal{N}(0, \sigma)$ is a noise vector sampled from the Gaussian distribution. The number of samples N is choosen between 10 and 50. The standard deviation σ is chosen dependent on the desired noise level $\nu = \frac{\sigma}{x_{\max} - x_{\min}}$, which depends on the maximum and minimum values of the input and is usually chosen to be between 10% and 20%. SmoothGrad can theoretically be applied to any explanation map $\tilde{h}$ but is most prominently used in combination with the gradient ∇f, which we also show in Figure 2.1.

β-smoothing The β-smoothing explanation method [37] was developed within the scope of this thesis. The idea is to replace the ReLU activation functions with softplus activations and then take the gradient of the original classification wrt the input using the modified network. The β parameter of the softplus activation controls how closely the ReLU is approximated and is a hyperparameter of the method. We find the value $\beta = 1$ works well in practice. Just as SmoothGrad, β-smoothing can be applied to various explanation methods. For the comparison in Figure 2.1 we apply it to the standard gradient. We discuss the motivation for β-smoothing and the relation to SmoothGrad in more detail in Chapter 4.

These presented methods cover two classes of attribution methods, namely *gradient-based* and *propagation-based* explanations, and are frequently used in practice [29, 128, 131]. To obtain a pixelwise relevance score, we sum over absolute values of the three color channels so that the explanation map only has one channel where large values indicate high relevance of the corresponding pixel (see Figure 2.1). To compare explanations with each other we normalize the explanation to have $\sum_i |h(x)_i| = 1$.

2.1.3. Counterfactual explanations

In Chapter 6 we analyse approaches which leverage generative models to find counterfactual explanations. This section serves as an introduction to counterfactual explanations in a broader sense.

Attribution-based explanation methods focus on the question "Why was this input classified as A?". While this approach seems intuitive, some studies show that humans prefer *counterfactual* explanations which aim to answer questions like "Why was this input classified as A and not as B" or "What would need to change in the input so that it is no longer classified as A but instead as B?" [34, 49, 132]. A counterfactual is similar to the original input but has a different prediction. Often we require the prediction to change in a significant way, for example, to cause a change in the predicted class. In addition to that, counterfactuals are sometimes required to be minimal, i.e. to describe the smallest change in the input feature values that result in the desired change of the prediction.

Counterfactuals are very easy to interpret by humans since they stand in contrast to the original input and usually focus on a limited number of input features. The following examples highlight their usefulness.

Example 1 Bob applies for a loan from a bank but is rejected by the automated ML system. He is interested in knowing what he needs to change in order to improve his chances to get a loan. One possible answer would be: if Bob was 5 years younger, he would get the loan. This is a counterfactual, though not a very actionable one since Bob cannot age backwards. Another possible counterfactual could be: if he

earned 5000€ more per year, or if he had held his current job for one more year, then he would get the loan. These counterfactuals are actionable since Bob can ask for a raise or wait a year and then apply for the loan again.

Example 2 Alice is a doctor that does breast cancer screenings. The mammogram of one of her patients is classified as clear from breast cancer by her ML assistance system. However she notices that the probability of pathology has increased compared to past mammograms. She runs the counterfactual XAI system and compares the counterfactual to the original mammogram. One specific tissue area changed in the counterfactual showing a small tumor. After a second glance Alice notices that the same area also looks suspicious on the original mammogram and decides to run additional tests.

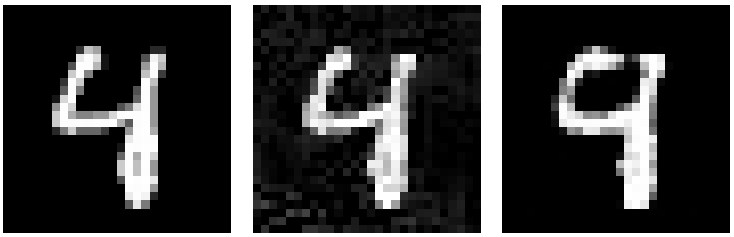
Connection to contrastive explanations and prototypes Counterfactuals are related to contrastive explations and the terms are sometimes used interchangeably. A contrastive explanation is a data point from the data set that has a different classification to the query sample. Thus contrastive explanations are related to prototypes which are usually single points from the data set or averaged representations of several points. One difference is that counterfactuals are generally not defined as existing samples from the data set but regarded as hypothetical alternatives, hence the name *counterfactuals*. They are similar to the original data point but differ in distinct features so that the confidence with respect to a certain classification is different.

Connection to adversarial examples Counterfactuals for tabular or numerical data [32, 133, 134] can often be found relatively easily by optimizing the input data to yield the desired classification while optionally enforcing constraints on features to make the explanation actionable [31].

In contrast, applying this simple optimization to high-dimensional data such as images usually leads to adversarial examples[2]. Adversarial examples are inputs that, to a human observer, look (approximately) identical to the original input but result in a drastically different classification [36]. In particular, adversarial examples are usually unlikely in the underlying data distribution. Figure 2.2 shows an image from the MNIST data set together with an adversarial example and a counterfactual, both generated by maximizing the target class "nine".

The relation between counterfactuals and adversarial examples has been pointed out repeatedly [19, 137, 138] but a clear distinction on all data sets is not possible since the data distribution is often not known precisely enough.

[2]Recent work [135, 136] shows that for adversarially trained models, simple optimization leads to less noisy "counterfactuals" than for standard models.



original adversarial example counterfactual

Fig. 2.2.: Example from the MNIST data set. The original is classified as "four". The adversarial example and the counterfactual look similar to the original but are both classified as "nine" with high confidence. While the counterfactual looks like an image from the training data set, the adversarial example contains noise that is untypical for the data distribution.

However, the intended use of counterfactuals and adversarial examples is very different: counterfactuals are meant to provide an explanation, while adversarials are meant to trick a neural network into misclassifying. Consequently, counterfactuals aim to be human interpretable and adversarials are usually constructed to be indistinguishable from the original for a human. This leads to different restrictions during their creation: Counterfactuals are often constrained to lie on the data manifold, i.e. they look like naturally occurring samples from the data distribution while adversarials include specifically engineered perturbations that do not occur in a natural setting but lead to high activation of some neurons in the network. One can regard counterfactuals as *natural* adversarial examples [138] that lie on the data manifold and thus "generalize to human agents". Or one can regard adversarial examples as counterfactuals that lie "off data manifold" and are indistinguishable from the original to a human observer.

To make sure that the optimization process results in a counterfactual, one introduces further constraints, so that the modified data point lies on the data manifold. We will look into this in greater detail in Chapter 6.

Connection to feature visualization Feature visualization [108,139] tries to solve a similar problem to counterfactual explanations, namely to change an input iteratively by maximizing the activation of a specific neuron. In the case of counterfactuals, we maximize the activation of the target output neuron (up to a desired confidence).

Feature visualization faces the same problem as counterfactual generation when it comes to optimization of high-dimensional inputs: we end up with some high frequency pattern that seems nonsensical to humans but produces a significantly increased activation in the target neuron, which is highly related to the concept of adversarial examples [36, 140].

To avoid these high frequency structures, various degrees of regularization are introduced, from frequency penalization to incorporating generative models [109, 141–143]. Similar efforts have also been made in the context of counterfactual

explanations [40, 125].

One fundamental difference between counterfactual generation and feature visualization is that feature visualization is intended to capture global model behavior. This means, the visualized features are not specific to one particular data input, but are generated from (several) random noise starting points.

We show an example of visualized features in Figure 2.3. The image stems from maximizing the pre-activation of a hidden neuron from the GoogLeNet architecture [144], which was trained on ImageNet.

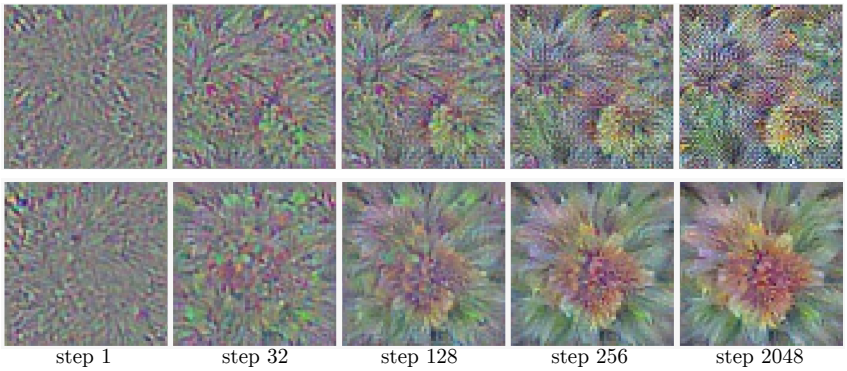


Fig. 2.3.: Visualizing features by maximizing a hidden neuron in a neural network (image source: [139]). Unregularized optimization (first row) produces high frequency artifacts while regularized optimization (second row) leads to smoother and more natural looking images.

2.2. Image data sets

Throughout this thesis we use images from various image data sets to test our algorithms. In this section we briefly present the data sets that we use and the pre-processing steps that we perform before applying our algorithms. Figure 2.4 shows four pre-processed example images for each data set.

MNIST The MNIST [145] data set from the Modified National Institute of Standards and Technology is a large image data set of hand written digits from zero to nine. The database contains 70k grayscale images of resolution 28×28 pixels. Modern neural networks usually achieve test accuracies above 99% after training.

CIFAR10 The CIFAR10 data set [146] from the Canadian Institute For Advanced Research contains 60k color images of ten different classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The images have a resolution of 32×32 pixels. Modern neural networks usually achieve test accuracies above 95% after training.

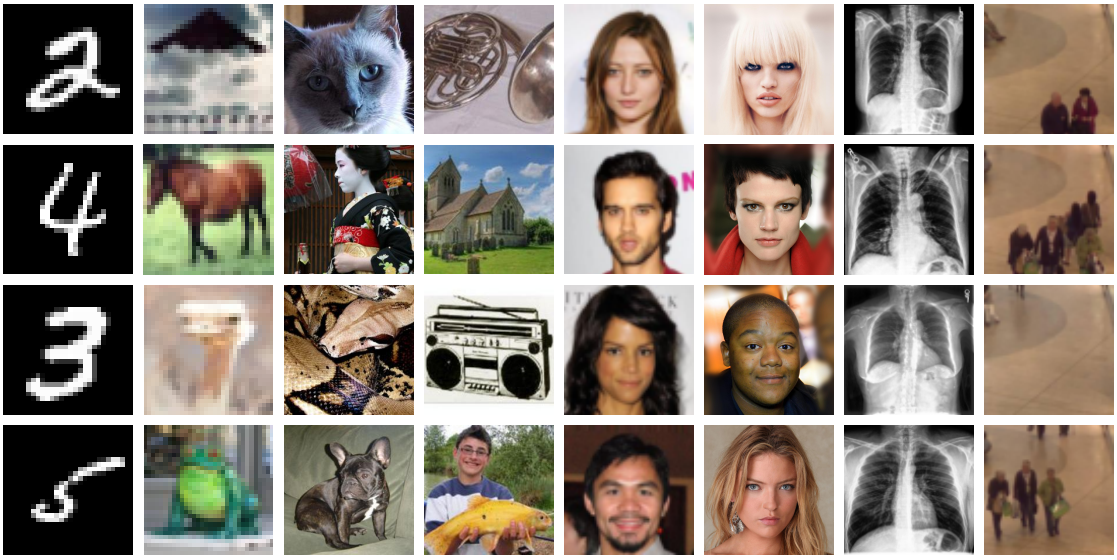


Fig. 2.4.: From left to right: columns show pre-processed images for data sets: MNIST, CIFAR10, ImageNet, ImageNette, CelebA, CelebA-HQ, CheXpert and Mall. Note that the images are rescaled for presentation purposes but do have very different resolution, which is why some images appear pixelated.

ImageNet ImageNet [43] is a large database of high resolution images that is continuously expanded. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [147] focuses on approximately 1.3 million color images, cropped to a resolution of 224×224 pixels, that are divided into 1000 object categories. We use the same resolution in this thesis. Modern neural networks usually achieve test accuracies of around 80-90% on this standardized benchmark.

ImageNette ImageNette [148] contains a subset of ImageNet pictures that are divided into ten easily distinguishable classes (tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, and parachute). We rescale and crop the images to a resolution of 224×224 pixels. Modern neural networks usually achieve test accuracies of around 85-95%.

CelebFaces Attributes Dataset (CelebA) CelebA [149] is a data set of over 200k celebrity images with 40 binary, non-exclusive attributes per image, which contain information about the hair color, eyeglasses, make-up, or age of a person. For the purposes of this thesis, we only use the blond attribute and rescale and crop images to a resolution of 64×64 pixels.

CelebA-HQ CelebA-HQ [150] is a selection of 30k high-quality images of resolution 1024×1024 from the original CelebA data set. Karras et al. [150] apply several

preprocessing steps to ensure consistent quality like the removal of artifacts and centering on the facial region. We then use the CelebA-HQ images without further pre-processing.

CheXpert CheXpert [151] is a large data set of chest X-rays containing more than 224k chest radiographs of over 65k patients. The data set contains frontal and lateral images. We only use frontal images, of which there are around 191k. The grayscale images have 14 labeled observations of different pathologies. In this thesis, we focus on the cardiomegaly (enlarged heart) attribute. We rescale and crop the images to a resolution of 128×128 pixels.

Mall The Mall data set [152] consists of 2000 video frames, collected from a publicly accessible web cam in a shopping mall. In all frames the head positions of the pedestrians are annotated. The color images have a resolution of 320×240 pixels. For the purpose of this thesis we resize the images so that the shortest side has 128 pixels. We then take a 64×64 pixel cutout (starting from pixel $[r = 64, c = 100]$) and use these cropped images for our experiments.

2.3. Similarity metrics

Throughout this thesis, we assess the similarity between images and between explanation maps using different similarity metrics, which we describe in this section. We show examples of how the different similarity measures might convey the perceived similarity of different pictures (see Figure 2.5) in Table 2.1.

For the formal definitions let $x \in \mathbb{R}^N$ and $\hat{x} \in \mathbb{R}^N$ be two data points that we want to compare.

Euclidean distance The Euclidean distance

$$d(x, \hat{x}) = \|x - \hat{x}\|_2 = \sqrt{\sum_{i=1}^{n} (x - \hat{x})^2} \tag{2.13}$$

measures the length of the line segment between two points x and $\hat{x}$ in space by adding up the squared differences of their coordinates and then taking the square root. To apply it to images, we reshape the tensor representation of the image and write the RGB values in a single column vector. The Euclidean distance is an absolute error measure with values in $[0, \sqrt{n(x_{\max} - x_{\min})^2}]$, where $x_{\max}$ and $x_{\min}$ denote the maximum and minimum possible pixel values respectively. A Euclidean distance close to zero indicates high similarity.

Mean squared error (MSE) The mean squared error

$$\mathrm{MSE}(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^{n} (x - \hat{x})^2 \tag{2.14}$$

measures the average squared difference between x and $\hat{x}$.

Just like the Euclidean distance, the MSE is an absolute error measure with values in $[0, (x_{\max} - x_{\min})^2]$, depending on the image representation, for which values close to zero indicate high similarity. To apply the MSE to images we reshape the tensor representation of the image and write the RGB values in a single column vector.

Structural similarity index measure (SSIM) The SSIM [153] is a similarity metric introduced specifically for images. It aims to mimic human perception of image similarity based on luminance, contrast and structure.

The SSIM is always calculated for a window, that slides over the image. The final SSIM score is obtained by averaging over all windows. For a window a of image x and a window b of image $\hat{x}$ the SSIM is then defined as

$$\mathrm{SSIM}(a, b) = \frac{(2\mu_a\mu_b + C_1)(2\sigma_{ab} + C_2)}{(\mu_a^2 + \mu_b^2 + C_1)(\sigma_a^2 + \sigma_b^2 + C_2)}, \tag{2.15}$$

where μ_a and μ_b are the averages, and σ_a^2 and σ_b^2 are the variances of a and b, respectively. The covariance of a and b is denoted by σ_{ab}. The two constants $C_1 = (0.01 \cdot R)^2$ and $C_2 = (0.03 \cdot R)^2$ are based on the data range of the input image (distance between minimum and maximum possible values). For color images the SSIM is calculated separately for each channel and then averaged over all channels. The SSIM is a relative similarity measure with values in $[0, 1]$, where high values indicate high similarity.

Pearson correlation coefficient (PCC) The PCC measures the linear correlation between two inputs x and $\hat{x}$. It is a normalized measurement of the covariance and thus has values in $[-1, 1]$, where high values indicate high similarity. The formula is then

$$\mathrm{PCC}(x, \hat{x}) = \frac{\sum_{i=1}^{n} (x_i - \mu_x)(\hat{x}_i - \mu_{\hat{x}})}{\sigma_x \sigma_{\hat{x}}}, \tag{2.16}$$

where μ_x and $\mu_{\hat{x}}$ are the averages, and σ_x and $\sigma_{\hat{x}}$ are the standard deviations of x and $\hat{x}$, respectively. To apply the PCC to images we reshape the tensor representation of the image and write the RGB values in a single column vector.

a) original b) blurred c) noisy d) shift e) color shift f) 2nd image

Fig. 2.5.: Different images on which we apply the similarity metrics (see Table 2.1). a) the original image, b) added Gaussian blur ($\sigma = 5$), c) added Gaussian noise ($\epsilon \in \mathcal{N}(\mu = 0, \sigma = 0.2)$), d) shifting the image by 2 pixels to the right, e) color shift in red channel by 0.4, f) a different image

Similarity metric	$\hat{x} = x$	blurred	noisy	shift	color shift	2nd image
$d(x, \hat{x})$	0.0	0.0559	0.0615	0.1495	0.2286	0.3006
$\text{MSE}(x, \hat{x})$	0.0	0.0031	0.0355	0.0038	0.0523	0.0904
$\text{SSIM}(x, \hat{x})$	1.0	0.6236	0.1310	0.7149	0.9181	0.2845
$\text{PCC}(x, \hat{x})$	1.0	0.9552	0.6836	0.9453	0.6631	0.1648

Tab. 2.1.: Different similarity metrics applied to pictures shown in Figure 2.5. The images have a value range of $[0, 1]$, hence the MSE can be maximally 1. For better comparison we divided the Euclidean norms by the maximal Euclidean norm $d_{\max}(x, \hat{x}) = \sqrt{(224 \times 224 \times 3)}$ two images can possibly have, therefore the maximum Euclidean distance is also 1. Values close to zero indicate high similarity for $d(x, \hat{x})$ and $\text{MSE}(x, \hat{x})$, while for $\text{SSIM}(x, \hat{x})$ and $\text{PCC}(x, \hat{x})$ values close to one indicate high similarity (see second column).

3. Mathematical background

In this thesis, we repeatedly use concepts from differential geometry to analyse phenomenona we observe experimentally and, vice versa, to gain new insights that then guide further development. This section will explain some of the fundamental concepts of differential geometry and place them into context to serve as a foundation for a more detailed, formal analysis in later chapters. For extended introductions to differential geometry see [154–156].

3.1. Basics

Differential geometry relies heavily on linear algebra which is why this section reviews some basic concepts from this field.

3.1.1. Change of basis

A *vector space* is a set V with axioms for vector addition and scalar multiplication, which again yield elements $v \in V$. Scalars are elements of a *field* F, for example the field of real numbers $\mathbb{R}$.

A subset of elements $\{e_1, \dots, e_n\}$ of V is called a *basis* e_i of V if the elements are linearly independent and for any given element $u \in V$ there exist scalars $u^1, \dots u^n$ such that

$$u = u^1 e_1 + \cdots + u^n e_n \,. \tag{3.1}$$

The scalars $u^1, \dots u^n$ are called the *components* of u with respect to e_i (see Figure 3.1) and n is called the dimension of V and is independent of the basis. With a change of basis the components of u change as

$$u_{\tilde{e}_i} = A u_{e_i} \,, \tag{3.2}$$

where the matrix A describes the change of basis so that

$$\tilde{e}_i = (A^{-1})_i{}^k e_k \qquad \text{and} \qquad e_i = A_i{}^k \tilde{e}_k \,. \tag{3.3}$$

We use Einstein notation to indicate summation over the repeated index k.

The j-th component $(e_i)^j$ of the i-th standard, or canonical, basis vector e_i is given by

$$(e_i)^j = \delta_{ij}\,, \tag{3.4}$$

where $i, j \in \{1, \ldots, n\}$ and δ_{ij} is the Kronecker delta

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}\,. \tag{3.5}$$

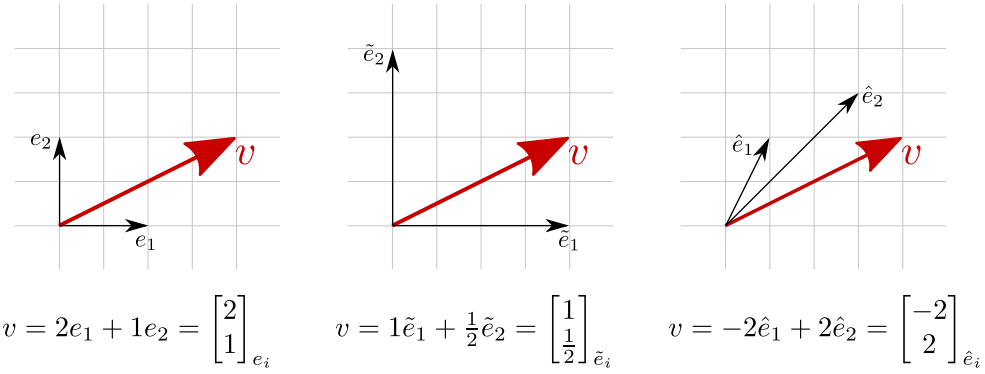


Fig. 3.1.: Representation of the same vector in different coordinate systems.

3.1.2. Vectors and dual vectors

The *dual space* V^* is the set of all linear maps $\alpha : V \to F$ from the vector space V to the underlying field F so that

$$\alpha(au + bv) = a\alpha(u) + b\alpha(v)$$

and

$$(a\alpha + b\beta)(v) = a\alpha(v) + b\beta(v) \tag{3.6}$$

for all $u, v \in V$, $a, b \in F$ and $\alpha, \beta \in V^*$. The dual space V^* is therefore itself a vector space.

Given a vector space V we can define a dual space V^* so that for any given basis e_i for V there exist a dual basis ϵ^i for V^* satisfying

$$\epsilon^i(e_j) = \delta^i{}_j\,. \tag{3.7}$$

We can interpret V as the dual of V^* so that $V^{**} = V$:

$$v \in V : V^* \to \mathbb{R} \qquad \text{by} \qquad v(\alpha \in V^*) = \alpha(v)\,. \tag{3.8}$$

A non-degenerate[1] *bilinear form* on V, such as an inner product, induces an *isomorphism*, i.e. an equivalence relation, between V and $V^* : v^* =< v, . >$. This is what lets us raise and lower indices later with the metric tensor (see Section 3.2.2).

Elements of the dual space are also called covectors, dual vectors, linear forms or one-forms.

Given a change of basis as in (3.3), where $e_i = A_i{}^k \tilde{e}_k$, vector components transform contra-variantly and dual vector components tranform co-variantly:

$$v_{e_i} = A^{-1} v_{\tilde{e}_i} \qquad \text{and} \qquad \alpha_{\epsilon_i} = \alpha_{\tilde{\epsilon}_i} A \,. \tag{3.9}$$

To indicate this we write vector components with upper indices and dual vector components with lower indices:

$$v = v^i e_i \qquad \text{and} \qquad \alpha = \alpha_i \epsilon^i \,. \tag{3.10}$$

Example 1: Velocity is a vector. If we scale our coordinate system by a factor of λ (for example $\lambda = 1000$ to go from the basis vector length representing 1m to representing 1km), we get the new basis vectors $\tilde{e}_i = \lambda e_i$. The vector components of some vector representing velocity are then divided by λ in the scaled coordinate system: $v_{\tilde{e}_i} = \frac{1}{\lambda} v_{e_i}$, as $1\frac{m}{h} = \frac{1}{1000}\frac{km}{h}$. Thus the velocity vector is a vector that contra-varies with a change of coordinates.

Example 2: The differential of a function is a dual vector. The total derivative of a function $f : \mathbb{R}^n \to \mathbb{R}$ at point p is a dual vector $Df_p = \left[\frac{\partial f}{\partial x_1}(p) \dots \frac{\partial f}{\partial x_n}(p)\right]$. This could be for example the inclination of a mountain at each point. If we scale our coordinate system by a factor of λ (for example $\lambda = 1000$ to go from the basis vector length representing 1m to representing 1km), the dual basis is given by (3.7) as $\epsilon^i(e_j) = \frac{1}{\lambda}\epsilon^i(\lambda e_j) = \tilde{\epsilon}_i(\tilde{e}_j)$. Any dual vector α then transforms in the same way as the basis vectors e_i, namely as $\alpha_{\tilde{\epsilon}^i} = \lambda \alpha_{\epsilon^i}$. Thus the inclination $\frac{1}{\text{m}} = 1000\frac{1}{\text{km}}$ co-varies with a change of coordinates.

3.1.3. Linear maps

Given two vector spaces V and W over the same field F a *linear map* is a function $f : V \to W$ preserving vector addition

$$f(u + v) = f(u) + f(v) \tag{3.11}$$

and scalar multiplication

$$f(cu) = cf(u) \,, \tag{3.12}$$

[1] A bilinear form $f : V \times V \to F$ is non-degenerate if and only if $f(x, y) = 0 \quad \forall \quad y \in V$ implies, that $x = 0$.

where $u, v \in V$ and $c \in F$.

If V and W are finite dimensional vector spaces of dimensions n and m respectively, the linear map $f : V \to W$ can be represented as an $m \times n$ matrix

$$M^i{}_j = f^i(e_j)\,, \tag{3.13}$$

where e_j are basis vectors of V and the images $f(e_j)$ are expressed in a basis of W, so that

$$f(u) = Mu\,. \tag{3.14}$$

Under a change of basis $\tilde{e}_i = (A^{-1})_i{}^k e_k$, M transforms as

$$M_{\tilde{e}_i} = A M_{e_i} A^{-1}\,. \tag{3.15}$$

The dual transformation $M^* : V^* \to W^*$ of a linear map represented by a matrix M is given by

$$M^*(\alpha) = \alpha \circ M = \beta \tag{3.16}$$

where $\beta_j = \alpha_i M^i{}_j$.

3.1.4. Inner product

The scalar product, or dot product, is an inner product in Euclidean space which maps two vectors $u, v \in \mathbb{R}^n$ onto a scalar

$$\langle u, v \rangle = \sum_{i=1}^{n} u^i v^i\,, \tag{3.17}$$

where u^i, v^i are the vector components in Cartesian coordinates.

The more general definition of an inner product is that of a map $\langle \bullet, \bullet \rangle : V \times V \to F$ which satisfies following properties:

(i) positive-definiteness: $\langle v, v \rangle \geq 0$ and $\langle v, v \rangle = 0$ if and only if $v = \mathbf{0}$, where $\mathbf{0}$ denotes the zero vector,

(ii) conjugate symmetry: $\langle v, w \rangle = \overline{\langle w, v \rangle}$, and

(iii) bilinearity: $\langle au + bv, w \rangle = a \langle u, w \rangle + b \langle v, w \rangle$,

where V is a vector space over field F, elements $u, v, w \in V$ are vectors and $a, b \in F$ are scalars. A vector space V together with an inner product $\langle \bullet, \bullet \rangle : V \times V \to F$ is called an inner product space and is a generalization of Euclidean space.

3.1.5. Eigenvectors and eigenvalues

Given a linear map $f : V \to V$ from a vector space V over the field $\mathbb{R}$ into V, an *eigenvector* v is a non-zero vector such that the application of f alters only the scale of v by *eigenvalue* $\lambda \in F$:

$$f(v) = \lambda v \,. \tag{3.18}$$

Any rescaled eigenvector sv with $s \in \mathbb{R}$ is also an eigenvector with the same eigenvalue. We therefore usually represent v as a unit vector.

For a linear map given by a square $n \times n$ matrix A with n linearly independent eigenvectors, we can write the eigendecomposition of A as

$$A = Q\Lambda Q^{-1} \,, \tag{3.19}$$

where Λ is a diagonal matrix with $\Lambda_{ii} = \lambda_i$ and Q is a square $n \times n$ matrix with the corresponding eigenvectors q_i as columns. If in addition A is a real symmetric matrix, the eigenvalues are real and the eigenvectors q_i are real and orthogonal.

Example 1: Rotation in $\mathbb{R}^3$. Rotations are linear operations and can thus be represented as matrices. If we have a rotation in 3D, the rotation axis is an eigenvector with eigenvalue $\lambda = 1$, since the space is only rotated and not stretched. The remaining eigenvalues are complex.

Example 2: Uniformly stretching space. If we have a transformation in $\mathbb{R}^2$, given by the matrix $A = cI$, where I is the identity matrix and $c \in \mathbb{R}$, the only eigenvalue is $\lambda = c$ and all vectors $v \in \mathbb{R}^2$ are eigenvectors.

3.1.6. Directional derivatives

The *directional derivative* of a scalar valued function $f : \mathbb{R}^n \to \mathbb{R}$ at point p is the instantaneous change of f when moving through p with a velocity $v \in \mathbb{R}^n$ and is defined by

$$D_v f(p) = \nabla_v f = \lim_{\epsilon \to 0} \frac{f(p + \epsilon v) - f(p)}{\epsilon} = \frac{\mathrm{d}}{\mathrm{d}t} f(p + \epsilon v)\bigg|_{\epsilon=0} = \nabla f(p) \cdot v \,, \tag{3.20}$$

where $\nabla f(p)$ is the gradient of f at p. The directional derivative of a scalar-valued function is itself a scalar.

If we have a vector valued function, $f : \mathbb{R}^n \to \mathbb{R}^m$ the directional derivative is given by

$$D_v f(p) = Df(p)v = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(p) & \dots & \frac{\partial f_1}{\partial x_n}(p) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(p) & \dots & \frac{\partial f_m}{\partial x_n}(p) \end{bmatrix} v, \tag{3.21}$$

where $Df(p)$ is the Jacobian matrix at p. The directional derivative of a vector valued function yields itself a vector $D_v f(p) \in \mathbb{R}^m$.

3.1.7. Tensors

The notion of a *tensor* is a generalization of vectors and dual vectors. A type (m, n) tensor T is a multilinear map

$$T : \underbrace{V^* \times \cdots \times V^*}_{m \text{ times}} \times \underbrace{V \times \cdots \times V}_{n \text{ times}} \to \mathbb{R} \tag{3.22}$$

from a collection of vectors and dual vectors to an element of $\mathbb{R}$. The symbol $\times$ denotes the *Cartesian product* with which we can combine vector spaces V and W into a new vector space $V \times W = \{(v, w) \mid v \in V \text{ and } w \in W\}$.

The space of all tensors of a fixed type (m, n) is itself a vector space with elements that can be added together and multiplied by real numbers. A tensor is a basis independent object in space that can be represented with respect to a basis by a multidimensional array whose components transform according to the covariant and contravariant transformation laws. We can find the components in a given basis by applying T to basis vectors e_i and dual vectors ϵ^i:

$$T^{\mu_1 \dots \mu_m}{}_{\nu_1 \dots \nu_n} = T(\epsilon^{\mu_1}, \dots \epsilon^{\mu_m}, e_{\nu_1}, \dots, e_{\nu_n}) \,. \tag{3.23}$$

We then write $T^{\mu_1 \cdots \mu_m}{}_{\nu_1 \dots \nu_n}$ with m contravariant (upper) and n covariant (lower) indices.

If T is an (m, n) tensor and S is an (i, j) tensor we can combine them using the *tensor product* $\otimes$. The resulting $(m + i, n + j)$ tensor $T \otimes S$ takes $m + i$ dual vectors $\alpha^{(i)}$ and $n + j$ vectors $v^{(i)}$ and maps them onto a scalar

$$\begin{aligned} T \otimes S(\alpha^{(1)}, \dots, \alpha^{(m+i)}, v^{(1)}, \dots v^{(n+j)}) \\ &= T(\alpha^{(1)}, \dots, \alpha^{(m)}, v^{(1)}, \dots v^{(n)}) \\ &\quad \times S(\alpha^{(m+1)}, \dots, \alpha^{(m+i)}, v^{(n+1)}, \dots v^{(n+j)}) \,. \end{aligned} \tag{3.24}$$

We can straightforwardly construct a basis for T by taking tensor products of the basis vectors and basis dual vectors so that

$$T = T^{\mu_1 \cdots \mu_m}{}_{\nu_1 \dots \nu_n} e_{\mu_1} \otimes \cdots \otimes e_{\mu_m} \otimes \epsilon^{\nu_1} \otimes \cdots \otimes \epsilon^{\nu_n} \,. \tag{3.25}$$

Example 1: Linear map. A linear map $L : V \to V$ is a linear combination of vector-dual vector tensor products so that $L = L^i_j(e_i \otimes \epsilon^j) \in V \otimes V^*$ is a (1,1)-tensor. In $\mathbb{R}^2$, the tensor products $\{e_1 \otimes \epsilon^1, e_1 \otimes \epsilon^2, e_2 \otimes \epsilon^1, e_2 \otimes \epsilon^2\}$ form a basis for all linear maps $A : V \to V$ since we can form any matrix A by a linear combination of the basis elements. If e_i and ϵ^i are the canonical vector and dual vector bases, respectively, we get

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + b \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + c \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + d \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Example 2: Bilinear form. A bilinear form $\mathcal{B} : V \times V \to \mathbb{R}$ is a (0,2)-tensor, i.e. a linear combination of dual vector–dual vector tensor products $\mathcal{B} = \mathcal{B}_{ij}(\epsilon^i \otimes \epsilon^j)$, which maps two elements of a vector space to a scalar. The tensor product $\mathcal{B}_{ij}(\epsilon^i \otimes \epsilon^j)$ in two dimensions, with ϵ^i as the canonical dual vector basis, gives

$$\mathcal{B}(u, v) = \mathcal{B}_{ij} u^i v^j = \begin{bmatrix} u^1 & u^2 \end{bmatrix} \begin{bmatrix} \mathcal{B}_{11} & \mathcal{B}_{12} \\ \mathcal{B}_{21} & \mathcal{B}_{22} \end{bmatrix} \begin{bmatrix} v^1 \\ v^2 \end{bmatrix}.$$

3.2. Differential geometry

Differential geometry is a mathematical discipline that studies the geometry of smooth manifolds. In this section, we introduce some of the most common concepts of differential geometry, which we will use for our theoretical analyses in later chapters.

3.2.1. Manifolds

An n-dimensional *manifold* $\mathcal{M}$ is a topological space that locally resembles n-dimensional Euclidean space. Manifolds are described by *coordinate charts* assembled into an *atlas*. A coordinate chart is a homeomorphism x from an open subset U of $\mathcal{M}$ to an open subset of Euclidean space $\mathbb{R}^n$ and an atlas is the collection $\{(U_i, x_i) : i \in I\}$ of charts, which completely covers $\mathcal{M}$.

Manifolds naturally arise as graphs of functions. For our purposes we consider smooth Riemannian manifolds as these allow us to define differentiable functions and measure notions like lengths, and angles of vector fields on the manifold.

Example: Surface of the earth. The surface of the earth is approximately a sphere and thus a two dimensional manifold, embedded in three dimensional space. When zooming in on a particular point on the earth, for example Berlin, we can use a two dimensional plane as an accurate map of the area. Globally though, we end up with large distortions when we try to map the entire surface of the earth to a single planar map (see Figure 3.2).

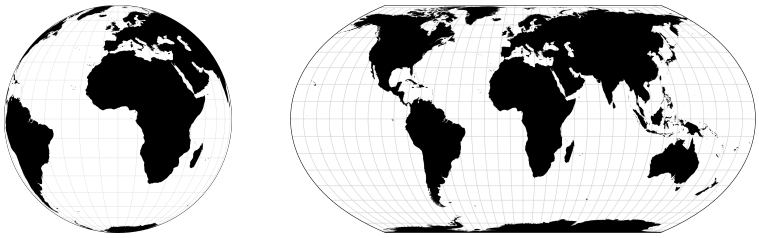


Fig. 3.2.: Projection of the surface of the earth (image source: [157]). Latitude and longitude intersect at 90° angles on the globe (left) but are distorted when projected to a plane (right).

Tangent space

We can assign a vector space, called the *tangent space* $T_p\mathcal{M}$, to each point p that lies on the manifold (see Figure 3.5). The dimension of the tangent space $T_p\mathcal{M}$ is the same as that of the manifold itself. The tangent space to a manifold at a point p can be thought of as the space of possible velocities $\tau'(t)$ at point p for a particle moving along all possible curves $\tau(t)$ on the manifold and passing through p. A coordinate chart induces a basis in the tangent space and a change of chart is called a change of coordinates.

Example: Tangent plane. For the 2D surface of a sphere embedded in $\mathbb{R}^3$ the tangent space at a point p is the plane that touches the sphere at p and is perpendicular to the sphere's radius through p.

Cotangent space

The *cotangent space*, or dual space $T_p^*\mathcal{M}$ of a manifold $\mathcal{M}$ at point p consists of all linear forms $\alpha \in T_p^*\mathcal{M}$ that map a vector of the tangent space to a real number $\alpha : T_p\mathcal{M} \to \mathbb{R}$. The dimension of the cotangent space is the same as the dimension of the tangent space and the manifold itself.

The differential $\mathrm{d}f$ of a function $f : \mathcal{M} \to \mathbb{R}$ at point p is the linear map

$$\mathrm{d}f_p(\tau'(0)) = (f \circ \tau)'(0)\,, \tag{3.26}$$

where $\tau : \mathbb{R} \to \mathcal{M}$ is a curve on $\mathcal{M}$ with $\tau(0) = p$. It is an element of the cotangent space as it maps a vector $\tau'(0) \in T_p\mathcal{M}$ from the tangent space to the derivative of f along τ', which is in $\mathbb{R}$ (see also 3.20).

3.2.2. Riemannian metric

A manifold can be endowed with a structure called a *Riemannian metric*. The Riemannian metric γ provides a way of measuring distances and angles on manifolds

as it defines lengths of vectors and angles between vectors in a basis independent manner. It is a smoothly varying inner product on the tangent space $T_p\mathcal{M}$ at each point $p \in \mathcal{M}$ that maps two elements $u, v \in T_p\mathcal{M}$ to $\gamma_p(u, v) \in \mathbb{R}$.

The metric tensor is an object that exists independent of any coordinate system, but the components γ_{ij} of the matrix representation of the metric tensor change depending on the coordinate system. This matrix is a symmetric positive definite matrix whose entries co-vary with changes to the coordinate system. This makes the metric tensor a special bilinear form as $\gamma(u, v) = \gamma(v, u)$ and $\gamma(u, u) > 0$ for all nonzero vectors u.

The metric tensor is also sometimes called the *first fundamental form* $\mathrm{I}(u, v)$.

The length of a tangent vector u is given by

$$\|u\| = \sqrt{\gamma(u, u)} \tag{3.27}$$

and the angle θ between two tangent vectors u and v at p is given by

$$\cos\theta = \frac{\gamma(u, v)}{\|u\| \, \|v\|}. \tag{3.28}$$

Example 1: Length of a vector in $\mathbb{R}^2$. We are used to calculating the length of a vector $v = [v^1, v^2]^T = v^1 e_1 + v^2 e_2$ as $\|v\|^2 = (v^1)^2 + (v^2)^2$ but this only works if v is given in orthonormal coordinates e_i. The vector v itself is invariant, so the length will stay the same even if the vector is represented in another coordinate system (see Figure 3.1). If we choose a different coordinate system in which we represent the vector as $v_{\tilde{e}_i} = \tilde{v}^1 \tilde{e}_1 + \tilde{v}^2 \tilde{e}_2$, we need the more general definition of the inner product to compute the vector length $\|v\|$.

$$\begin{aligned}
\|v\|^2 = \langle v, v \rangle &= (\tilde{v}^1 \tilde{e}_1 + \tilde{v}^2 \tilde{e}_2) \cdot (\tilde{v}^1 \tilde{e}_1 + \tilde{v}^2 \tilde{e}_2) \\
&= (\tilde{v}^1)^2 \, (\tilde{e}_1 \cdot \tilde{e}_1) + 2\tilde{v}^1 \tilde{v}^2 \, (\tilde{e}_1 \cdot \tilde{e}_2) + (\tilde{v}^2)^2 \, (\tilde{e}_2 \cdot \tilde{e}_2) \\
&= \begin{bmatrix} \tilde{v}^1 & \tilde{v}^2 \end{bmatrix} \begin{bmatrix} \tilde{e}_1 \cdot \tilde{e}_1 & \tilde{e}_2 \cdot \tilde{e}_1 \\ \tilde{e}_1 \cdot \tilde{e}_2 & \tilde{e}_2 \cdot \tilde{e}_2 \end{bmatrix} \begin{bmatrix} \tilde{v}^1 \\ \tilde{v}^2 \end{bmatrix} \\
&= \begin{bmatrix} v^1 & v^2 \end{bmatrix} \begin{bmatrix} e_1 \cdot e_1 & e_2 \cdot e_1 \\ e_1 \cdot e_2 & e_2 \cdot e_2 \end{bmatrix} \begin{bmatrix} v^1 \\ v^2 \end{bmatrix}.
\end{aligned}$$

The matrix used for the matrix vector multiplication is the representation of the metric tensor in the respective coordinate system, i.e.

$$\gamma_{e_i} = \begin{bmatrix} e_1 \cdot e_1 & e_2 \cdot e_1 \\ e_1 \cdot e_2 & e_2 \cdot e_2 \end{bmatrix} \qquad \text{and} \qquad \gamma_{\tilde{e}_i} = \begin{bmatrix} \tilde{e}_1 \cdot \tilde{e}_1 & \tilde{e}_2 \cdot \tilde{e}_1 \\ \tilde{e}_1 \cdot \tilde{e}_2 & \tilde{e}_2 \cdot \tilde{e}_2 \end{bmatrix}.$$

We can also use the metric to define a canonical isomorphism between $T_p\mathcal{M}$ and its dual space $T_p^*\mathcal{M}$. This implies that contraction with the metric γ_{ij} and its inverse

γ^{ij} is used to raise and lower indices:

$$v_i = \gamma_{ij} v^j \quad \text{and} \quad v^i = \gamma^{ij} v_j \,. \tag{3.29}$$

3.2.3. Curves

A parametrized curve is a smooth map $\tau : [t_1, t_2] \to \mathcal{M}$, that maps the scalar parameter t to points $p = \tau(t)$ on a manifold $\mathcal{M}$. The set of points are called a geometric curve and define the image of the parametrized curve. One geometric curve can have many different parametrizations. A parametrized curve is called regular when its velocity $\tau'(t)$ is never zero in $[t_1, t_2]$.

One can calculate the arc length of the curve by integrating over the norm of its velocity $\tau'(t)$:

$$l = \int_{t_0}^{t_1} \|\tau'(u)\| \, \mathrm{d}u \,. \tag{3.30}$$

We can use the arc length to parametrize any regular curve. For this we define the arc length function $s : [t_0, t_1] \to [0, l]$ as

$$s(t) = \int_{t_0}^{t} \|\tau'(u)\| \, \mathrm{d}u. \tag{3.31}$$

As $s(t)$ is a monotonically increasing function the inverse $t(s)$ exists and we can write the arc length parametrization of our regular curve $\tau(t)$ as $\tau(s) = \tau(t(s))$. This is useful since it is a particularly simple parametrization of the curve, since

$$\|\tau'(s)\| = \|\tau'(t(s))\| \, |t'(s)| = \frac{\mathrm{d}s}{\mathrm{d}t} \left| \frac{\mathrm{d}t}{\mathrm{d}s} \right| = \left| \frac{\mathrm{d}s}{\mathrm{d}t} \frac{\mathrm{d}t}{\mathrm{d}s} \right| = 1 \,.$$

A curve that is parametrized by arc length is then equivalent to a unit speed curve $\tau(t)$ that starts at $t = 0$.

A *geodesic* is a parametrized curve with minimal arc length, i.e. the shortest possible path on the manifold $\mathcal{M}$ that connects two points $p_1 = \tau(t_1)$ and $p_2 = \tau(t_2)$. It is thus the generalization of a "straight line" to curved surfaces. Figure 3.3 shows two examples of curves on a cone. The geodesic corresponds to a straight line when the cone rolls on a plane.

If the length of the tangent vector $\frac{\mathrm{d}\tau(\eta)}{\mathrm{d}\eta}$ of a geodesic $\tau(\eta)$ is constant (as measured by the metric) along $\tau(\eta)$, the geodesic is *affinely-parametrized*. The parameter η is affinely related to the arc length $s = a\eta + b$. Importantly, the notion of an affinely parametrized geodesic is coordinate independent and can therefore itself be used to construct coordinates on $\mathcal{M}$ (see Chapter 6).

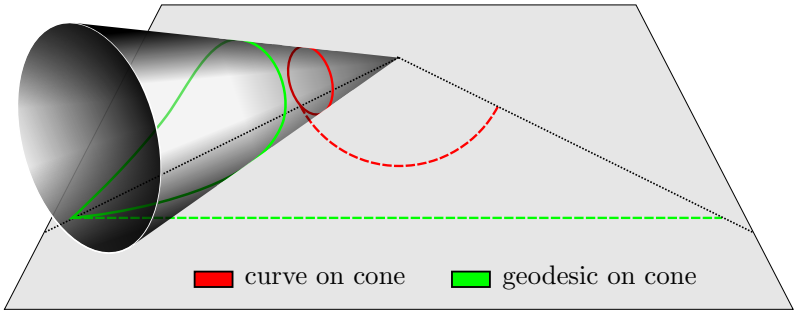


Fig. 3.3.: Two curves on a cone. The geodesic curve maps out a straight line, when the cone rolls on the plane.

Curvature of a plane curve

For a curve in a plane $\tau : [0, l] \to \mathbb{R}^2$ parametrized by arc length, the velocity vector $\tau'(s)$ has unit length and is tangent to τ at $p = \tau(s)$. The curvature at p is given by the derivative of the tangent vector

$$\tau''(s) = \frac{\mathrm{d}\tau'}{\mathrm{d}s}(s)\,.$$

As the parametrization imposes unit speed, the change in velocity $\tau''(s)$ can only occur perpendicular to the velocity $\tau'(s)$. Therefore $\tau''(s)$ must be a multiple of the normal n:

$$\tau''(s) = \kappa n(s).$$

The normal $n(s)$ is usually chosen so that the pair $(\tau'(s), n(s))$ is oriented positively in the plane. If κ is positive the curve bends towards n and if κ is negative the curve bends away (see Figure 3.4). The magnitude of κ tells us how much the curve bends.

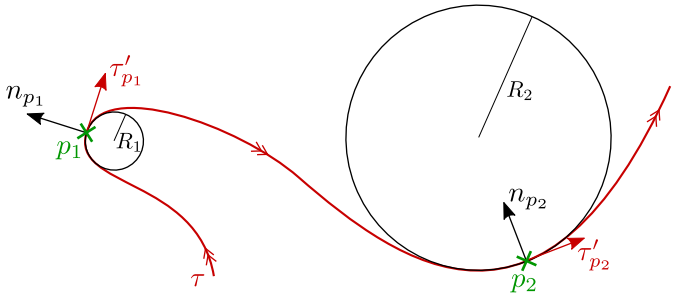


Fig. 3.4.: The circle of curvature at two points of an oriented plane curve τ. The curvature κ is inversely proportional to the radius, so that $\kappa_{p_1} = -\frac{1}{R_1}$ and $\kappa_{p_2} = \frac{1}{R_2}$.

There are two possible arc length parametrizations, i.e. we can just switch start and end point and traverse the curve in the opposite direction. Reversing the orientation of the curve then translates to flipping the sign of the curvature at each point.

3.2.4. Surfaces

To generalize the notion of curvature to 2D manifolds $\mathcal{M}$ in Euclidean space $\mathbb{R}^3$, imagine a curve $\tau(s)$ on $\mathcal{M}$, parametrized by arc length, with $\tau(0) = p$. We can then imagine slicing the surface with the plane spanned by the unit normal N_p and the unit tangent vector $T_p = \tau'(0)$ at point p (see Figure 3.5). The set of the unit tangent vectors of all possible curves through p spans a circle S^1 and we can define a function $\kappa : S^1 \to \mathbb{R}$ as

$$\kappa(T_p) = \langle \tau''(0), N_p \rangle \ , \tag{3.32}$$

where $\langle \bullet, \bullet \rangle$ is the usual inner product in $\mathbb{R}^3$. The maximum and minimum values, κ_1 and κ_2, over the choice of curve of this function are called the *principal curvatures* of the surface at point p. Their product $\kappa_1\kappa_2$ is the Gaussian curvature K, which is invariant under isometries (see Theorema Egregium of Gauss). The respective directions are called principal directions.

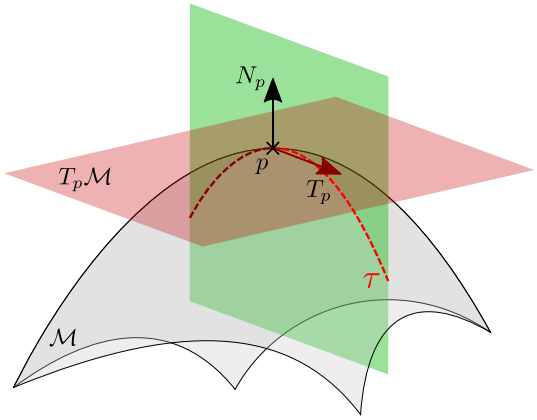


Fig. 3.5.: The normal section spanned by unit normal N_p and unit tangent $T_p = \tau'(s)$ at point p is marked in green. The tangent plane $T_p\mathcal{M}$ at p spanned by the velocity vectors of all possible curves through p is depicted in red.

In general, we have different notions of curvature for parametrized curves on curved surfaces. The normal curvature describes how the surface bends and the geodesic curvature describes how the curve bends if projected onto the tangent plane (Section 3.2.3). If we denote normal curvature as k_n and the geodesic curvature as k_g, we can define the ambient curvature as $k = \sqrt{k_g^2 + k_n^2}$. For a plane curve, the normal curvature is zero and $k = k_g$. For a geodesic on a curved surface, the geodesic curvature is zero and $k = k_n$.

Example 1: Relation of curvature to acceleration of a car. Imagine we are driving a car with constant speed. When driving on a curvy road on a plane ($k = k_g$) one can still feel the acceleration although its component in direction of the velocity is zero. If we are driving straight ahead, but the underlying terrain is very hilly ($k = k_n$) we can feel the acceleration perpendicular to the surface.

Example 2: Flat metric. If a Riemannian manifold has a zero curvature tensor we call it a flat manifold and the corresponding metric a *flat metric*. Euclidean space itself has a flat metric.

3.2.5. Shape operator/Weingarten map

The Weingarten map generalizes the notion of curvature to higher dimensions. For a smooth n-surface $\mathcal{M}$ in $\mathbb{R}^{n+1}$, we define the directional derivative along a tangent vector $T_p \in T_p\mathcal{M}$ by

$$L_p(T_p) = -D_{T_p} N(p)\,, \tag{3.33}$$

where N is the unit normal vector field on $\mathcal{M}$. L_p is a directional derivative of a vector field and is called the Weingarten map of the surface $\mathcal{M}$ at p. It is a linear map $T_p\mathcal{M} \to T_p\mathcal{M}$ that projects an element of the tangent space at p to an element of that same tangent space. As a geometric intuition one can think of a curve $\tau(t)$ through $p = \tau(t_0)$. The Weingarten map $L_p(\tau'(t_0))$ then measures the rate of change of the normal vector N when passing through p along the curve $\tau(t)$ with velocity $\tau'(t_0)$. Measuring the turning of the normal, and thus the turning of the tangent space, provides information about the shape of the surface $\mathcal{M}$, which is why L_p is also called the shape operator.

In general, the eigenvectors and eigenvalues of the shape operator at each point determine the directions in which the surface bends at each point. The eigenvalues are called the principal curvatures of the surface and the eigenvectors are the corresponding principal directions. The Gaussian curvature is the product of the principal curvatures $K(p) = \prod_{i=1}^{n} \lambda_i = \det(L_p)$.

3.2.6. Second fundamental form

If T_p is a unit vector, tangential to a smooth n-surface $\mathcal{M}$ in $\mathbb{R}^{n+1}$, i.e. it is the velocity $\tau'(0) = T_p$ at $p = \tau(0)$ of a curve $\tau(s)$ parametrized by arc length, then we can express the normal curvature $\kappa_n(T_p) = \langle L_p(T_p), T_p \rangle$ with respect to the shape operator.

We can generalize this to express the *second fundamental form*

$$\mathrm{I\!I}(X_p, Y_p) = \langle L_p(X_p), Y_p \rangle\, N_p = -\left\langle D_{X_p} N_p, Y_p \right\rangle N_p\,, \tag{3.34}$$

which is bilinear and symmetric.

While the first fundamental form tells us how to calculate distances on the manifold, the second fundamental form describes the intrinsic and extrinsic curvature of the manifold when embedded into the ambient space. Together they uniquely describe the surface in space.

3.2.7. Pushforward and pullback

Every differentiable map $\phi : \mathcal{M} \to \mathcal{N}$ between manifolds induces a linear map $\phi_* : T_p\mathcal{M} \to T_{\phi(p)}\mathcal{N}$ at point p between the tangent spaces of these manifolds. This is called the *pushforward* or the differential of ϕ. So while ϕ maps points on $\mathcal{M}$ to points on $\mathcal{N}$ the pushforward ϕ_* maps tangent vectors on $\mathcal{M}$ to tangent vectors on $\mathcal{N}$ (see Figure 3.6):

$$\phi_*(\tau'(0)) = \mathrm{d}\phi_p(\tau'(0)) = (\phi \circ \tau)'(0)\,, \tag{3.35}$$

where τ is a curve with $\tau(0) = p$.

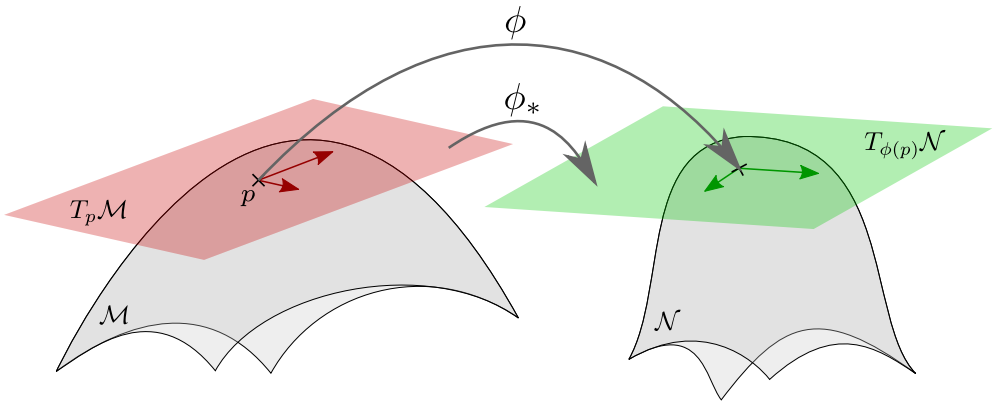


Fig. 3.6.: The map ϕ carries every point on the manifold $\mathcal{M}$ to a point on the manifold $\mathcal{N}$. The pushforward ϕ_* of ϕ carries vectors in the tangent space at every point in $\mathcal{M}$ to a tangent space at every point in $\mathcal{N}$.

The *pullback* of a function $f : \mathcal{N} \to \mathbb{R}$ by ϕ is a linear map given by

$$(\phi^* f)(x) = f(\phi(x)). \tag{3.36}$$

While the pushforward $\phi_* : T_p\mathcal{M} \to T_{\phi(p)}\mathcal{N}$ is a mapping between tangent spaces, the pullback $\phi^* : T^*_{\phi(p)}\mathcal{N} \to T^*_p\mathcal{M}$ maps between cotangent spaces in the reverse direction. Given a dual vector $\alpha \in T^*_{\phi(P)}\mathcal{N}$ we can calculate the pullback of α by ϕ as

$$(\phi^*\alpha)(X_p) = \alpha(\phi_* X_p)\,, \tag{3.37}$$

where $X_p \in T_pM$.

Similarly we can pull a metric γ defined on $\mathcal{N}$ back onto $\mathcal{M}$ by

$$(\phi^*\gamma)(X_p, Y_p) = \gamma(\phi_* X_p, \phi_* Y_p)\,, \tag{3.38}$$

where $X_p, Y_p \in T_pM$.

3.2.8. Diffeomorphism

A bijective function $\phi : \mathcal{M} \to \mathcal{N}$ is called a *diffeomorphism* if it is differentiable and has an inverse $\phi^{-1} : \mathcal{N} \to \mathcal{M}$ which is differentiable as well (see Figure 3.7). The existence of a diffeomorphism implies that the dimensions of $\mathcal{M}$ and $\mathcal{N}$ agree. A manifold $\mathcal{M}$ is equipped with coordinate charts $x : \mathcal{M} \to \mathbb{R}^n$.

A change of coordinates is a diffeomorphism and therefore induces a change of basis in the tangent space. If we denote coordinates in one basis as x^μ and in the other as x^α, the components of a vector $v \in T_pM$ transform as

$$v^\alpha = \frac{\partial \phi^\alpha}{\partial x^\mu} v^\mu \tag{3.39}$$

under the diffeomorphism ϕ.

The components of the metric tensor $\gamma_{\mu\nu}$ transform under the diffeomorphism ϕ as

$$\gamma_{\alpha\beta} = \frac{\partial \phi^\mu}{\partial x^\alpha} \frac{\partial \phi^\nu}{\partial x^\beta} \gamma_{\mu\nu}\,. \tag{3.40}$$

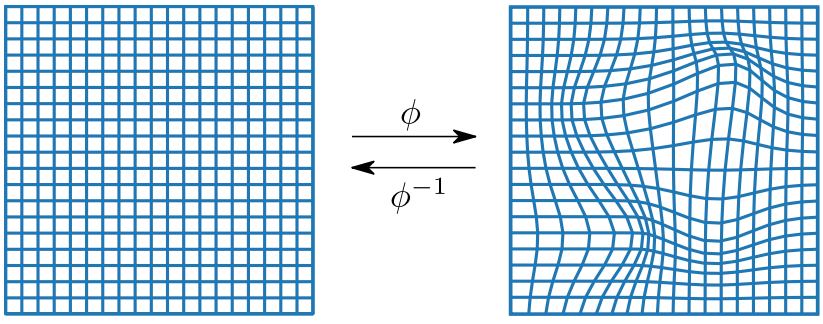


Fig. 3.7.: The image of a diffeomorphism of a rectangular grid from the square onto itself.

Example Let us consider a one dimensional manifold given in polar coordinates by the curve

$$\tau(t) = \begin{bmatrix} 1 \\ t \end{bmatrix},$$

where $t \in [0, 2\pi)$. This curve maps out the unit circle in Cartesian coordinates. The tangent vector $\tau'(t) = [0, 1]^T$ in polar coordinates is constant. We can define a change from polar to Cartesian coordinates as the diffeomorphism

$$\phi(r, \varphi) = \begin{bmatrix} r\cos(\varphi) \\ r\sin(\varphi) \end{bmatrix},$$

with the Jacobian matrix

$$J_\phi(r, \varphi) = \begin{bmatrix} \frac{\partial \phi_1}{\partial r} & \frac{\partial \phi_1}{\partial \varphi} \\ \frac{\partial \phi_2}{\partial r} & \frac{\partial \phi_2}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & -r\sin(\varphi) \\ \sin(\varphi) & r\cos(\varphi) \end{bmatrix}.$$

We can then push the tangent vector $\tau'(t)$ at point $p = \tau(t)$ forward to the respective tangent vector $T_{\phi(p)}$ in Cartesian coordinates at point $q = \phi(p)$ by

$$T_{\phi(p)} = J_\phi(p)\tau'(t) = \begin{bmatrix} -r\sin(\varphi) \\ r\cos(\varphi) \end{bmatrix}(p).$$

For an example point at $t = \frac{\pi}{4}$ consider the resulting numerical values:

$$p = \tau(t) = \begin{bmatrix} 1 \\ \frac{\pi}{4} \end{bmatrix} \qquad \tau'(t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$q = \phi(p) = \begin{bmatrix} \cos(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \end{bmatrix} \qquad T_q = J_\phi(p)\tau'(t) = \begin{bmatrix} -\sin(\frac{\pi}{4}) \\ \cos(\frac{\pi}{4}) \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}.$$

4. Manipulating explanations

In this chapter we analyse the robustness of attribution methods. We propose an algorithm which allows us to manipulate an image with hardly perceptible perturbations such that the prediction is unchanged but the explanation map matches an arbitrary target map. The approach works analogous to conventional adversarial attacks that add imperceptible perturbations to an image which cause a drastic change in the prediction of a neural network. The crucial difference is that we aim to change the explanation of the image while keeping the prediction at its original value. Figure 4.1 shows an image and a manipulated version of the same image. The two pictures look identical but have drastically different explanation maps shown below the respective images.

Fig. 4.1.: Original image with corresponding explanation map on the left. Manipulated image with its explanation on the right. The chosen target explanation was an image with a text stating "this explanation was manipulated".

We demonstrate the effectiveness of these adversarial attacks on six different explanation methods, covering propagation-based as well as gradient-based explanations, and on four network architectures as well as two data sets.

Untrustworthy explanations are evidently problematic for various reasons. For a large number of applications, one is interested in the prediction as well as in the explanation

of a phenomenon. Examples include medical and natural science applications. As some explanations are susceptible even to random input perturbations, it seems questionable if much insight can be derived from inspecting such explanations. In a setting where explanations are legally required [31], explanation manipulability obviously raises serious concerns as they cannot be considered trustworthy evidence. An example for this is credit risk assessment: The supplier can obfuscate that a decision was made based on racist, sexist or other discriminating features by manipulating the model [158]. Similarly, attacks from the user side are possible by manipulating the input as they can create the impression that the decision was based on unaccepted features and thus subvert the result.

Motivated by this unexpected susceptibility to manipulation, we provide a theoretical analysis that establishes a relation of this phenomenon to the geometry of the neural network's output manifold. More specifically, we derive a bound proportional to two differential geometric quantities: the principal curvatures and the geodesic distance between the original input and its manipulated counterpart. This implies a constraint on the maximal change of the explanation map due to small perturbations. Based on these theoretical results we propose a modification applicable to all of the previously investigated explanation method, which reduces susceptibility to manipulation.

4.1. Method

We manipulate an image $x_{\text{adv}} = x + \delta x$ so that the explanation map of a given explanation method resembles a specified target map $h^t \in \mathbb{R}^N$. We want the manipulated image to have the following characteristics:

1. The output of the network stays approximately constant, i.e. $f(x_{\text{adv}}) \approx f(x)$.
2. The explanation is close to the target map, i.e. $h(x_{\text{adv}}) \approx h^t$.
3. The norm of the perturbation δx added to the input image is small, i.e. $\|\delta x\| = \|x_{\text{adv}} - x\| \ll 1$ and therefore not perceptible.

Such manipulations can be obtained by minimizing the loss function

$$\mathcal{L} = \left\|h(x_{\text{adv}}) - h^t\right\|^2 + \alpha \left\|f(x_{\text{adv}}) - f(x)\right\|^2 , \tag{4.1}$$

with respect to x_{adv} using gradient descent. We clip x_{adv} after each iteration to the valid range of pixel values for an image. The first term in the loss function (4.1) ensures that the manipulated explanation map closely reproduces the target explanation and the second term encourages the network to have the same output as the original input. The relative weighting of these two terms is controlled by the hyperparameter $\alpha \in \mathbb{R}_+$.

To apply our method, we need to determine the derivative of the loss function which includes the gradient of the explanation $\nabla h(x)$ with respect to the input. For

ReLU-networks, this gradient usually depends on the vanishing second derivative of the non-linearities which leads to problems during optimization. As an example, consider the gradient method which leads to

$$\partial_{x_{\text{adv}}} \left\| h(x_{\text{adv}}) - h^t \right\|^2 \propto \frac{\partial h}{\partial x_{\text{adv}}} = \frac{\partial^2 f}{\partial x_{\text{adv}}^2} \propto \text{ReLU}'' = 0\,.$$

We therefore replace the ReLU-functions with softplus non-linearities

$$\text{softplus}_\beta(x) = \frac{1}{\beta} \log(1 + e^{\beta x})\,. \tag{4.2}$$

For large β values, the softplus reproduces the output of the ReLU accurately (see Figure 4.2) but has a well-defined second derivative. After optimization is complete, we test the manipulated image with the original ReLU network.

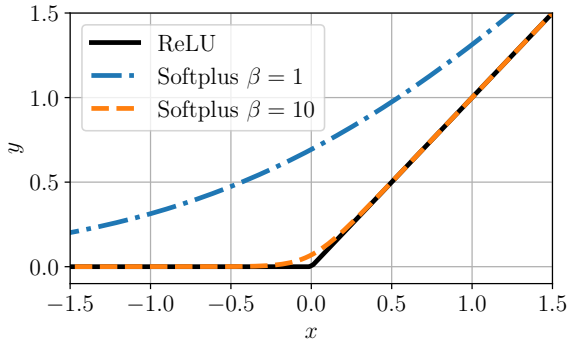


Fig. 4.2.: The ReLU activation can be approximated by the softplus activation with large β value.

4.2. Experiments

In this section we evaluate experimentally to which extend explanation methods can be manipulated using the approach described in the previous section. We demonstrate that many popular explanation methods are affected, including Gradient, Gradient×Input, Integrated Gradients, Guided Backpropagation, Layer-wise Relevance Propagation and Pattern Attribution. These explanation methods span two sub-categories of attribution methods, namely gradient-based explanations and propagation-based explanations. We refer to Section 2.1.2 for more detailed information on each of these methods.

We apply our algorithm from Section 4.1 to 100 randomly selected images for each explanation method. For each run, we select two images from the test set. One image serves to generate the target explanation map h^t. The other image is perturbed by our algorithm with the goal of replicating the target h^t using a few hundred iterations of gradient descent. We sum over the absolute values of the colour channels of the

explanation map to get the relevance per pixel. For a detailed description of the hyperparameters, see Appendix A.

For an example we refer to Figure 4.3 where we apply our method to an image of a dog while the target explanation is based on a cat image. For all explanation methods, the target map is closely reproduced and the perturbation of the dog image is small.

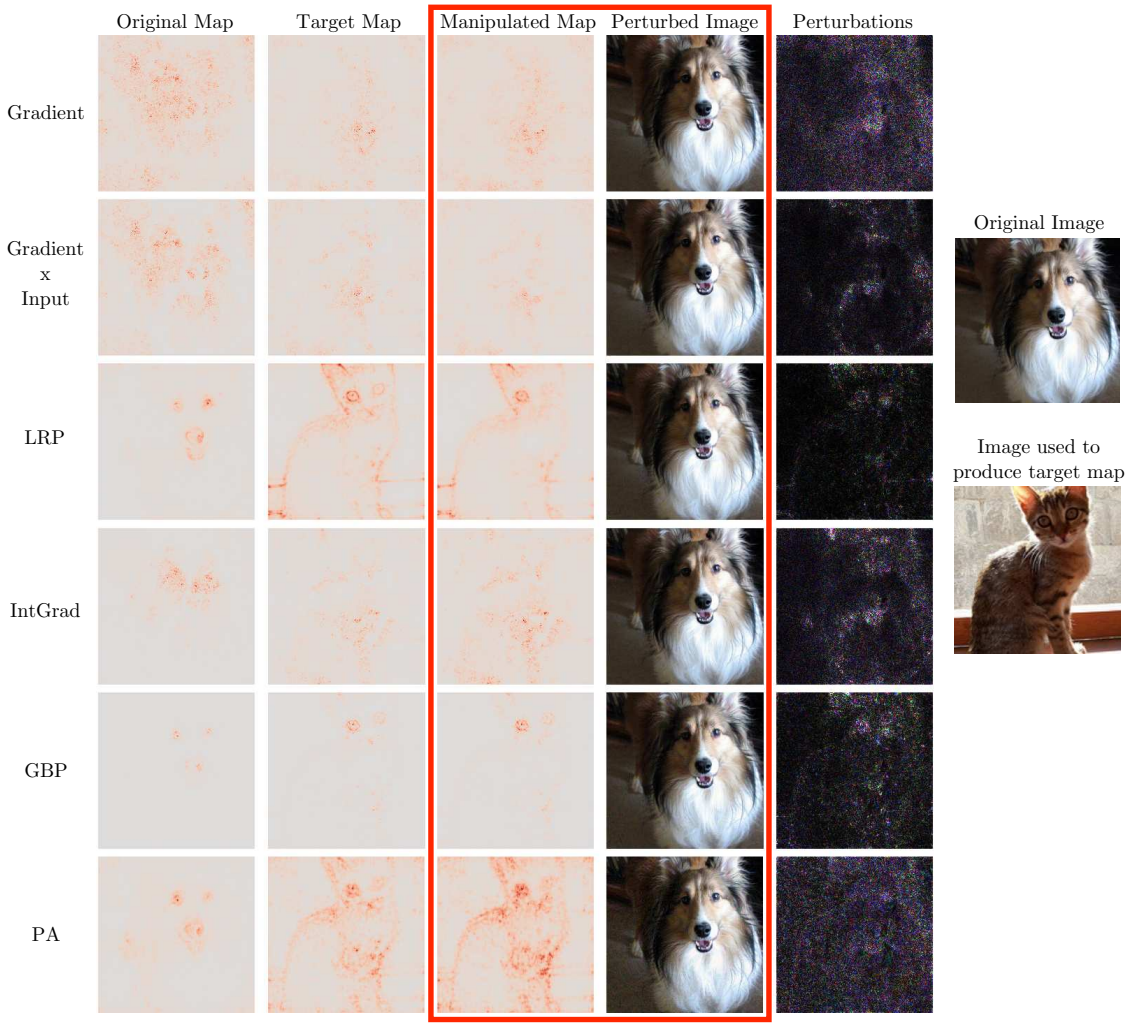


Fig. 4.3.: The explanation map of the cat is used as the target and the image of the dog is perturbed. The first column corresponds to the original explanations of the unperturbed dog image. The target map, shown in the second column, is the corresponding explanation of the cat image. The red box contains the manipulated images and the corresponding explanations. The last column visualizes the perturbations.

4.2.1. VGG

We perform our main experiments using a pre-trained VGG16 network [159] and images from the ImageNet data set.

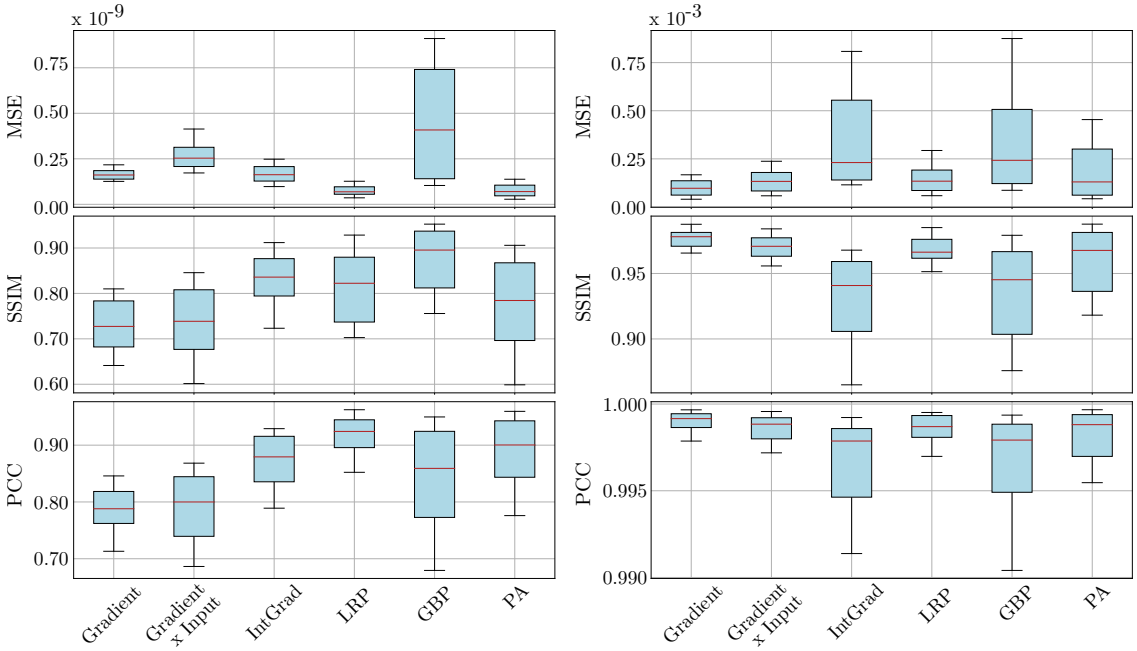


Fig. 4.4.: Left: Similarity measures between target h^t and manipulated explanation map $h(x_{\text{adv}})$. Right: Similarity measures between original image x and perturbed image x_{adv}. For SSIM and PCC large values indicate high similarity while for MSE small values correspond to similar images. For fair comparison, we use the same 100 randomly selected images for each explanation method. Boxes denote 25^{th} and 75^{th} percentiles, whiskers denote 10^{th} and 90^{th} percentiles, and solid lines show the medians.

Figure 4.3 illustrates qualitatively how our method works for different explanation methods on one example image. For a quantitative assessment, Figure 4.4 summarizes statistics for all 100 runs of our method using three different similarity metrics, namely the mean squared error (MSE), the structural similarity index measure (SSIM), and the Pearson correlation coefficient (PCC) (see Section 2.3 for an introduction to these similarity metrics). We show the similarity measures between the target h^t and the manipulated explanation map $h(x_{\text{adv}})$ as well as between the original image x and perturbed image x_{adv}. All considered metrics show that the perturbed images have an explanation closely resembling the target explanations. At the same time, the perturbed images are very similar to the corresponding original images. In addition, the output of the neural network is approximately unchanged by the perturbations, i.e. the classification of all examples is unchanged and the median of $\|f(x_{\text{adv}}) - f(x)\|$ is of the order of magnitude 10^{-3} for all explanation methods. We also verified by visual inspection that the target map is closely reproduced while the image is minimally perturbed. We have uploaded the results of all runs so that interested

readers can assess their similarity themselves[1] and published our code[2] to reproduce them.

4.2.2. Additional architectures and data sets

Manipulable explanations are not only a property of the VGG16 network. In this section, we show that our algorithm to manipulate explanations can also be applied to other architectures and data sets.

For the experiments, we optimize the same loss function (4.1) as before. We analyse the explanation's susceptibility to manipulations for the pre-trained ResNet [9], AlexNet [160] and DenseNet [161] architectures and observe that our approach generalizes well. Figure 4.5 shows how the Gradient explanation is manipulated for the different network architectures.

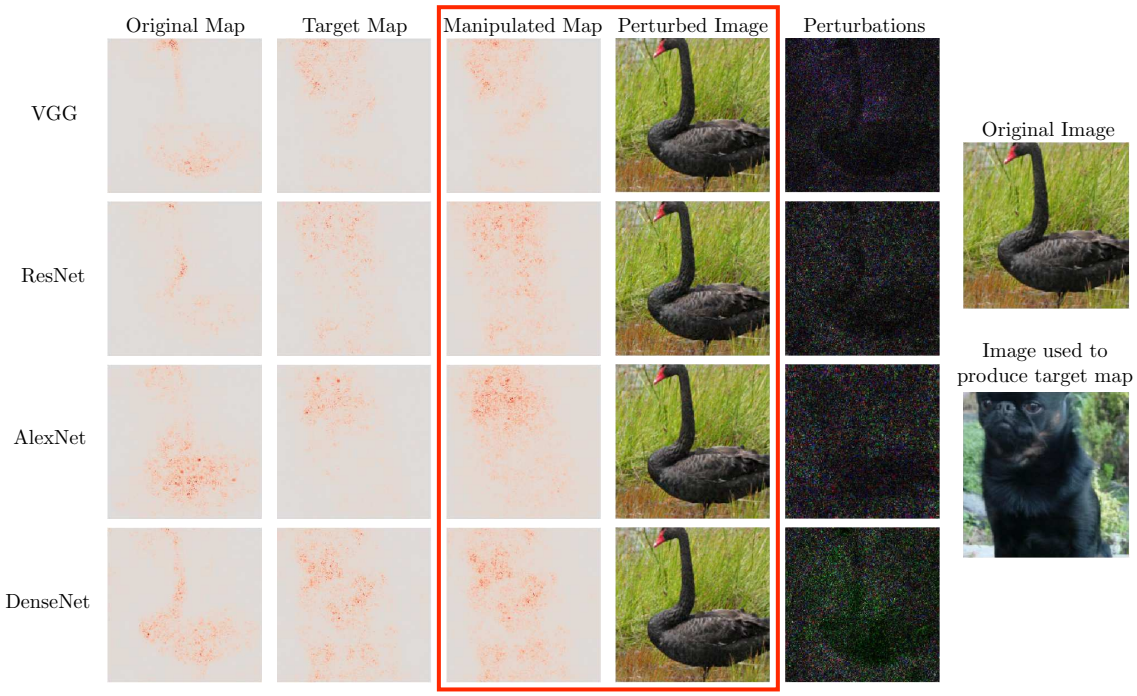


Fig. 4.5.: Manipulating the Gradient explanation maps of different network architectures.

Our method is also applicable to other data sets. To show this we trained the adapted VGG16 architecture on the CIFAR10 data set, reaching a test accuracy of approximately 92%. We then use our algorithm to manipulate the explanations for the LRP method. Just as for conventional adversarial attacks, perturbations become

[1] https://drive.google.com/drive/folders/1TZeWngoevHRuIw6gb5CZDIRrc7EWf5yb?usp=sharing

[2] https://github.com/pankessel/adv_explanation_ref

more visible when the input dimension is smaller. This can be seen for two examples in Figure 4.6.

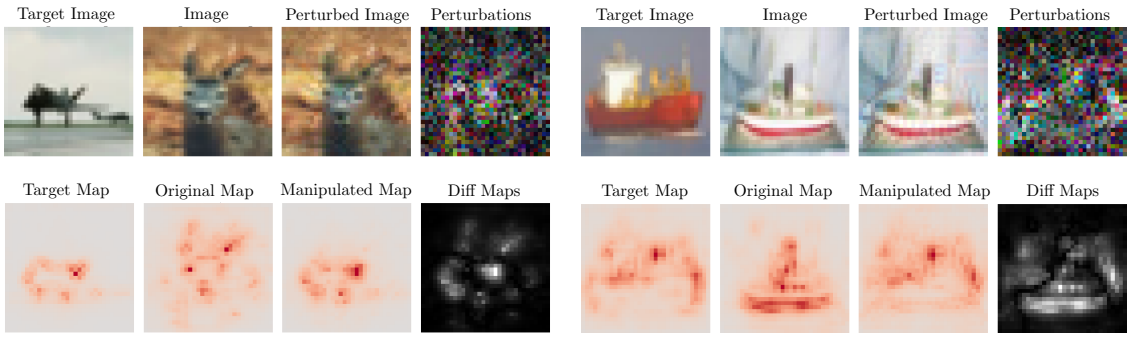


Fig. 4.6.: Manipulating the LRP explanation maps of a network trained on CIFAR10. Compared to higher dimensional (ImageNet) images, the perturbations are more visible. This is analogous to conventional adversarial attacks on low dimensional data.

4.2.3. Transferability

It is well-known that conventional adversarial examples (which aim to change the classification) can generalize across a wide variety of network architectures [162]. Adversarial examples which change the explanation map can generalize in two ways: firstly with respect to different explanation methods and secondly with respect to a change of the architecture of the network.

Transferability between methods In this section, we consider the following setup: we use the adversarial examples generated for the analysis of Section 4.2.1. We denote an adversarial example for a given method m by

$$x^m_{\text{adv}} \qquad m \in \{\text{Gradient, Gradient}\times\text{Input, IntGrad,...}\}. \tag{4.3}$$

We then calculate the explanation map of the manipulated image x^m_{adv} with respect to *another method*, i.e. $h^{m'}(x^m_{\text{adv}})$ with $m' \neq m$. We then analyse the correlation of this map with the target $h^t = h^{m'}(x^t)$, where x^t denotes the image used to generate the target map. If this correlation is high, it implies that adversarial examples of method m generalize to method m'.

Figure 4.7 (top) shows the results of the 100 runs using LRP to generate the adversarial examples, i.e. m = LRP. For GBP and PA, we observe a pronounced correlation between the manipulated map $h(x_{\text{adv}})$ and the target explanation map h^t, while the correlation with the original explanation map is small. This implies that the adversarial images of LRP generalize to GBP and PA. For Gradient, Gradient×Input and Integrated Gradients this is not the case. We see a decrease of the correlation

with the original explanation and the correlation with the target explanation map stays small. This may be due to the fact that the target explanation maps for these methods differ substantially from the LRP target explanation map and that these methods are gradient-based as opposed to propagation-based.

Figure 4.7 (bottom) shows the results for the 100 runs using the Gradient method to generate the adversarial image. For the method Gradient×Input, a small correlation with the target explanation map is visible, which points to the close relation of these two explanation methods as well as to the similarity between their explanation maps. Correlations for the other explanation methods are substantially lower. This suggest that while Gradient generalizes to Input×Gradient, it does not seem to generalize to the other methods.

These experiments may suggest a possible defence mechanism against adversarial attacks on the explanation, namely to combine gradient-based and propagation-based explanation methods to obtain more robust explanation maps (see also [163]).

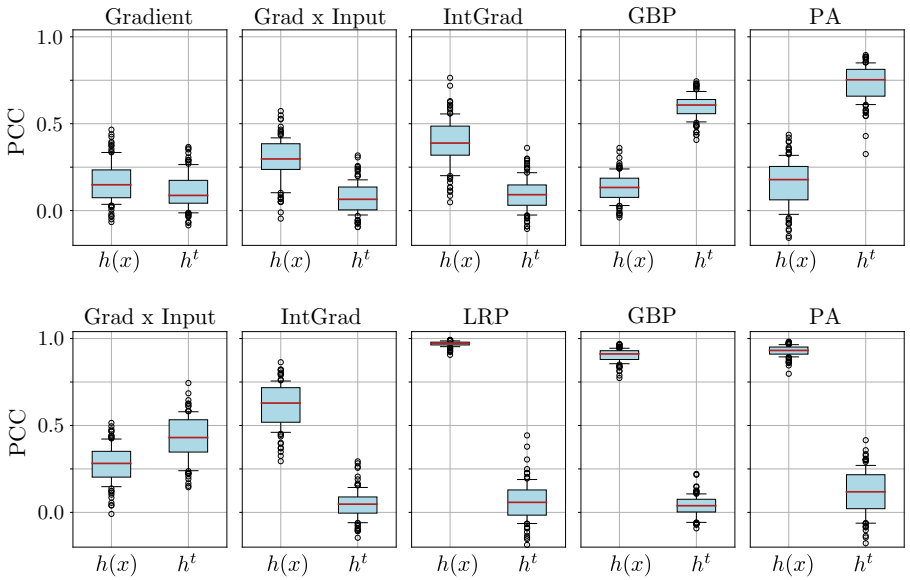


Fig. 4.7.: The adversarial images x_{adv} generated by using LRP (top) and Gradient (bottom) are used to calculate the explanation maps of all other methods. We show the correlations of the explanation map of the manipulated image $h(x_{\text{adv}})$ with the original explanation map $h(x)$ and with the target explanation map h^t for all explanation methods respectively.

Transferability between architectures We again consider the adversarial examples generated for the analysis of Section 4.2.1 using a VGG16 network and assess how well they generalize to the AlexNet architecture with pre-trained weights [160]. The explanation maps for VGG16 and AlexNet for a given unperturbed image look fairly similar, especially for propagation-based methods. However, Figure 4.8 shows no significant correlation between the explanation map when using AlexNet

in combination with x_{adv} (generated with VGG) and the target h^t for any of the examined methods. This indicates that the adversarial examples calculated with VGG16 do not generalize well to AlexNet. This is despite the fact that AlexNet and VGG16 architectures are relatively similar and the predictions for the manipulated images stay close to those for the original images.

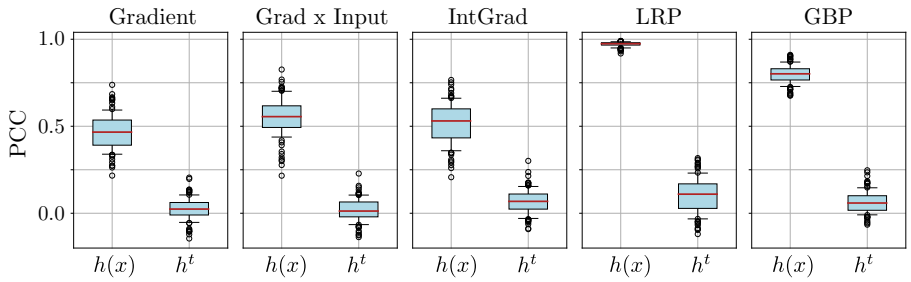


Fig. 4.8.: The adversarial images x_{adv} of all methods generated with VGG16 are used to calculate the explanation maps on AlexNet for the respective methods. We show the correlations of the explanation map of the manipulated image $h(x_{\text{adv}})$ with the original explanation map $h(x)$ and with the target explanation map h^t for all explanation methods respectively.

4.3. Theoretical considerations

In this section, we analyse the vulnerability of explanations theoretically. We argue that the susceptibility to manipulation can be related to the large curvature of the output manifold of the neural network. We start with an intuitive discussion of the gradient method, before developing mathematically precise statements.

We have demonstrated that one can drastically modify the explanation map while keeping the output of the neural network constant, adding only a small perturbation δx to the input, i.e.

$$f(x + \delta x) = f(x) = c\,. \tag{4.4}$$

The perturbed image $x_{\text{adv}} = x + \delta x$ therefore lies on the hypersurface of constant network output $S = \{p \in \mathbb{R}^N | f(p) = c\}$[3]. As the Gradient method (and, usually, attribution methods in general) only depends on the winning class output, we can exclusively consider this component of the output, i.e. $f(x) := f(x)_k$ with $k = \arg\max_i f(x)_i$. Therefore, the hypersurface S is of co-dimension one. The gradient ∇f for every $p \in S$ is normal to this hypersurface, since it is the direction of steepest ascent. The fact that the normal vector ∇f can be drastically changed by slightly perturbing the input along the hypersurface S suggests that the curvature of S is large.

[3]It is sufficient to consider the hypersurface S in a neighbourhood of the original input x.

While the above statement sounds intuitive, it requires non-trivial concepts of differential geometry to make it precise, in particular the notion of the second fundamental form (see Section 3.2.6). To this end, it is advantageous to consider a normalized version of the gradient method

$$n(x) = \frac{\nabla f(x)}{\|\nabla f(x)\|}\,. \tag{4.5}$$

This normalization does not change the relative importance of any pixel with respect to the others and is thus merely conventional. For any point $p \in S$, we define the tangent space T_pS as the vector space spanned by the tangent vectors $\tau'(0) = \frac{d}{dt}\tau(t)|_{t=0}$ of all possible curves $\tau : \mathbb{R} \to S$ with $\tau(0) = p$. For $u, v \in T_pS$, we denote their inner product by $\langle u, v\rangle$. For any $u \in T_pS$, the *directional derivative* of a function f is uniquely defined for any choice of τ by

$$D_u f(p) = \frac{d}{dt} f(\tau(t))\bigg|_{t=0} \qquad \text{with} \qquad \tau(0) = p \;\text{ and }\; \tau'(0) = u. \tag{4.6}$$

We then define the *Weingarten map* as[4]

$$L : \begin{cases} T_pS & \to\; T_pS \\ u & \mapsto\; -D_u n(p)\,, \end{cases}$$

where the unit normal $n(p)$ can be written as (4.5). This map quantifies how much the unit normal changes as we infinitesimally move away from p in the direction u. The *second fundamental form* is then given by

$$\mathcal{L} : \begin{cases} T_pS \times T_pS & \to\; \mathbb{R} \\ u, v & \mapsto\; -\langle v, L(u)\rangle = \langle v, D_u n(p)\rangle\,. \end{cases}$$

It can be shown that the second fundamental form is bilinear and symmetric $\mathcal{L}(u, v) = \mathcal{L}(v, u)$. It is therefore diagonalizable with real eigenvalues $\lambda_1, \ldots \lambda_{d-1}$ which are called *principal curvatures*.

We have thus established the remarkable fact that the sensitivity of the gradient map (4.5) is described by the principal curvatures, a key concept of differential geometry.

This observation allows us to derive an upper bound on the maximal change of the gradient map $h(x) = n(x)$ as we move slightly on S. To this end, we define the *geodesic distance* $d_g(p, q)$ of two points $p, q \in S$ as the length of the shortest curve on S connecting p and q. In the supplement A.1, we prove the following theorem:

Theorem 1. *Let $f : \mathbb{R}^N \to \mathbb{R}$ be a network with softplus$_\beta$ non-linearities and $\mathcal{U}_\epsilon(p) = \{x \in \mathbb{R}^N; \|x - p\| < \epsilon\}$ an environment of a point $p \in S$ such that $\mathcal{U}_\epsilon(p) \cap S$*

[4]The fact that $D_u n(p) \in T_pS$ follows by taking the directional derivative with respect to u on both sides of $\langle n, n\rangle = 1$.

is fully connected. Let f have bounded derivatives $\|\nabla f(x)\| \geq c$ for all $x \in \mathcal{U}_\epsilon(p) \cap S$. It then follows for all $p_0 \in \mathcal{U}_\epsilon(p) \cap S$ that

$$\|h(p) - h(p_0)\| \leq |\lambda_{max}|\, d_g(p, p_0) \leq \beta\, C\, d_g(p, p_0), \tag{4.7}$$

where λ_{max} is the principal curvature with the largest absolute value for any point in $\mathcal{U}_\epsilon(p) \cap S$ and the constant $C > 0$ depends on the weights of the neural network.

This theorem can intuitively be motivated as follows: for ReLU non-linearities, the lines of equal network output are piece-wise linear and therefore have kinks, i.e. points of divergent curvature. These ReLU non-linearities are well approximated by softplus non-linearities (4.2) with large β. Reducing β smooths out the kinks and therefore leads to reduced maximal curvature, i.e. $|\lambda_{\max}| \leq \beta\, C$. For each point on the geodesic curve connecting p and p_0, the normal can at worst be affected by the maximal curvature, i.e. the change in explanation is bounded by $|\lambda_{\max}|\, d_g(p, p_0)$.

There are two important lessons to be learned from this theorem: Firstly, the geodesic distance can be substantially greater than the Euclidean distance for curved manifolds (see Figure 4.9). In this case, inputs which are very similar to each other, i.e. the Euclidean distance is small between the original and the adversarially perturbed input, can have explanations that are drastically different. Secondly, the upper bound is proportional to the β parameter of the softplus non-linearity. Therefore, all else equal, smaller values of β provably result in increased robustness with respect to manipulations.

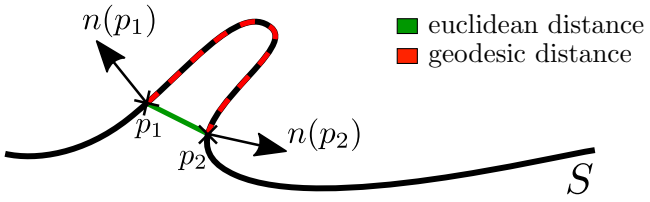


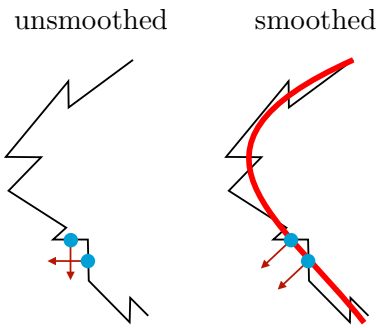
Fig. 4.9.: Two points p_1 and p_2 on a manifold S can have a large (geodesic) distance *along* the manifold even if the Euclidean distance is small. The normal vectors n, and thus the corresponding (Gradient) explanations at the points, can be drastically different.

4.4. Smoothing explanations

After showing to what extend different explanation methods are susceptible to adversarial perturbations experimantally, and analysing the problem of manipulability of explanation methods theoretically, we are now interested in finding ways to mitigate this behavior.

Based on our theoretical analysis in the previous section, we propose a modification that can be applied to any of the considered explanation methods, and results in increased robustness against adversarial attacks on the explanation.

Using the fact that the upper bound (4.7) is proportional to the β parameter of the softplus non-linearities, we propose *β-smoothing*. This method calculates an explanation using a network for which the ReLU non-linearities are replaced by softplus activations with a small β parameter to smooth the principal curvatures (see figure on the right for an intuition). The precise value of β is a hyperparameter of the method, but we find that a value around $\beta = 1$ works well in practice.



We emphasize that we use the substituted activation functions *only* for the explanation process. The prediction and therefore the winning class for which we calculate the explanation is determined using the original ReLU network.

There exist a relationship between our proposed β-smoothing and the SmoothGrad [28] explanation method.

Theorem 2. *For a one-layer neural network $f(x) = ReLU(w^T x)$ and its β-smoothed counterpart $f_\beta(x) = softplus_\beta(w^T x)$, it holds that*

$$\mathbb{E}_{\epsilon \sim p_\beta} \left[\nabla f(x + \epsilon) \right] = \nabla f_{\frac{\beta}{\|w\|}}(x) \,, \tag{4.8}$$

where $p_\beta(\epsilon) = \frac{\beta}{(e^{\beta\epsilon/2} + e^{-\beta\epsilon/2})^2}$.

Since $p_\beta(x)$ closely resembles a normal distribution with variance $\sigma = \log(2)\frac{\sqrt{2\pi}}{\beta}$, β-smoothing can be understood as the $N \to \infty$ limit of SmoothGrad. We provide the full proof of the above theorem in Appendix A.1.2. We emphasize that this theorem only holds for a one-layer neural network, but for deeper networks we empirically observe that both explanation methods, SmoothGrad and β-smoothing, lead to visually similar maps as they are considerably less noisy than the Gradient explanation. The theorem therefore suggests that SmoothGrad can similarly be used to smooth the curvatures and can thereby make explanations more robust.

4.4.1. Experiments

We can use β-smoothing in several different scenarios. If an attacker manipulated the image in order to evoke a specific explanation, β-smoothing allows us to recover the original explanation map by substituting the ReLU activations with softplus and decreasing the value of the β parameter. Figure 4.10 demonstrates this for the explanation methods Gradient and LRP on one example. In Figure 4.11, we see that this also holds when evaluated quantitatively: with decreasing β value the PCC

between the target map and the explanation of the manipulated image (shown in red) shrinks while the PCC between the original map and the explanation map of the manipulated image (shown in green) increases. For LRP, the PCC between original and manipulated explanation approaches one for small β values, since the β-smoothed LRP explanations are very similar to the original LRP explanations. For the Gradient explanation we do not reach a PCC of one as the β-smoothed map is less noisy than the original Gradient map we compare it with (see Figure 4.10 for an intuition).

We stress that this works for all considered methods. We also note that the same effect can be observed using SmoothGrad by successively increasing the standard deviation σ of the noise distribution (see Figure A.6). This further underlines the similarity between the two smoothing methods.

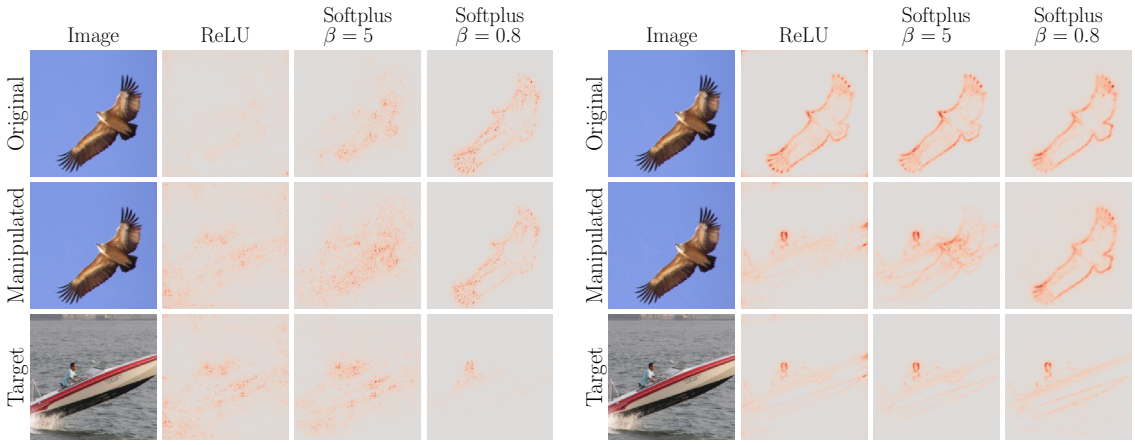


Fig. 4.10.: Network input and the respective explanation maps as β is decreased for Gradient (center) and LRP (right).

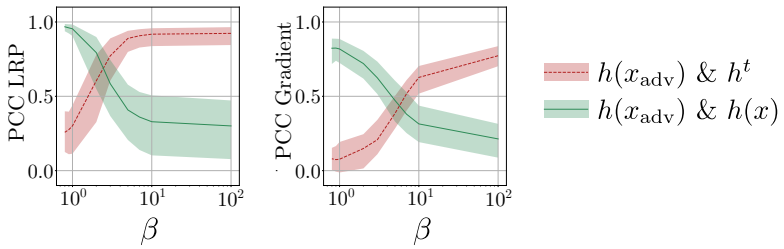


Fig. 4.11.: β dependence for the correlations of the manipulated explanation (here Gradient and LRP) with the target and original explanation. For small β values the similarity between $h(x_{\text{adv}})$ and $h(x)$ is recovered. Lines denote the medians, 10^{th} and 90^{th} percentiles are shown in semitransparent color.

If an attacker knew that smoothing was used to undo the manipulation, they could try to attack the smoothed method directly. However, both β-smoothing and SmoothGrad are substantially more robust than their non-smoothed counterparts as they do

not suffer from drastically varying Gradients when moving along the hypersurface of constant network output. We confirm this by running adversarial attacks on the β-smoothed and SmoothGrad explanations for 100 images each. Figure 4.12 shows that the correlation between the manipulated β-smoothed explanations and the target maps is on average much lower than the correlation between the unsmoothed Gradient explanation and the respective target maps. For SmoothGrad we observe similar or slightly higher correlation between manipulated and target map showing that both smoothing methods offer similar protection against adversarial attacks on the explanation.

It is important to note that β-smoothing achieves this at considerably lower computational cost: β-smoothing only requires a single forward and backward pass, while SmoothGrad requires as many as the number of noise samples (typically between 10 and 50).

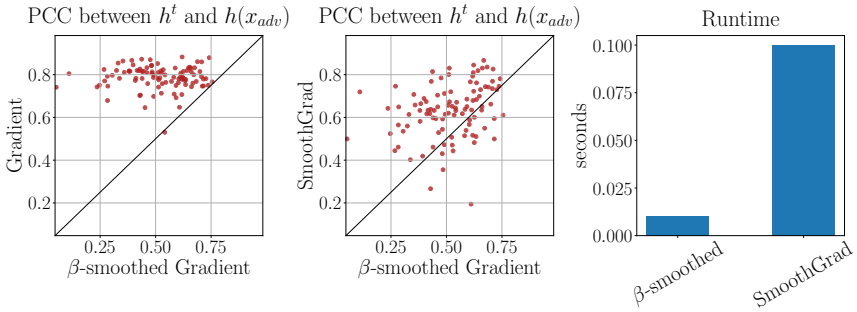


Fig. 4.12.: Left: markers are clearly left of the diagonal, i.e. explanations are more robust to manipulations when β-smoothing is used. Center: SmoothGrad has comparable results to β-smoothing, i.e. markers are distributed around the diagonal. Right: β-smoothing has significantly lower computational cost than SmoothGrad.

We can further analyse the performance of Gradient, β-smoothed Gradient, and SmoothGrad by using the pixel flipping metric [26,164]. This framework progressively removes information from the image by replacing the RGB pixel value with a predefined value (we use zero). The order, in which pixel values are replaced, is defined by the explanation map, so that the most relevant pixels are replaced first.

Figure 4.13 demonstrates that β-smoothing leads to better performance than the Gradient method and to comparable performance with SmoothGrad on the pixel-flipping metric [26, 164]. This suggests that β-smoothing is not only more robust against adversarial attacks but also more reliable at highlighting features that are truly relevant for the prediction.

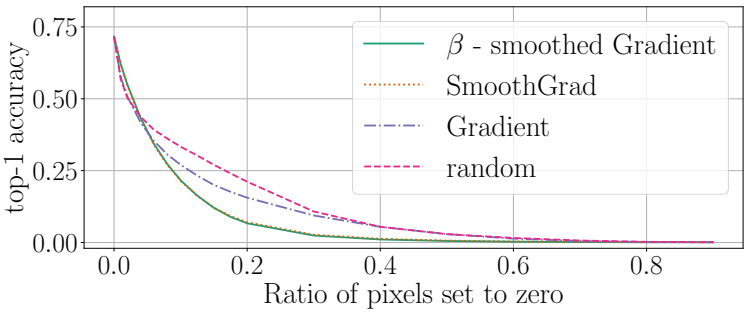


Fig. 4.13.: Pixel flipping performance compared to random baseline (the lower the accuracy the better the explanation): the metric sorts pixels of images by relevance and incrementally sets the pixels to zero starting with the most relevant. In each step, the network's performance is evaluated on the complete ImageNet test set.

4.5. Related work

The concept of manipulating inputs to neural networks in a targeted way was first introduced to alter a neural networks prediction [36, 162, 165]. These so called *adversarial attacks* focus on altering the networks prediction adding imperceptible perturbations to the image. While conventional adversarial attacks require computation of the gradient of the prediction with respect to the input, that same gradient can itself be considered as an explanation for the network prediction [23, 24]. In order to manipulate a gradient-based explanation map, one thus needs to compute second derivatives. Together with the fact that the input and thus the explanation have generally much higher dimension than the network prediction makes our findings a non-trivial extension to conventional adversarial attacks.

With rising popularity of explanation methods, research on their properties has also emerged. In the remainder of this section we give an overview of works that have investigated the robustness against manipulation of such methods.

In [17], Ghorbani et al. demonstrate that explanation maps can be sensitive to small perturbations in the image which lead to an unstructured change in the explanation map. They compare several different approaches, which all rely on perturbing the input to a neural network and measuring the resulting change in explanation: adding random perturbations serves as baseline, the top-k attack decreases the importance of the k most relevant features, the mass-center attack aims to move the center of relevance as far away from the original center as possible, and the targeted attack maximizes the relevance in a predefined region.

Our method optimizes a much more specific objective, namely to accurately reproduce a target map on a pixel-by-pixel basis. Compared to our method, Ghorbani et al. can thus be considered as evoking *unstructured* changes in the explanation. Furthermore, their attacks only keep the classification result the same which often leads to significant

changes in the network output (see [17], Figure 1). It is therefore not clear from their analysis whether the explanation *or* the network is vulnerable (and the explanation map simply reflects the relevance of the perturbation faithfully). Our method on the other hand keeps the output of the network (approximately) constant, which we also use as a basis for our theoretical analysis in terms of differential geometry.

Zhang et al. [166] take a very similar approach to that of Ghorbani et al. In contrast to our approach their attack aims to manipulate the prediction as well as the attribution map for an example input. More specifically, in all their examples the explanation is kept constant although the prediction changes. It is thus not clear from their analysis if the explanation can be changed *arbitrarily*.

Subramanya et al. [167] use an adversarial patch (analogous to many conventional adversarial attacks) to fool the classifier and the explanation method. The approach is different to Zhang et al. and Ghorbani et al. since in this case the region of the adversarial patch is solely responsible for the misclassification, which is also apparent in the explanation when using conventional attacks. Their modification then tricks the explanation into highlighting areas of the input apart from the adversarial patch, and thus clearly not relevant for the prediction. In contrast to our work, the patches make the manipulated images very clearly identifiable and the attack on the prediction is again mixed with the attack on the explanation, blending the two effects.

Yeh et al. [168] analyse fidelity and sensitivity of explanations pointing out that a good explanation should accurately capture to which degree the prediction changes in response to significant perturbations (fidelity) but not be overly sensitive to small perturbations that do not change the prediction. Unlike our work, their added perturbations are not adversarial, as in they are not targeted to change the explanation in a specific way.

Other approaches modify the network's parameters to produce a change in the explanation. In [18], explanation maps are changed by randomization of (some of) the network weights and in [169] the complete network is fine-tuned to produce manipulated explanations while the accuracy remains high. Another approach [158] finds alternative weights that provably keep the predictions constant on any input data, but drastically alter the explanation. Kindermans et al. [127] manipulate input and network parameters by adding a constant shift to the input image, which is then eliminated by changing the bias of the first layer. For some methods, this leads to a change in the explanation map. Many other works [170–172] research how to manipulate an explanation by modifying the model parameters in a similar way. These findings are concerning in a setting where biased models might be deployed and we hope to use explainability to uncover these biases, since such biases can be hidden and thus go undetected in the explanation [173].

These approaches are different from our method as they do not aim to change the explanation to a specific target explanation map but instead modify the parameters

of the network, effecting all explanations.

Susceptibility of explanations to manipulation is not only a problem in the setting of image classification but has also been shown to be problematic in domains such as natural language processing [174, 175] and deep reinforcement learning [176].

Some explanations [28, 65] rely on input perturbation to create the explanation map and are thus naturally more robust to input manipulation [177].

A few works [178, 179] do not directly improve robustness of explanations but aim to model uncertainty in explanations to enable the user to make informed decisions on the trustworthiness.

Methods to counteract attacks on the explanations have been proposed in various works [37, 38, 158, 163]. Within this thesis we investigate two approaches towards better robustness of explanations: one based on the explanation process in Section 4.4 and another based on robust training of neural networks in Chapter 5.

4.6. Limitations

Attacks on the explanation have larger limitations than conventional adversarial attacks, mostly due to the much higher dimensional target domain.

Generalization As discussed in Section 4.2.3, our manipulated images do not generalize well between architectures or explanation methods. We suspect that this is due to the more complex task of altering the complete explanation which has the same dimension as the input and is thus typically much higher dimensional than the output vector. Explanation maps for different explanation methods also significantly diverge, in contrast to targets for a conventional adversarial attack, which renders generalization difficult. The lack of generalization can be used to defend against adversarial attacks on the explanation for example by averaging explanations over several network architectures or explanation methods [163]. Although we do not test this explicitly, we expect that the poor generalization of the manipulated images also makes the success of a black box attack, i.e. where one does not have access to the model and cannot directly compute the gradient, less likely.

Three channel explanations We test our manipulation method on three channel explanations and found those much harder to manipulate. For three channel attacks on gradient based explanations we end up with explanations that do not closely reproduce the target map. For propagation based methods we got mixed results: for LRP and Pattern Attribution the explanation maps are accurately reproduced while for GBP we observe stronger perturbations in the manipulated image. We show some examples in the appendix, Figure A.4. As explanations are usually used for visual inspection, structural similarity (which can be achieved with attacks on

the sum over absolute channel values) seems more relevant than the reproduction of exact color channel values. Therefore the reduced potency of our method for three channel explanations does not significantly reduce its relevance.

Robustness of the manipulated explanation In Section 4.4.1 we show how the original explanation can be recovered using β-smoothing, and in Appendix A.4 we show the same for SmoothGrad. While these two methods work exceptionally well for the recovery of the original explanation, the manipulated explanation map is in general not very robust, meaning any additional perturbation applied to the manipulated image can lead to (partial) recovery of the original explanation. Therefore, we expect approaches analogous to defenses against conventional adversarial attacks to be a lot more effective when defending against an attack on the explanation. Again, we suspect this is due to the much higher dimensional target domain requiring more complex structured adversarial perturbations.

Evaluation of β-smoothed explanations We evaluate our proposed explanation, on the one hand, by pointing out the similarity (and in the case of one-layer neural networks the equivalence) to the broadly accepted SmoothGrad explanation and, on the other hand, using the pixel flipping metric. The latter is widely used but comes with a few drawbacks. For example, one can replace pixels with other values than zero (for example averages) which might lead to different curves for the accuracy. Another problem is that images with large areas of replaced pixels are not part of the data distribution the network was originally trained on. The behavior of the network on these out of distribution data might therefore be very different. A related method [180] thus proposes retraining the network on the modified images. We abstain from evaluating our explanations with this method due to the high computational cost.

Limitations of our theoretical results Our theoretical analysis focuses on the Gradient explanation, i.e. we derive our upper bound (4.7) for $h(x) = \nabla f(x)$. We do not explicitly derive upper bounds on the change in explanation for other explanation methods. We expect that our derived counter measure, β-smoothing, has positive effects on the robustness of all considered explanations, since they depend on the gradient. However, we only experimentally verify these assumptions. Furthermore, we establish the relation between β-smoothing and SmoothGrad only for one-layer networks.

4.7. Summary

Explanation methods have recently become increasingly popular among practitioners. In this chapter, we showed that dedicated imperceptible manipulations of the

input data can yield arbitrary and drastic changes in the explanation map. We demonstrated both qualitatively and quantitatively that explanation maps of many popular explanation methods can be arbitrarily manipulated like this. Crucially, this can be achieved while keeping the model's output (approximately) constant. Not being able to trust an explanation is problematic if we hope to gain insights into the decision-making processes of deep neural networks by studying these explanations.

Our theoretical analysis revealed that the large curvature of the network's decision function is one important culprit for this unexpected vulnerability. Using this theoretical insight, we can profoundly increase the resilience to manipulations by smoothing *only* the explanation process while leaving the prediction process of the model itself unchanged. To this end, we introduce β-smoothing, for which we substitute the ReLU activations with softplus units with a small β value to calculate the explanation map. We establish a connection to the well-known SmoothGrad explanation theoretically and confirm the similarities experimentally. The resulting β-smoothed explanations deliver smoother explanation maps and are more robust against adversarial attacks.

5. Towards networks with robust explanations

In the previous chapter we discussed the unexpected susceptibility of explanations to adversarial attacks. We showed that by adding imperceptible perturbations to the input the explanation maps of many popular explanation methods can be arbitrarily manipulated. We analysed this phenomenon theoretically and proposed a modification to the explanation process to mitigate the effects.

In this chapter, our focus lies on exploring how to make neural network models themselves more robust against explanation manipulation. More specifically, we analyse the difference between the original and the manipulated explanation maps theoretically and provide a unified framework which allows us to derive bounds on the maximal change in explanation map. Based on these theoretical insights, we derive several techniques to make neural networks more resilient against attacks on the explanation: We regularize during training using *weight decay*, we train with *smoothed activation functions* and we *minimize the Hessian norm* of the network with respect to the input during the training process.

All these methods reduce the curvature of the output manifold in different ways. This leads to more stable gradients when moving along the output manifold. As most explanation methods rely on the gradient for the explanation map, this effects their robustness to input perturbation positively.

We emphasize that our robustness enhancing interventions lead to increased robustness for *all* explanation methods.

We demonstrate the effectiveness of our proposed methods experimentally for several different explanation methods and network architectures on the CIFAR10 and on the ImageNette data set. Our main experiments focus on the gradient explanation and random input perturbations to minimize secondary effects of hyperparameters, which can appear for targeted adversarial attacks. However, we also show that our methods lead to increased robustness against adversarial attacks on the explanation.

Figure 5.1 provides an intuition for why explanations are susceptible to manipulation and how our methods lead to more robust explanations.

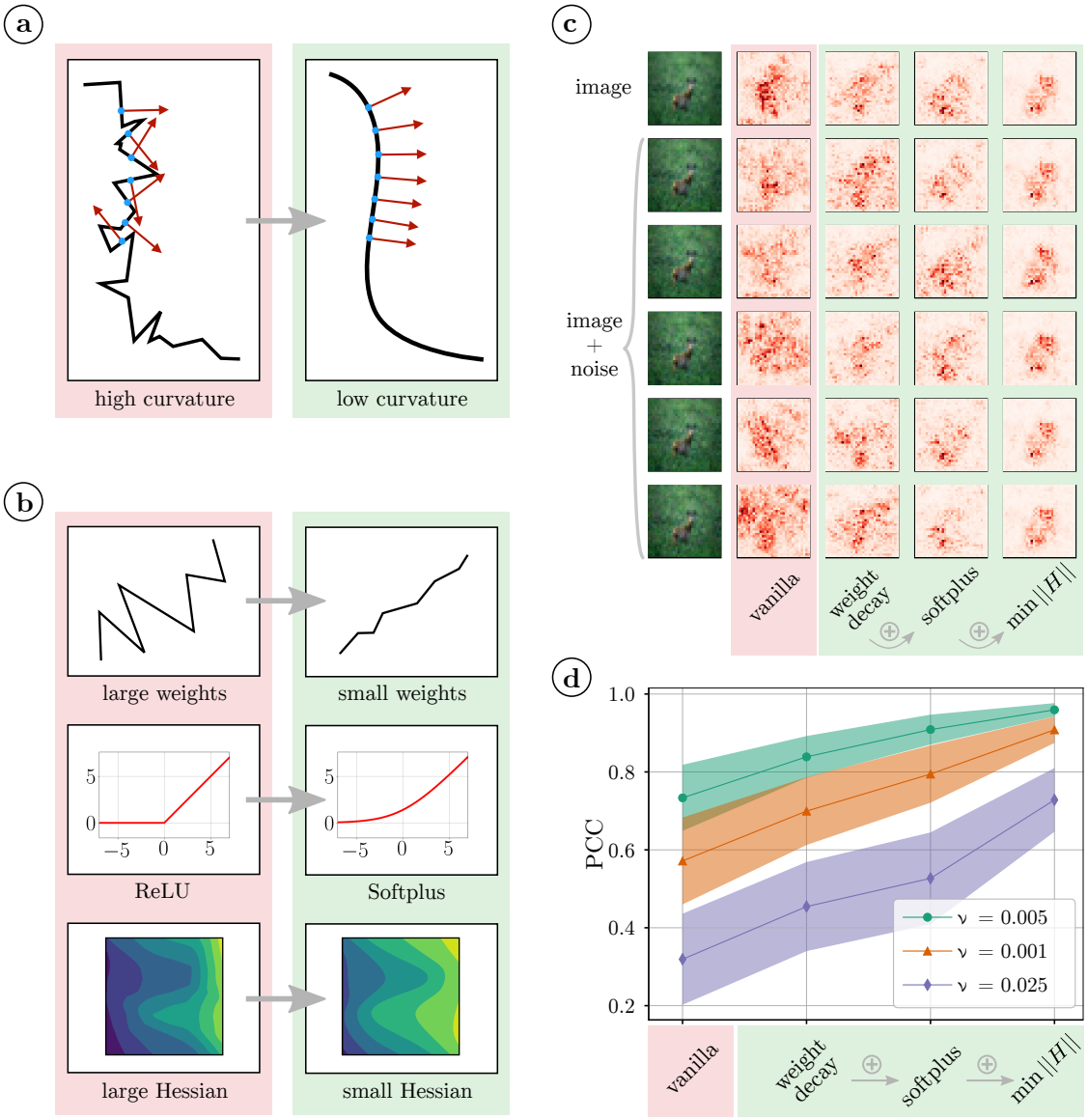


Fig. 5.1.: Intuition and results for our approach. **a)** The gradient (red arrows) changes drastically when moving along a line with high curvature but changes only gradually when the curvature is low. **b)** We propose several techniques that reduce curvature when incorporated in the training procedure. Weight decay flattens the angles between piecewise linear functions, softplus smooths out the kinks of the ReLU function, Hessian minimization reduces curvature locally at the data points. **c)** For the vanilla net the gradient explanation maps differ strongly. For networks trained with a combination of our proposed methods the explanation maps become robust to the input perturbations. **d)** A quantitative analysis on the complete test set confirms our theoretical findings. The similarity between original explanation and explanation of a perturbed input is significantly higher for networks trained with our methods. We show results for three different noise levels ν.

5.1. Theoretical considerations

In the previous chapter we derived an upper bound on the change in explanation map (4.7) that depends on the maximal principal curvature and the geodesic distance between the original and the manipulated input. In this section, we derive three efficient ways to make neural networks more robust, namely by applying weight decay, by using smooth activation functions and by directly regularizing the Hessian norm. Before we address these effective counter measures, we recap briefly how we derive an intermediate result for (4.7) that depends on the Hessian norm $||H(f)||$ of the neural network f.

We consider the gradient explanation for concreteness and restrict to the output of the winning class, i.e. $f(x) := f(x)_k$ with $k = \arg\max_i f(x)_i$, since the Gradient method only depends on this component of the output. To manipulate the explanation of an input $x \in \mathbb{R}^N$ of a classifier $f : \mathbb{R}^N \to \mathbb{R}$, we construct a perturbed input $x_{\text{adv}} = x + \delta x$ such that the output of the network is (approximately) unchanged, i.e.

$$f(x) \approx f(x_{\text{adv}}) \tag{5.1}$$

but the corresponding (Gradient) explanations $h = \nabla f$ are drastically different, i.e.

$$||h(x) - h(x_{\text{adv}})|| \gg 1\,. \tag{5.2}$$

Typically, the perturbation is assumed to be small, i.e. $||\delta x|| \ll 1$, such that it is imperceptible. For a theoretical analysis, we would like to derive upper bounds on the change of Gradient map $||h(x) - h(x + \delta x)||$ by any such perturbation δx. To this end, we consider a curve $\tau : \mathbb{R} \to \mathbb{R}^N$ with affine parameter t connecting the unperturbed data point x with its perturbed counterpart x_{adv}, i.e.

$$\tau(t = -\infty) = x\,, \qquad\qquad \tau(t = +\infty) = x_{\text{adv}}\,. \tag{5.3}$$

One can then use the gradient theorem to rewrite the change in the j-th component of the explanation h as[1]

$$\begin{aligned} h_j(x) - h_j(x_{\text{adv}}) = \partial_j f(x) - \partial_j f(x_{\text{adv}}) &= \int_\tau \sum_i \partial_i \partial_j f(x)\, \mathrm{d}x_i \\ &= \int_{-\infty}^{\infty} \sum_i \partial_i \partial_j f(\tau(t))\, \tau_i'(t)\, \mathrm{d}t\,, \end{aligned} \tag{5.4}$$

Let the Frobenius norm of the Hessian $H_{ij}(f) = \partial_i \partial_j f$ be bounded, i.e.

$$||H(f)(x)|| \leq H^* \in \mathbb{R}_+\,, \qquad\qquad \forall x \in \mathbb{R}^N\,.$$

[1] Here we assume that the classifier f is twice differentiable. However, this assumption can, under certain circumstances, be relaxed as discussed in Section 5.1.1.

It then follows immediately that the maximal change in explanation is also bounded:

$$\begin{aligned} ||h(x) - h(x_{\text{adv}})|| &\leq \int_{-\infty}^{+\infty} ||H(f)\,\tau'(t)||\,\mathrm{d}t \\ &\leq H^* \int_{-\infty}^{+\infty} ||\tau'(t)||\,\mathrm{d}t = H^*\,L(\tau)\,, \end{aligned} \tag{5.5}$$

where $L(\tau) = \int_{-\infty}^{+\infty} ||\tau'(t)||\,\mathrm{d}t$ is the length of the curve τ. We have therefore deduced that bounding the Frobenius norm of the Hessian implies a bound on the maximal possible change in explanation by input manipulation.

Based on this upper bound, we propose three approaches to reduce the Frobenius norm of the Hessian and thereby increase the robustness of explanations against manipulation.

5.1.1. Weight decay

The first approach starts from the observation that the Frobenius norm of the Hessian depends on the weights of the neural network. More precisely, in Appendix B.1 we prove the following theorem.

Theorem 3. *Let $f : \mathbb{R}^N \to \mathbb{R}$ be a fully-connected neural network with L layers. The weights of the l-th layer are denoted by $W^{(l)}$ and its activation functions σ are twice-differentiable and bounded*

$$|\sigma'(x)| \leq \Sigma_1\,, \qquad\qquad |\sigma''(x)| \leq \Sigma_2\,. \tag{5.6}$$

The Hessian of the network is then bounded by

$$||H(f)||_F \leq \sum_{m=1}^{L} \left(\prod_{l=1}^{m} ||W^{(l)}||_F^2 \prod_{l=m+1}^{L} ||W^{(l)}||_F \right) \Sigma_1^{L+m-2}\,\Sigma_2\,. \tag{5.7}$$

As a practical consequence of the theorem, we can reduce the maximal possible change in explanation by decreasing the Frobenius norms of the weights. Motivated by this theoretical insight, we propose to use weight decay for training neural networks such that their explanations are more robust to manipulation.

Note while it is well-known that weight decay can improve generalization of neural networks [181–183], its effect on the manipulability of explanations has not previously been established.

Weight decay can be implemented by adding an additional term to the loss function

$$\mathcal{L} = \mathcal{L}_0 + \frac{\lambda}{2}||W||_2^2\,, \tag{5.8}$$

where $\mathcal{L}_0$ is the unregularized loss, W contains all network weights and λ is a hyper-parameter, influencing how strongly large weights are penalized. Implementations for weight decay can deviate slightly from (5.8) and other regularizations that reduce the weight norms (L^1-regularization, variants of L^2-regularization, etc [42]) may have similar effects.

Note on ReLU non-linearites For Theorem 3 we require twice differentiable activation functions. The popular ReLU non-linearity can be recovered from the twice-differentiable softplus activation in the limit $\beta \to \infty$. Note however that the bound (5.7) diverges in this limit since $\Sigma_2 \to \infty$, see (5.10). The fundamental underlying difficulty is that the second derivative $\mathrm{ReLU}''(x)$ is ill-defined at $x = 0$. In Appendix B.2, we therefore generalize the bound (5.7) to the case of ReLU non-linearities. For this, we use the fact that a distributional generalization of the second derivative of the ReLU non-linearity can be defined, i.e. $\mathrm{ReLU}''(x) = \delta(x)$ where δ denotes the Dirac distribution. The corresponding right-hand-side of this generalized bound only depends on the weights of the neural network. Thus, this result establishes theoretically that weight decay also certifiably improves robustness for ReLU non-linearities.

5.1.2. Smooth activation functions

As a second approach, we note that the bound of the network's Hessian (5.7) also depends on the maximal values of the activation function's first and second derivatives (5.6). Choosing activation functions whose derivatives can be bounded by smaller values will therefore lead to robuster explanations.

As a concrete example, consider the softplus activation function

$$\sigma(x) = \frac{1}{\beta} \ln(1 + e^{\beta x}) \,, \tag{5.9}$$

where $\beta \in \mathbb{R}_+$ is a hyperparameter. Its first and second derivative are bounded by

$$|\sigma'(x)| \leq 1 \,, \qquad\qquad |\sigma''(x)| \leq \frac{1}{4}\beta \,, \tag{5.10}$$

and thus $\Sigma_1 = 1$ and $\Sigma_2 = \frac{1}{4}\beta$, see (5.6). From the bound (5.7), it then follows that networks with softplus non-linearities with smaller β value have robuster explanations compared to networks with larger values of β (provided that the Frobenius norms of the weights is the same).

We hence propose to use smoother non-linearities when creating the neural network, i.e. functions with small Σ_1 and Σ_2, so that after the training, these networks' explanations are more robust.

5.1.3. Curvature minimization

As a third approach, we propose to modify the training procedure such that a small value of the Frobenius norm of the Hessian is part of the objective. To this end, we add an additional term to the loss function which penalizes the Frobenius norm, i.e.

$$\mathcal{L} = \mathcal{L}_0 + \zeta \sum_{x \in \mathcal{T}} \|H\|_F^2 (x) \,, \tag{5.11}$$

where $\mathcal{T}$ denotes the training set, $\mathcal{L}_0$ is the unregularized loss function, and ζ is a hyperparameter regulating how strongly the Hessian norm is minimized. A related approach has been previously proposed in [184] in the context of conventional adversarial attacks.

Calculating the Frobenius norm of the Hessian is expensive, i.e. to obtain the exact second derivative we would have to backpropagate through the network once per input pixel. For larger images, this becomes unfeasible, especially when we want to include the norm minimization in the training procedure.

We therefore propose to estimate the Frobenius norm stochastically. Let $v \sim \mathcal{N}(0, 1)$, which implies that $\mathbb{E}[v_i] = 0$ and $\mathbb{E}[v_i v_j] = \delta_{ij}$. We can then rewrite the Frobenius norm of the Hessian as follows

$$\begin{aligned}
||H||_F^2 &= \sum_{i,j} \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)^2 \\
&= \sum_{i,j} \mathbb{E}\left[v_j^2\right] \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)^2 + \underbrace{\sum_{i,j \neq k} \mathbb{E}\left[v_j v_k\right] \frac{\partial^2 f}{\partial x_i \partial x_j} \cdot \frac{\partial^2 f}{\partial x_i \partial x_k}}_{\text{error}} \\
&= \sum_{i,j,k} \mathbb{E}\left[v_j \frac{\partial^2 f}{\partial x_i \partial x_j} \cdot v_k \frac{\partial^2 f}{\partial x_i \partial x_k} \right] \\
&= \mathbb{E}\left[\sum_i \left(\frac{\partial}{\partial x_i} \sum_j v_j \frac{\partial f}{\partial x_j} \right)^2 \right] .
\end{aligned}$$

We can estimate the final expected value using the Monte-Carlo method, i.e. we draw a random vector v, and compute $v^T \nabla f(x)$ at the usual cost of a single backward pass. Since the resulting expression is a scalar, we can calculate its derivative at the cost of another single backward pass [185]. Multiple samples can be combined in mini-batches. The average over the mini-batch is then an unbiased estimator for the expected value.

5.2. Experiments

In this section, we compare the performance of the proposed methods experimentally.

Briefly summarized, we measure the degree of robustness as follows: we perturb an input sample x with Gaussian[2] noise $\delta x \sim \mathcal{N}(0, \sigma^2)$. We then calculate the explanation $h(x_{\text{adv}})$ for the resulting perturbed input $x_{\text{adv}} = x + \delta x$, and measure its similarity to the original explanation $h(x)$. The standard deviation σ is chosen such that the output of the neural network is approximately unchanged, i.e. $f(x) \approx f(x_{\text{adv}})$. We repeat this procedure for various explanation methods.

In more detail, our experiments use the following setup:

Similarity Scores for Explanations In order to quantify the visual similarity of the explanations, we use three different measures following [18]: the Pearson correlation coefficient (PCC), the structural similarity index measure (SSIM) and the mean squared error (MSE), introduced in Section 2.3. We apply these metrics to measure the similarity between the original explanation $h(x)$ and the explanation of the perturbed input $h(x_{\text{adv}})$.

Model and Dataset To demonstrate the proposed robustness effects generically, in our main experiments we use the same convolutional neural network (CNN) architecture for all our models and train on the CIFAR10 data set [146]. The models achieve up to 88% test set accuracy. For more details on the network architecture and training, we refer to Appendix B.4.1.

Noise Level We choose the level of noise such that it does not significantly change the network's output. To this end, we perturb all 10k images of the CIFAR10 test set with Gaussian noise of a given standard deviation σ. It is convenient to express the standard deviation σ in terms of the noise level ν by

$$\sigma = (x_{\max} - x_{\min})\nu\,, \tag{5.12}$$

where $x_{\max}$ and $x_{\min}$ denote the maximum and the minimum values of the input domain.

Figure 5.2 shows the classification accuracy and the PCC similarity score between the original and the perturbed explanations for different noise levels ν. Smaller noise levels (between 0.005 and 0.025) lead to a comparatively mild drop in accuracy but result in a significant reduction in the similarity of the explanations. We therefore restrict the noise levels to this interval for our experiments.

[2]For a discussion of other noise distributions, we refer to Appendix B.4.4.

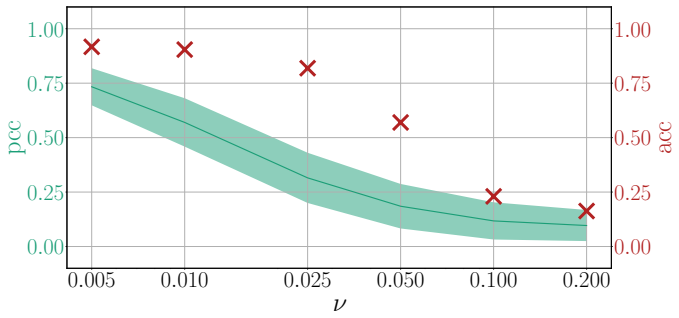


Fig. 5.2.: PCC$(h(x), h(x_{\text{adv}}))$ between explanations drops more rapidly than accuracy when adding noise with small ν to the original image. We show mean +/- std for PCC.

5.2.1. Robustness from weight decay

Weight decay adds a regularizing term to the update rule of the network parameters w_i so that large values are penalized. The update is then given by

$$w_i \to w_i - \alpha\left(\frac{\partial \mathcal{L}_0}{\partial w_i} + \lambda w_i\right) \tag{5.13}$$

where α is the learning rate and $\mathcal{L}_0$ is the unregularized loss. The hyperparameter λ controls how strongly the network parameters are penalized. We choose five different values for λ and train the CNN for each. Figure 5.3 shows higher PCC values for larger values of λ, i.e. weight decay increases the robustness of explanations with respect to input manipulation. As was to be expected, there is a trade-off between robustness and accuracy of the networks. For networks trained with strong weight decay ($\lambda > 10^{-2}$), the accuracy decreases drastically. On the other hand, networks trained with $5 \times 10^{-5} \leq \lambda \leq 5 \times 10^{-3}$ achieve comparable accuracy but are significantly more robust to manipulations than a network trained with $\lambda = 0$.

5.2.2. Robustness from softplus activations

To see how the β value of the softplus activations (5.9) affects the robustness, we train networks with four different β values. We do this for all but the largest value of the weight-decay hyperparameter λ from the previous section; in total $4 \cdot 5 = 20$ networks. With decreasing β values, the explanations become less prone to input manipulations. Figure 5.4 shows the results for networks trained with $\lambda = 5 \times 10^{-4}$ and different values for β. For β values smaller than 5, the accuracy of the network decreases slightly. Crucially, comparable accuracy is achieved for β values of 5 and 10, while the corresponding networks show increased robustness. Results for other choices of the weight decay parameter λ look qualitatively similar. We list results for all combinations in Appendix B.4.1.

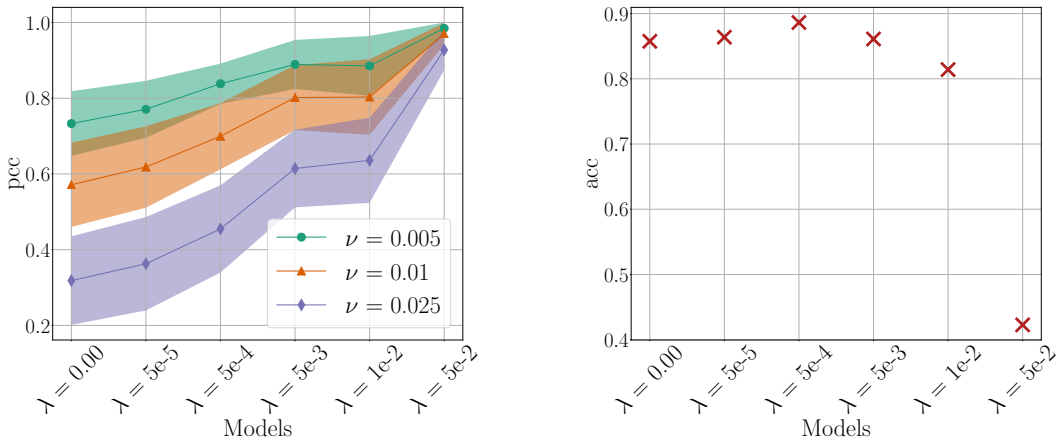


Fig. 5.3.: Left: PCC$(h(x), h(x_{\text{adv}}))$ increases with stronger weight decay (higher λ). Therefore, weight decay improves robustness of explanations. We show mean +/- std for three different noise levels ν. Right: For moderate weight decay ($\lambda \approx 5 \times 10^{-4}$) accuracy increases, while for strong weight decay ($\lambda \geq 10^{-2}$) accuracy drops.

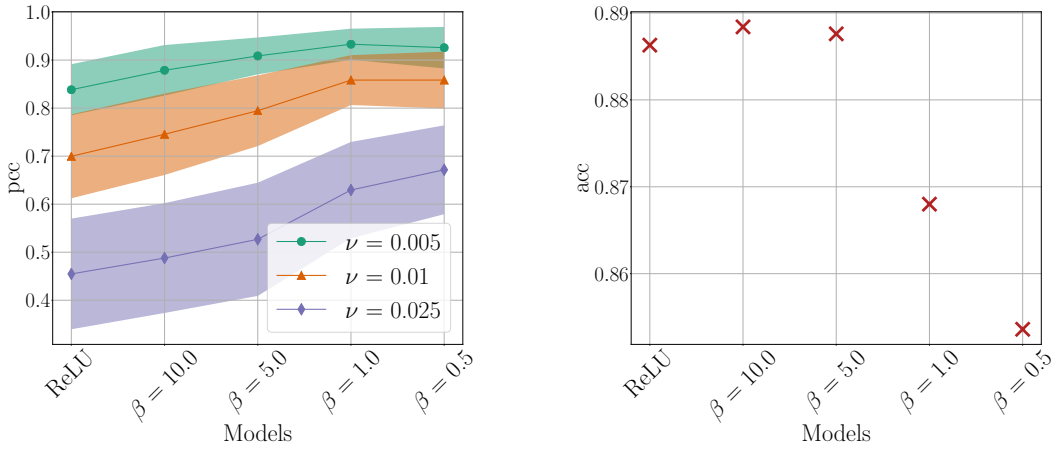


Fig. 5.4.: Left: PCC$(h(x), h(x_{\text{adv}}))$ is higher for networks trained with softplus activation that have small β value. This means, replacing ReLU with softplus activations improves robustness of explanations. We show mean +/- std for three different noise levels ν. Right: Accuracy decreases if β is very small. All networks were trained with weight decay ($\lambda = 5 \times 10^{-4}$).

5.2.3. Robustness from curvature minimization

To evaluate the effectiveness of Hessian norm minimization, we train networks with different values of the hyperparameter ζ which controls the degree of regularization in the modified loss in Eq. (5.11).

We approximate the Hessian norm only for softplus networks since we need to

calculate second derivatives and[3]

$$\frac{\partial^2 f}{\partial x^2} \propto \mathrm{ReLU}'' = 0$$

for ReLU networks. We consider six different values for ζ for each of the networks from the previous section, i.e. we train $6 \cdot 20 = 120$ networks in total.

Figure 5.5 shows how curvature minimization affects the robustness against random perturbations, when using weight decay with $\lambda = 5 \times 10^{-4}$ and softplus activations with $\beta = 10$. Even a small value for ζ results in significant improvement. For larger ζ values, and low noise level, the PCC value slowly converges to (approximately) one but the accuracy of the network decreases. We list results for all combinations of the weight decay parameter λ and the softplus parameter β in Appendix B.4.1.

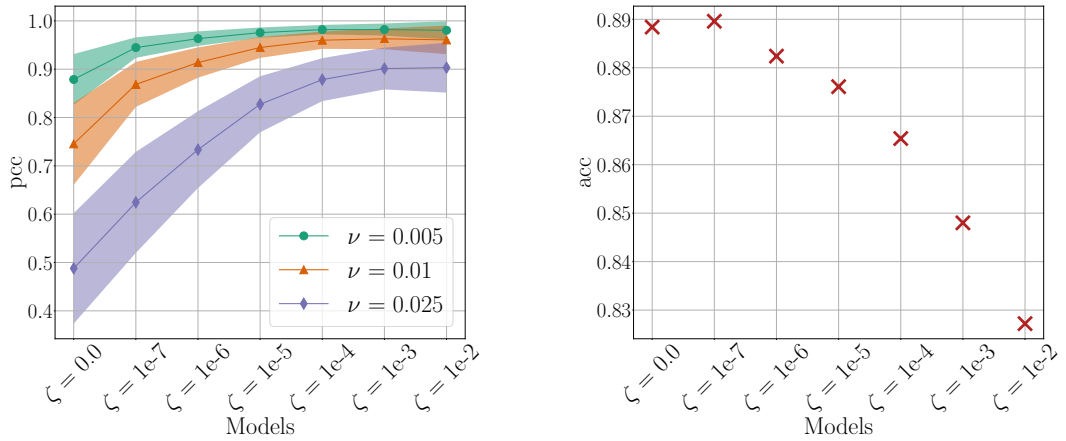


Fig. 5.5.: Left: $\mathrm{PCC}(h(x), h(x_{\mathrm{adv}}))$ is larger for networks trained with strong minimization of the Hessian norm $\|H\|$ (larger ζ values). Therefore, minimizing $\|H\|$ improves robustness of explanations. We show mean +/- std for three different noise levels ν. Right: accuracy decreases when ζ gets large. All networks were trained with weight decay ($\lambda = 5 \times 10^{-4}$) and softplus activations ($\beta = 10$).

To see the effects on the explanation more directly, we show a concrete example[4] in Figure 5.6. In the top row, we show an image and several samples with added Gaussian noise (with noise level $\nu = 0.025$). Below, we show the Gradient explanation maps of two different networks. For the first network (middle row) the explanations appear noisy and vary strongly. This network was trained without any techniques to enhance robustness (no weight decay, ReLU activations, no Hessian minimization). For the second network (bottom row) the explanations stay relatively steady. This network was trained with measures that enhance robustness (weight decay with $\lambda = 5 \times 10^{-4}$, softplus activations with $\beta = 10$, Hessian minimization with $\zeta = 10^{-7}$).

[3]More precisely, the second derivative $\mathrm{ReLU}''(x)$ is not defined for $x = 0$ and the relation only holds up to such root points of the non-linearity.

[4]Another example can be seen in Figure 5.1 c).

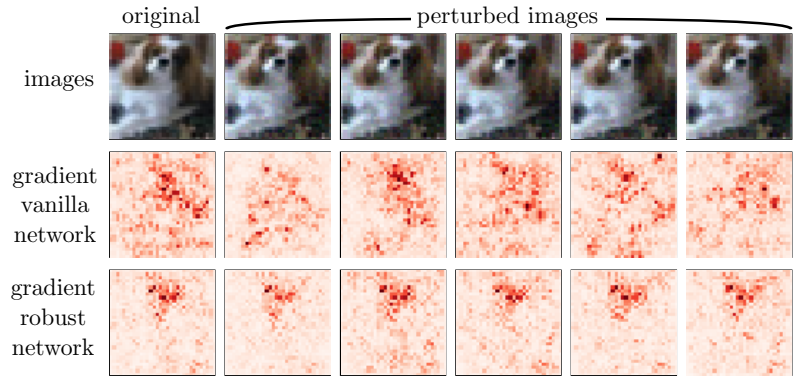


Fig. 5.6.: Top row: original image and image with added noise ($\nu = 0.025$). Middle row: Gradient explanations for a network trained with $\lambda = 0$, ReLU activations and $\zeta = 0$. Bottom row: Gradient explanations for a network trained with weight decay ($\lambda = 5 \times 10^{-4}$), softplus activations ($\beta = 10$) and Hessian minimization ($\zeta = 10^{-7}$). The explanations of the robust network in the bottom row are clearly more resilient to random input perturbations.

5.2.4. Additional experiments

In this section we discuss additional architectures, data sets, and explanation methods as well as robustness to adversarial attacks on the explanation.

Other explanation methods So far we have focused on Gradient explanation maps. However, we can apply any other suitable explanation method to our networks. In Table 5.1, we show results for different explanation methods. Specifically, PCC values (mean and standard deviation over the complete test set) between original and manipulated explanations, when perturbing images with a noise level of $\nu = 0.025$, are listed. In the first row, we show how the respective explanations change when using the original, vanilla network ($\lambda = 0$, ReLU activations, $\zeta = 0$) and in the second row we show the values for a network trained with all our robustness measures ($\lambda = 5 \times 10^{-4}$, $\beta = 10$, $\zeta = 10^{-6}$). While the Gradient explanation is most vulnerable to random perturbations, the results for Gradient×Input and Integrated Gradients look qualitatively similar to the Gradient explanation. When using all our robustifying measures, the PCC similarity between these explanations improves by around 30 to 40 percentage points. Guided Backpropagation (GBP) and Layerwise Relevance Propagation (LRP) are noticeably more resilient to random input perturbations. However, our robust network still achieves significantly higher PCC similarities, demonstrating that even more robust explanation methods can profit from our proposed methods.

Additional architectures and data sets In order to demonstrate that our results also hold for input data with higher dimensions, we consider the ImageNet data

Network	Gradient	Grad×Input	IntGrad	GBP	LRP
vanilla	0.32 ± 0.12	0.44 ± 0.13	0.53 ± 0.12	0.78 ± 0.10	0.91 ± 0.06
robust	0.73 ± 0.08	0.76 ± 0.08	0.82 ± 0.06	0.94 ± 0.03	0.98 ± 0.01

Tab. 5.1.: PCC (mean ± std) between original explanations and explanations of perturbed inputs (noise level $\nu = 0.025$) for explanation maps: Gradient, Gradient×Input, Integrated Gradients, Guided Backpropagation (GBP), and Layerwise Relevance Propagation (LRP). High PCC values indicate high robustness.

set. Due to the substantial computational costs, we restrict our analysis to the ImageNette [148] classes. We train with the original train-test split, and use 50 images per class for testing. We then apply our methods to the network architectures of VGG [159] and ResNet [9] trained on ImageNette. The results are illustrated in Figure 5.7. We observe similar behavior as for the convolutional architecture used on the CIFAR10 data set. Notably, the largest increase in robustness is obtained by substituting ReLU with softplus activations while curvature minimization does not seem to have a comparative effect. For a more detailed discussion, we refer to Appendix B.6.

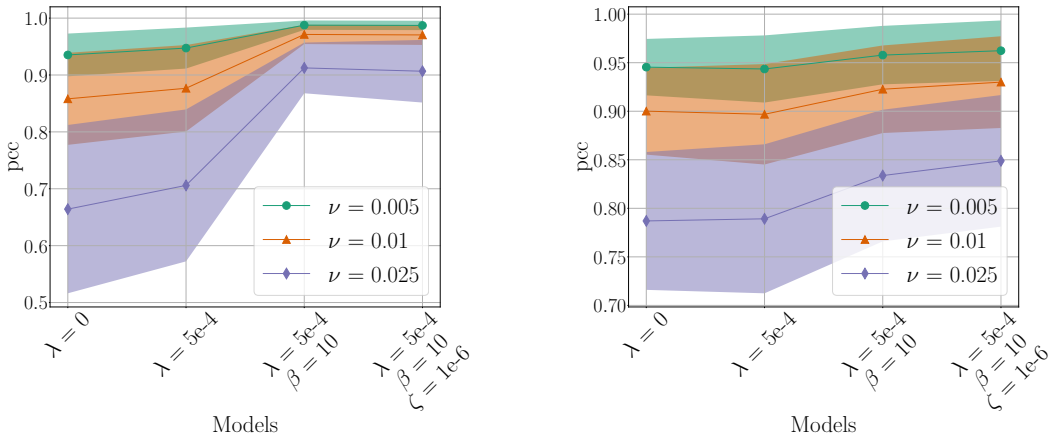


Fig. 5.7.: VGG16 (left) and ResNet (right) trained on (a subset of) ImageNet. $\mathrm{PCC}(h(x), h(x_{\mathrm{adv}}))$ is larger for networks trained with robustness methods.

Adversarial manipulation In addition to perturbing the explanation by using unstructured random noise, we also consider targeted manipulations. We apply the same manipulation method we introduced in Chapter 4. For this, we use the CNN model on CIFAR10 as well as VGG16 and ResNet trained on ImageNette. We choose 100 randomly selected test samples. For each of them, the target explanations is chosen to be the explanation of another randomly selected test sample. As ReLU networks are not twice differentiable, we substitute the ReLU with softplus activations

with large β value during the attack. For the final comparison, we restore the original ReLU activations. We stop an attack when the mean squared error between original and manipulated input exceeds a specified threshold, i.e. we keep the perturbation δx small. As shown in Figure 5.8, our methods for robust networks also significantly increase the robustness against these targeted attacks.

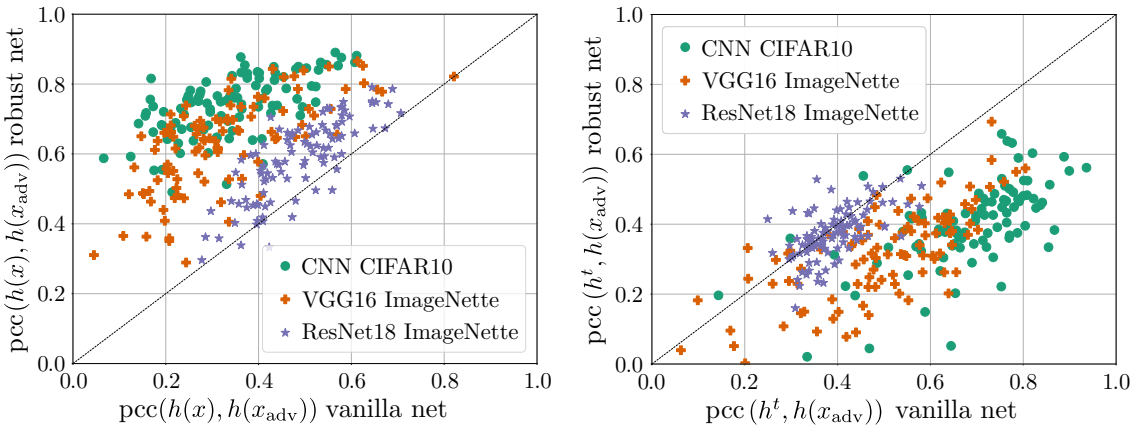


Fig. 5.8.: Targeted adversarial attacks on different models. Left: similarity between manipulated explanation $h(x_{\text{adv}})$ and original explanation $h(x)$ is *higher* for the robust nets trained with our methods. Right: similarity between manipulated explanation $h(x_{\text{adv}})$ and target explanation h^t is *lower* for the robust nets trained with our methods. This shows that our methods also improve robustness against targeted adversarial attacks.

5.2.5. Comparison of proposed methods

All our proposed methods can improve robustness of explanations against input manipulations. We observe this trend for all considered explanation methods, similarity measures, and noise levels.

We note that each method appears to improve robustness in a different manner. As evident from our theoretically-derived upper bound (5.7), both weight decay and small β values for the softplus activations affect the Hessian norm. Weight decay leads to smaller Hessian norms by minimizing the weight norms. Replacing ReLU activations with softplus activations with comparatively small β parameter also leads to smaller Hessian norms but the weight norms stay approximately constant for different β values. When minimizing the Hessian norm directly during training, the Hessian norms decrease significantly while the weight norms decrease only minimally. This shows that Hessian norm minimization does not just improve robustness by indirectly minimizing the weight norms.

Figure 5.9 shows how weight norms and Hessian norms change when applying weight decay (varying λ), substituting ReLU with softplus (varying β) and minimizing the Hessian norm directly (varying ζ). We average over all softplus networks trained

with $\zeta = 0$ for the first two plots and we average over all networks trained with Hessian minimization for the last plot.

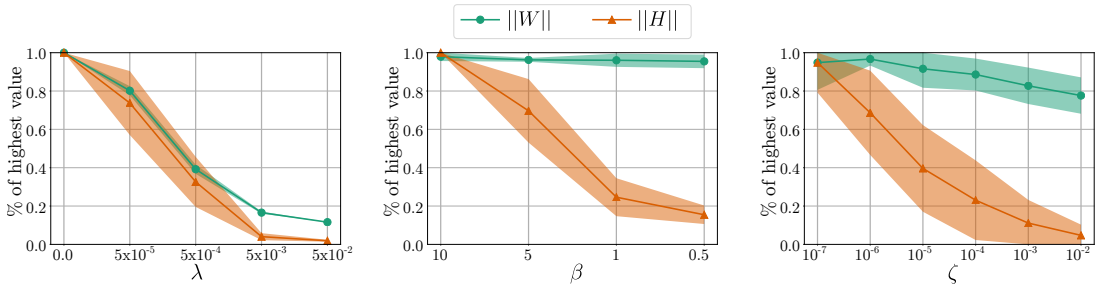


Fig. 5.9.: The connection between weight norms $\|W\|$ and Hessian norms $\|H\|$ is different for our three methods. Left: $\|W\|$ and $\|H\|$ both decrease in a similar manner when applying stronger weight decay (increasing λ). Middle: $\|W\|$ stays relatively constant while $\|H\|$ decreases with increasing softplus β. Right: $\|W\|$ decrease slightly while $\|H\|$ decrease strongly with stronger Hessian minimization (increasing ζ). We show mean $\pm$ std.

While we showed that each method separately improves robustness—we keep the weight-decay hyperparameter λ constant when evaluating different smoothing parameters β for the softplus activations and we keep the weight-decay hyperparameter λ and the smoothing parameter β constant when evaluating different values for hyperparameter ζ for the Hessian norm minimization—we get most benefits when combining them. Besides enhancing robustness, weight decay plays an essential role for the accuracy—as expected and well-known in the literature [181]: all networks trained without weight decay stay at an accuracy below 86.5%.

5.3. Related work

Attention towards (adversarial) manipulation of explanations has developed relatively recently. We give a detailed overview of works that focus on manipulation of explanations in Chapter 4, Section 4.5.

In this section we therefore focus on related works which aim to make explanations more robust against input perturbations. More specifically, we are interested in research related to our work, i.e. approaches that make the neural network itself more robust against manipulation of the explanations by modifying the training process.

Robust training of neural networks has mainly been researched in relation to conventional adversarial attacks that aim to change the neural network's output [162, 186–190].

As explanations and the manipulation thereof is a relatively new field, few works have explored increasing robustness of neural networks during training in the context

of attacks on the explanation.

Wang et al. [191] propose to include a penalty on the largest principal curvature in the loss function to train networks that are more resilient to attacks on the explanation. This is different to our Hessian norm training which can be roughly understood as a penalty on all principal curvatures.

Lakkaraju et al. [192] propose adversarial training (analogous to adversarial training as a defense against conventional adversarial attacks) to construct black box explanations that are robust to input perturbations and distribution shifts.

Patro et al. [193] propose a method to obtain explanations for visual and textual question answering which is robust to input perturbation. More specifically, they develop a collaborative correlated module which simultaneously checks whether the predicted answer and the corresponding explanation are correct or not. In contrast to our work, their application requires the existence of ground truth explanations, as their distance to the model generated explanations is minimized during training.

Zeng et al. [194] propose to train an explainer model that jointly outputs a prediction and a corresponding explanation. They train a (conventional) adversarially robust model. The SmoothGrad explanations of that model are then used as ground truth to train the explainer model, which is applied to a pre-trained network for which the explanations are desired. In contrast to our work, their approach requires training two networks and does not generalize to arbitrary explanation methods.

5.4. Limitations

Adversarial perturbations The majority of our experiments focus on random perturbations of the input rather than adversarial perturbations. This is crucial for a fair analysis of the robustness of the network since networks with different training procedures can be applied to inputs with the same perturbations. Adversarial perturbations on the other hand depend on the specific network. Creating adversarial examples thus depends on network specific hyperparameters. These could for example include optimizer step size and number of iterations. It is thus not clear how a fair comparison can be achieved and how much of the difference in robustness can be attributed to the hyperparameter settings. We hence restricted our analysis of robustness against adversarial attacks to only a few examples. We do observe a trend that confirms our expectations but to reduce the probability of secondary effects due to hyperparameter choices one might have to average over more attacks and training runs.

Hyperparameters Even though we restrict our perturbations to random noise the training process of the networks is still dependent on hyperparameters. It is not entirely clear how to achieve a fair comparison between networks that are trained

with very different regularization methods or different activation functions. For example, do we consider two networks as comparable when they were trained for the same number of epochs or when they reach the same training/validation/test accuracy? This choice could have effects on the robustness of the network that is independent of the approaches to increase robustness that we considered. To rule this out one might have to perform even more training runs.

Other architectures and data sets Our main experiments focus on a relatively simple network and the CIFAR10 data set. Our experiments with more complex architectures and higher dimensional data sets are limited to a few configurations due to the high computational cost. Therefore, our results on VGG16 and ResNet18 should be seen as a proof of concept and a trend observation rather than a careful analysis like we performed with the simpler CNN architecture applied to the CIFAR10 data set.

Rescaling the network parameters One could argue that rescaling all network parameters, i.e. multiplying all weights and biases with a value $c \leq 1$, would result in smaller weight norms and therefore a smaller Hessian norm. While this is true, it does not affect the properties of the gradient which we are interested in. As we normalize the explanation maps, only relative changes in the gradient, that is, only changes to the direction of the gradient, matter. A rescaling of all network parameters would affect only the gradient magnitude and not the gradient direction. While we mention this in Section 4.3, we do not emphasize it in the current chapter. To include this in the theoretical analysis one would have to divide Equation (5.7) by the gradient magnitude or a lower bound thereof. This limitation might also partially explain why we do not see significant improvement of the robustness for VGG16 and ResNet18 when using weight decay (see Figure 5.7), although the weight norms significantly decrease (see Figure B.16).

5.5. Summary

In this chapter, we have addressed the need for the robustness of explanation methods against the manipulation of input data. Rather than introducing a new explanation method, we focused on enhancing the robustness of the networks themselves. As a result, any applied explanation method was shown to profit.

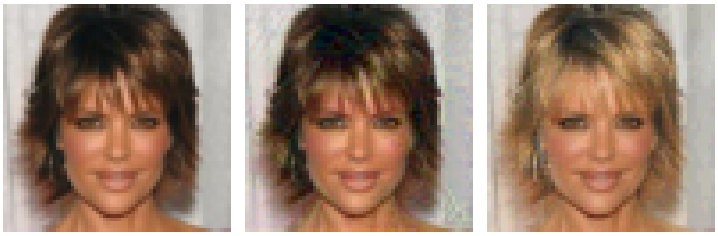
Based on the derived bounds for the maximal change in explanation, we proposed three approaches to increase the robustness of explanations. Specifically, we showed that weight decay, known to improve accuracy, can efficiently boost the robustness of explanations. We furthermore propose to use networks with smoothed activation functions and to include a regularizer for the network's curvature in the training process, which leads to significantly enhanced resilience against manipulated inputs.

Our extensive experiments with a CNN architecture and the CIFAR10 data set, as well as further experiments with ImageNet data and the VGG16 and ResNet18 architectures, demonstrated the validity and usefulness of our proposed methods. We consider random input perturbations with different noise levels as well as targeted adversarial attacks on the explanation and confirm that our methods improve robustness in both settings.

6. Counterfactual explanations with generative models

While the previous two chapters treated attribution methods, in this chapter we focus on counterfactual explanations.

We introduce a simple—yet effective—algorithm to construct counterfactual explanations for neural network classifiers on high-dimensional input data such as images. More specifically, we perform a suitable diffeomorphic coordinate transformation and then apply gradient ascent in these coordinates to find counterfactuals which are classified with great confidence as a specified target class. Leveraging different kinds of generative models, we propose two methods to construct such suitable coordinate systems that are either exactly or approximately diffeomorphic. We analyse the generation process theoretically using Riemannian differential geometry and validate the quality of the generated counterfactuals using various qualitative and quantitative measures.



original
not blond

adversarial
blond ($p \approx 0.99$)

counterfactual
blond ($p \approx 0.99$)

Fig. 6.1.: Example from the CelebA data set. The original is classified as not blond. The adversarial example and the counterfactual are classified as blond with high confidence. The counterfactual shows semantic differences to the original in the hair region. This stands in stark contrast to the adversarial example, for which we only note hardly perceptible, seemingly unstructured noise.

Figure 4.1 shows an image, a conventional adversarial example and an example of a counterfactual which was generated with our method. The binary classifier in this example differentiates between 'blond' and 'not blond'. The original image is classified as 'not blond'. When we perform gradient ascent in image space, we usually get an adversarial example (shown in the middle). The difference between the original

and the adversarial example is almost imperceptible but the latter is classified as the target class 'blond' with high confidence. Performing the gradient ascent updates in the latent space of a generative model, we get a counterfactual (shown on the right). In contrast to the adversarial example, the counterfactual shows semantic changes in the hair region of the person, which suggests that the classifier has correctly identified this region as most relevant for hair color classification. Crucially, the other facial features remain unchanged.

6.1. Generative Models

In this section we give a short introduction to the generative models we use for finding counterfactual images, namely variational autoencoders, generative adversarial networks, and normalizing flows (see Figure 6.2). The idea for all these generative models is to approximate the data distribution $p(x)$ with a distribution $q(x)$ which is based on a simple base distribution $p_z(z)$ (for example a multivariate Gaussian). The modeled distribution is an approximation of the data manifold, which we use in our method to produce counterfactuals that lie on the data manifold.

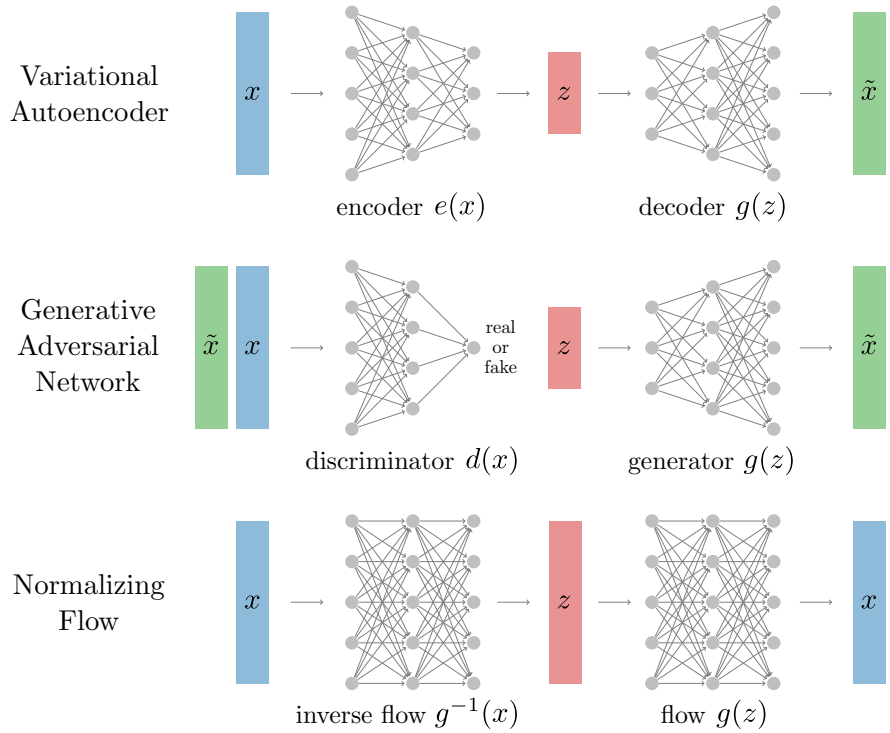


Fig. 6.2.: Schematic illustration of different structures for generative models.

6.1.1. Variational autoencoders

A variational autoencoder (VAE) [195] has an architecture similar to an autoencoder, including an encoder e that is usually only used during the training process and a decoder g that is used to generate new samples after training.

The loss is based on the *evidence lower bound*

$$\text{ELBO} = \log p(x) - D_{\text{KL}}[q_z(z|x)||p_z(z|x)]\,. \tag{6.1}$$

We want to maximize the likelihood of the data $p(x)$. The second term in 6.1 is the (reverse) Kullback–Leibler divergence

$$D_{\text{KL}}(Q||P) = \sum_{x\in\mathcal{X}} Q(x)(\log Q(x) - \log P(x))\,, \tag{6.2}$$

which can be seen as a regularizer that ensures that the encoder learns a distribution $q_z(z|x)$ similar to the true posterior $p_z(z|x)$. As the KL divergence is always positive or zero, maximizing the ELBO maximizes $p(x)$. The VAE loss is then defined as the negative ELBO and can also be written as

$$L_{\text{VAE}} = D_{\text{KL}}(q_z(z|x)||p_z(z)) - \mathbb{E}_{z\sim q(z|x)}[p(x|z)]\,. \tag{6.3}$$

When minimizing L_{VAE} the KL divergence term in 6.3 regularizes the parameters of the encoder so that the approximate posterior $q_z(z|x)$ is close to the prior $p_z(z)$, usually a standard Gaussian $p_z(z) = \mathcal{N}(z; 0, 1)$. The output of the encoder is therefore divided into mean $\mu_e = \mu(e(x))$ and standard deviation $\sigma_e = \sigma(e(x))$ parametrizing a Gaussian distribution $q_z(z|x) = \mathcal{N}(z; \mu_e, \sigma_e)$.

The second term ensures that the original image is likely under the learned probability distribution $p(x|z)$. This probability distribution is approximated by the decoder g, that produces the parameters $\tilde{x} = g(z)$ for the distribution. The latent variable z is sampled from the Gaussian $z \sim \mathcal{N}(\mu_e, \sigma_e)$. To enable computation of gradients this sampling is re-parametrized to $z = \mu_e + \sigma_e \odot \epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$. The probability density $p(x|z)$ is assumed to be Gaussian for real valued data and the generator serves to parametrize μ, while σ is assumed to be 1. We can then calculate

$$-\log p(x|z) = -\log \mathcal{N}(x; \tilde{x}, 1) = \log\sqrt{(2\pi)^n} + \frac{1}{2}\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2\,, \tag{6.4}$$

where $\tilde{x} = g(z)$ and n is the number of dimensions of $x \in \mathbb{R}^n$. Minimizing this term is equivalent to minimizing the squared Euclidean distance between the original image x and the reconstructed image $\tilde{x}$.

Assuming Gaussian distributions for the approximate posterior $q_z(z|x)$ and the probability density $p(x|z)$ the actual loss that is then used for training simplifies to

$$L_{\text{VAE}} = \mathbb{E}_{x\sim p(x)}[\log \mathcal{N}(z; \mu_e, \sigma_e) - \log \mathcal{N}(z; 0, 1) + ||g(e(x)) - x||^2]\,, \tag{6.5}$$

where z depends on the data sample x.

Improvements on the loss function and network architectures of decoder and encoder [196–199] have made high-quality image generation with VAEs possible. In Section 6.4.2 we apply our method to a relatively simple convolutional VAE architecture.

6.1.2. Generative adversarial networks

A generative adversarial network (GAN) [200] consists of two models: a generator $g : \mathcal{Z} \to \mathcal{X}$ that is trained to map samples from a Gaussian distribution $z \sim \mathcal{N}(\mu, \sigma)$ to the desired target distribution and a discriminator $d : \mathcal{X} \to [0, 1]$ that is trained to distinguish real data samples from generated ones.

During the training process, the two models compete against each other and consequently improve their abilities. The loss function which we optimize is then given by

$$\min_g \max_d L_{\text{GAN}} = \mathbb{E}_{x\sim p(x)}[\log d(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - d(g(z)))] \tag{6.6}$$

where $p(x)$ is the (real) data distribution and $p(z)$ is the distribution over latent variables z, which is usually chosen to be Gaussian or uniform.

A global optimum is reached when the real data distribution is closely approximated by the generator and the discriminator therefore cannot distinguish between generated and real images.

To use the generative model one does not require the discriminator d but only the generator g.

GANs have made a lot of progress in their capabilities in recent years [3, 201–203]. In Section 6.4.2 we apply our method to three different GAN architectures.

6.1.3. Normalizing flows

While for VAEs and GANs the latent variable z is usually of lower dimension than the data x, for normalizing flows [204] they have the same dimension. The flow $g : \mathcal{Z} \to \mathcal{X}$ is then defined as a bijective function that maps a sample from the latent distribution $z \sim q_z$ to a sample $x = g(z)$ from the approximated data distribution $q(x)$. This makes the flow invertible so that $z = g^{-1}(x)$.

By using the change of variables theorem one can express q in terms of q_z and the normalizing flow g:

$$q(x) = q_z(z) \left| \det \frac{\partial z}{\partial x} \right| = q_z(g^{-1}(x)) \left| \det \frac{\partial g^{-1}(x)}{\partial x} \right| , \tag{6.7}$$

where $\frac{\partial g^{-1}(x)}{\partial x}$ is the Jacobian of g^{-1}. The probability q_z is usually chosen to match the standard Gaussian $p_z = \mathcal{N}(0, 1)$. The flow can be trained by maximum likelihood, i.e. by minimizing

$$\begin{aligned} L_{\text{flow}} = \text{KL}(p|q) &= -\mathbb{E}_{x\sim p} \log q(x) + \text{const.} \\ &\approx -\frac{1}{N} \sum_{i=1}^{N} \log(q(x_i)) + \text{const.}\,, \end{aligned} \tag{6.8}$$

where $x_i \sim p$ are samples from the data density p. The flow g is implemented by a neural network with a specific structure to ensure invertibility.

Normalizing flows have very recently become powerful image generators [205–208]. Our main theoretical results from Section 6.3.2 are based on normalizing flows and we apply our method to two different flow architectures in Section 6.4.1.

6.2. Methods

In this section, we introduce diffeomorphic and approximately diffeomorphic counterfactuals. We will start by briefly reviewing the basics of counterfactual explanations, and then present the two methods. Figure 6.3 provides an intuition of how our method works and showcases the difference between counterfactuals found using our method and adversarial examples found when doing gradient ascent in data space.

6.2.1. Counterfactual explanations

We start with a classifier $f : \mathcal{X} \to \mathbb{R}^C$ which assigns a probability $f(x)_c$ for class $c \in \{1, \dots, C\}$ to an input $x \in \mathcal{X}$. A counterfactual explanation of this classifier is an alternative input $x' = x + \delta x$ with deformations δx such that the prediction of the classifier is changed.

Usually the data lies approximately on a submanifold $\mathcal{D} \subset \mathcal{X}$ which is of significantly lower dimension $N_{\mathcal{D}}$ than the dimension $N_{\mathcal{X}}$ of the input space $\mathcal{X}$. This is commonly known as the manifold hypothesis (see e.g. [42]). For counterfactual explanations, as opposed to adversarial examples, we are interested in inputs x' which lie on the data manifold. Additionally, we require the deformations to the original data to be minimal as they should only target features that are relevant for the classification, i.e. the perturbation δx should be as small as possible. The relevant norm should however be measured along the data manifold and should not be calculated in the input space. For example, a slightly shifted number in an MNIST image may have large Euclidean distance between pixel values but should be considered an infinitesimal perturbation of the original image.

We formalize the manifold hypothesis mathematically by assuming that the data is concentrated in a small region of extension δ around $\mathcal{D}$. As we will show in Section 6.3,

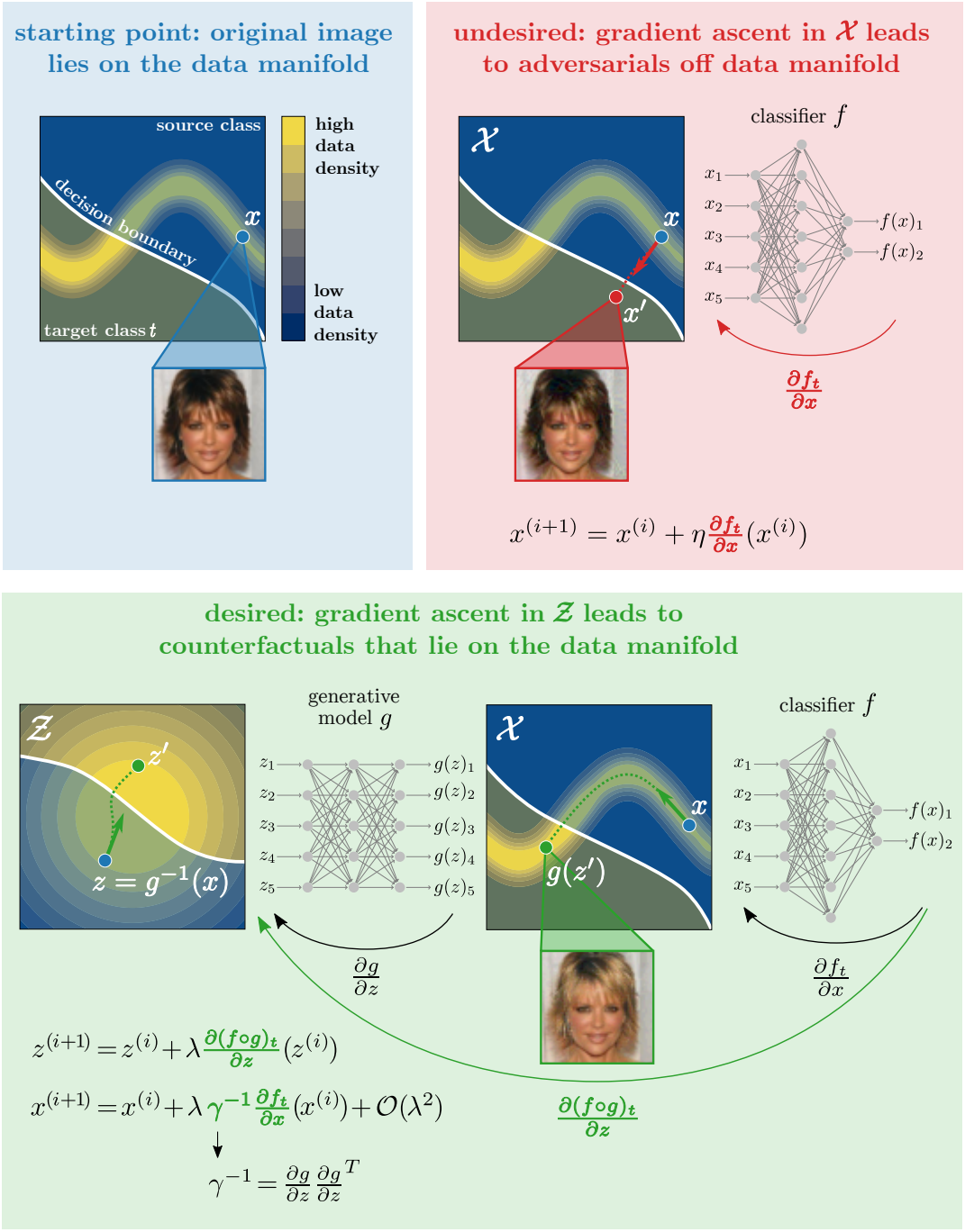


Fig. 6.3.: Intuition for our method. **Blue box:** Image data usually lies on a lower dimensional manifold embedded in high-dimensional space. **Red box:** Gradient ascent in $\mathcal{X}$ leads to adversarial examples. The changes to the original image resemble unstructured noise. **Green box:** Gradient ascent in $\mathcal{Z}$ leads to counterfactuals. The changes to the original image are semantic. This is achieved by scaling the directions of the gradient leading off manifold to effectively zero, using the inverse metric γ^{-1}.

this implies that the support S of the data density p is a product manifold

$$S = \mathcal{D} \times \mathcal{I}_{\delta_1} \times \cdots \times \mathcal{I}_{\delta_{N_\mathcal{X} - N_\mathcal{D}}}\,, \tag{6.9}$$

where $\mathcal{I}_\delta = (-\frac{\delta}{2}, \frac{\delta}{2})$ is an open interval of length δ (with respect to the Euclidean distance in the input space $\mathcal{X}$). We assume that δ is small, i.e. the data lies approximately on the low-dimensional manifold $\mathcal{D}$ and thus fulfills the manifold hypothesis. We can think of the $\mathcal{I}_\delta$ as arising from the inherent noise in the data.

Furthermore, we define the set of points in S classified with confidence $\Lambda \in (0, 1)$ as class $t \in \{1, \ldots, C\}$ by

$$S_{t,\Lambda} = \{x \in S \mid t = \text{argmax}_j f_j(x) \text{ and } f_t(x) > \Lambda\}\,. \tag{6.10}$$

A *counterfactual* $x' \in \mathcal{X}$ for class t of the original sample $x \in \mathcal{X}$ then is the closest point to x in $S_{t,\Lambda}$,

$$x' \in S_{t,\Lambda} \quad \text{and} \quad \text{argmin}_y d_\gamma(x, y) = x'\,, \tag{6.11}$$

where $d_\gamma(x', x)$ is the distance computed by the Riemannian metric γ on S (which is induced from the flat metric by the diffeomorphism given by the generative model). We can think of this as the geodesic distance on the data manifold. We refer to Chapter 3 for an introduction to the necessary concepts of Riemannian geometry.

6.2.2. Generation of counterfactuals

Counterfactuals are often generated by performing gradient ascent in the input space $\mathcal{X}$. More precisely, for step size η and target class t, one adapts the original input, using the update

$$x^{(i+1)} = x^{(i)} + \eta \frac{\partial f_t}{\partial x}(x^{(i)})\,, \tag{6.12}$$

until the classifier has reached a desired confidence Λ for target class t, i.e. $f(x^{(i+1)})_t > \Lambda$. However, the result will generally not lie on the data manifold and differs from the original input x only in unstructured noise which, for high-dimensional data, is usually imperceptible to humans. Inputs that were perturbed in this way are broadly referred to as adversarial examples and not counterfactuals. For a valid counterfactual we desire an alternative input x' that differs from the original in an interpretable, semantically meaningful manner.

Therefore, we propose to estimate the counterfactual x' of the original data point x by using a diffeomorphism $g : \mathcal{Z} \to S$. We then perform gradient ascent in the latent space $\mathcal{Z}$, i.e.

$$z^{(i+1)} = z^{(i)} + \lambda \frac{\partial (f \circ g)_t}{\partial z}(z^{(i)}) \tag{6.13}$$

with step size $\lambda \in \mathbb{R}_+$. This has the important advantage that the resulting counterfactual will lie on the data manifold. Furthermore, since we consider a diffeomorphism g, and thus in particular a bijective map, no information will be lost by considering the classifier $f \circ g$ on $\mathcal{Z}$ instead of the original classifier f on the data manifold S, i.e. there exists a unique $z = g^{-1}(x) \in \mathcal{Z}$ for any $x \in S$. Algorithm 1 shows pseudo code for our approach.

Algorithm 1 Generating counterfactuals

Require: $x, f, g, g^{-1}, t, \Lambda, \lambda, N$
$z \leftarrow g^{-1}(x)$
for i in range(N) **do**
  $\nabla_z \leftarrow \frac{\partial (f \circ g)_t}{\partial z}$
  $z \leftarrow$ optimizer.step(λ, ∇_z)
  if $f(g(z))_t > \Lambda$ **then**
    return $g(z)$
  end if
end for
return None

Note: x is the input for which we desire to find a counterfactual explanation, f the predictive model, g the generative model, g^{-1} the (approximate) inverse of g, t the target class, Λ the target confidence, λ the learning rate and N the maximum number of update steps.

The schematic illustration in Figure 6.4 shows, how gradient ascent in $\mathcal{X}$ and $\mathcal{Z}$ are well-suited to generate adversarial examples and counterfactuals, as the (scaled) gradient direction points off manifold and along the manifold respectively.

For regression tasks there is no explicit decision boundary, but we can still follow the the same algorithm by directly maximizing (or minimizing) the output r of regressor $f(x)$ until we reach the desired target regression value.

6.2.3. Diffeomorphic counterfactuals

We propose to model the map g using a normalizing flow and will refer to the corresponding modified inputs x' as *diffeomorphic counterfactuals* in the following.

Since the flow is bijective on the entire input space $\mathcal{X}$, it will, in particular, be bijective on the data manifold $S \subset \mathcal{X}$. Furthermore, a well-trained flow maps (to very good approximation) only to the data manifold, i.e. $g(\mathcal{Z}) \approx S$, which we thoroughly show in Section 6.3.2.

Normalizing flows thus guarantee that no information is lost when performing gradient ascent in the latent space $\mathcal{Z}$ and also ensure that the resulting counterfactuals lie

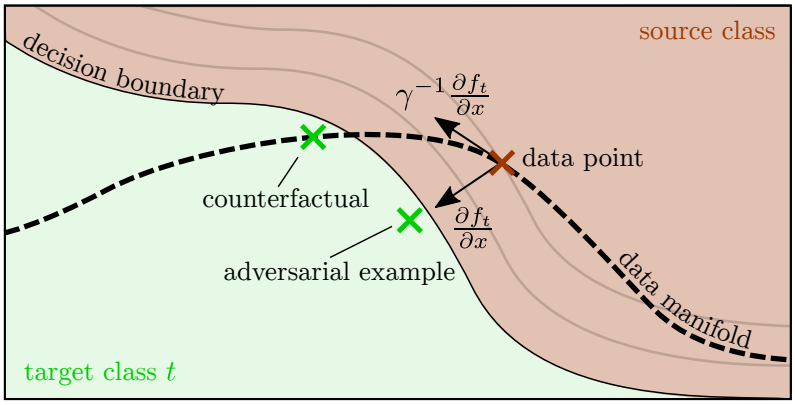


Fig. 6.4.: When the gradient ascent optimization of the target class is performed in the input space of the classifier, one leaves the data manifold and obtains an adversarial example. If instead the gradient ascent is performed in the latent space of a generative model, one stays on the data manifold, resulting in a counterfactual example.

on the data manifold S. Indeed, the flow can be understood as inducing a certain coordinate change of the input space $\mathcal{X}$ which is particularly suited for the generation of counterfactuals.

6.2.4. Approximately diffeomorphic counterfactuals

Using normalizing flows to model the data manifold seems very appealing as these models come with strong theoretical guarantees. However, it is challenging to scale them to very high-dimensional data. This is due to the fact that flows have a very large memory footprint since each layer has the same dimension as the data space $\mathcal{X}$ to ensure bijectivity, leading to high hardware requirements and long training times. We therefore propose an alternative method, which we term *approximately diffeomorphic counterfactuals*, which comes with less rigorous theoretical guarantees, but can scale better to very high-dimensional data. Specifically, we propose two kinds of approximately diffeomorphic counterfactuals:

VAE-based counterfactuals: The reconstruction loss, i.e. $\mathbb{E}_{x\sim p}||g(e(x)) - x||^2$, of a VAE, with encoder $e : \mathcal{X} \to \mathcal{Z}$ and generator $g : \mathcal{Z} \to \mathcal{X}$ is minimal if the encoder is the inverse of the generator on the data manifold S, i.e.

$$e|_S = g^{-1}|_S \,. \tag{6.14}$$

This implies, in particular, that $g(\mathcal{Z}) = S$ if $\dim(\mathcal{Z}) = \dim(S)$. Equivalent to normalizing flows, the image of the autoencoder is the data manifold if the model has been perfectly trained. However, an autoencoder will only be invertible on the data manifold in this perfect training limit and if the latent space $\mathcal{Z}$ has the same dimension as the data space S. This is in contrast to normalizing flows which are invertible on all of $\mathcal{X}$ by construction. As a result, autoencoders will necessarily lead

to loss of information unless the model is perfectly trained and the dimension of the latent space perfectly matches the dimension of the data.

GAN-based counterfactuals: It can be shown (see Section 4.1 of [200] for a proof) that the global minimizer of the GAN loss function (6.6) ensures that samples of the optimal generator g are distributed according to the data distribution, i.e.

$$g(z) \sim p \quad \text{for} \quad z \sim q_Z \,. \tag{6.15}$$

However, the optimal generator g is not necessarily bijective on the data manifold. This means that there may not exist a unique $z \in \mathcal{Z}$ for a given data sample $x \in \mathcal{X}$ such that $x = g(z)$, even if the GAN is perfectly trained. Furthermore, GANs have no inherent mechanism to obtain the corresponding latent sample $z \in \mathcal{Z}$ for a given input $x \in \mathcal{X}$. In contrast to that, the inverse map $g^{-1} : \mathcal{X} \to \mathcal{Z}$ is explicitly known for normalizing flows and approximately known for VAEs.

However, there is extensive literature for GAN inversion, see [209] for a recent review. Approaches to GAN inversion may propose training an encoder network by minimizing a reconstruction loss, similar to the autoencoder pipeline. Furthermore, there are optimization-based approaches which update the latent representation $z \in \mathcal{Z}$ for a specific data sample $x \in \mathcal{X}$, so that $x \approx g(z)$. While a naive approach would be to just directly minimize the Euclidean distance $||g(z) - x||$, avoiding local minima can often be accomplished to some extend by minimizing the difference between the activations $a^{(l)}$ of an intermediate layer l of some auxiliary network, i.e.

$$z = \text{argmin}_{\hat{z} \in \mathcal{Z}} ||a^{(l)}(g(\hat{z})) - a^{(l)}(x)|| \,. \tag{6.16}$$

For example, $a^{(l)}$ can be chosen to be an intermediate layer of an Inception network [10], trained on samples from the data distribution p. Hybrid methods exploit the advantages of both approaches, as an initial latent representation can be found quickly using the trained encoder network and an optimization based approach can then further improve upon the initially chosen z. Of course, there is no guarantee for perfect inversion since the optimization objective is generally non-convex.

6.3. Theoretical analysis

In this section, we use tools from differential geometry to show that for well-trained generative models, the gradient ascent update (6.13) in the latent space $\mathcal{Z}$ does indeed stay on the data manifold. Intuitively, since in (6.13) we take small steps in $\mathcal{Z}$, where the probability distribution is, for example, a normal with unit variance, we do not leave the region of high probability in the latent space and hence stay in a region of high probability also in $\mathcal{X}$.

We prove this statement for the case of diffeomorphic counterfactuals, i.e. for normalizing flows, and (under stronger assumptions) also for approximately diffeomorphic counterfactuals, i.e. for VAEs and GANs.

6.3.1. Mathematical setup

In this section we define the necessary manifolds and coordinates, in order to analyse the gradient ascent (6.13) in the latent space $\mathcal{Z}$.

As above, let $\mathcal{X}$ be an $N_{\mathcal{X}}$-dimensional manifold which is the input space of the classifier $f : \mathcal{X} \to \mathbb{R}^C$ with C classes. An implementation of the classifier corresponds to a function on $\mathbb{R}^{N_{\mathcal{X}}}$ and we denote the coordinates on $\mathcal{X}$ in which our classifier is given by x^α. These coordinates could e.g. be suitably normalized pixel values. We furthermore use an $N_{\mathcal{Z}}$-dimensional manifold $\mathcal{Z}$ as the latent space for our generative model $g : \mathcal{Z} \to \mathcal{X}$. For GANs and AEs, we typically have $N_{\mathcal{Z}} < N_{\mathcal{X}}$ and for normalizing flows $N_{\mathcal{Z}} = N_{\mathcal{X}}$. In the latter case we have moreover $\mathcal{X} = \mathcal{Z}$ and g bijective with differentiable inverse implying that g is a diffeomorphism. Similarly to the classifier, also the generative model is implemented in specific coordinates on $\mathcal{Z}$ which we denote by z^a.

We equip $\mathcal{Z}$ with a flat Euclidean metric δ_{ab}. Then, the generative model g induces an inverse metric $\gamma^{\alpha\beta}$ on $g(\mathcal{Z})$ by

$$\gamma^{\alpha\beta} = \delta^{ab} \frac{\partial g^\alpha}{\partial z^a} \frac{\partial g^\beta}{\partial z^b} \,. \tag{6.17}$$

in the case of $N_{\mathcal{Z}} < N_{\mathcal{X}}$, γ is singular. This metric is the crucial new ingredient when performing the gradient ascent update in the latent space (6.13) as opposed to in the input space (6.12), as the following calculation shows.

One step of gradient ascent in the latent space $\mathcal{Z}$ is given by the image under g of the update step (6.13). In x^α coordinates and to linear order in the learning rate λ, it is given by

$$\begin{aligned} g^\alpha(z^{(i+1)}) &= g^\alpha(z^{(i)}) + \lambda \frac{\partial g^\alpha}{\partial z^a} \frac{\partial (f \circ g)_t}{\partial z^a} + \mathcal{O}(\lambda^2) \\ &= g^\alpha(z^{(i)}) + \lambda \frac{\partial g^\alpha}{\partial z^a} \frac{\partial g^\beta}{\partial z^a} \frac{\partial f_t}{\partial x^\beta} + \mathcal{O}(\lambda^2) \\ &= g^\alpha(z^{(i)}) + \lambda \, \gamma^{\alpha\beta} \frac{\partial f_t}{\partial x^\beta} + \mathcal{O}(\lambda^2) \,. \end{aligned} \tag{6.18}$$

If we start from the same points, $x^{(i)} = g(z^{(i)})$, the difference between gradient ascent in latent space (6.13) and input space (6.12) is just given by the contraction of the gradient of f with respect to x with the inverse induced metric $\gamma^{\alpha\beta} = \frac{\partial g^\alpha}{\partial z^a} \frac{\partial g^\beta}{\partial z^a}$. Hence, in order to understand why the prescription (6.13) stays on the data manifold, we will in the following investigate the properties of γ for the case of well-trained generative models.

Before returning to γ, we will first discuss the structure of the data. The probability density of the data on $\mathcal{X}$ is denoted by $p : \mathcal{X} \to \mathbb{R}$ and the probability density induced by g is denoted by $q : \mathcal{X} \to \mathbb{R}$. For q in x^α coordinates, we use the notation $q_x : \mathbb{R}^{N_{\mathcal{X}}} \to \mathbb{R}$. The data is characterized by $S = \mathrm{supp}(p) \subset \mathcal{X}$ which becomes

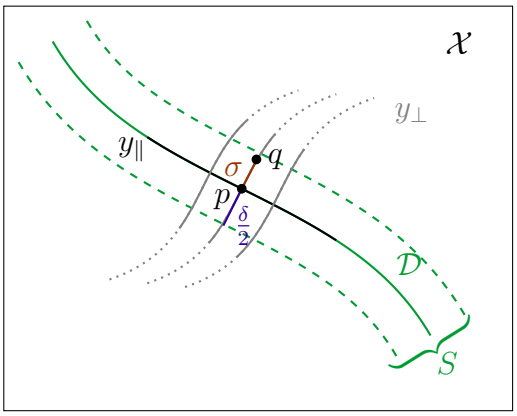


Fig. 6.5.: Construction of the y^μ coordinates which are aligned with the data manifold $\mathcal{D}$.

$S_x \subset \mathbb{R}^{N_\mathcal{X}}$ in x^α coordinates. We will assume that the data lives approximately on a submanifold $\mathcal{D} \subset S$ of $\mathcal{X}$ with dimension $N_\mathcal{D} \ll N_\mathcal{X}$. In relation to the dimension of our generative model, we assume that $N_\mathcal{D} \leq N_\mathcal{Z} \leq N_\mathcal{X}$. As a subset of $\mathcal{X}$ and in x^α coordinates, $\mathcal{D}$ will be denoted by $\mathcal{D}_x \subset \mathbb{R}^{N_\mathcal{X}}$. To capture that the data does not extend far beyond $\mathcal{D}$, we assume that S has Euclidean extension $\delta \ll 1$, normal to $\mathcal{D}$ in x^α coordinates, i.e.[1]

$$S_x = \left\{ x_\mathcal{D} + x_\delta \;\middle|\; x_\mathcal{D} \in \mathcal{D}_x,\ x^\alpha_\delta \in \left(-\frac{\delta}{2}, \frac{\delta}{2} \right) \right\} . \tag{6.19}$$

Next, we will define coordinates in a neighborhood of $\mathcal{D}$ which separate the directions tangential and normal to $\mathcal{D}$ as illustrated in Figure 6.5. Our construction is similar to the constructions of Riemannian and Gaussian normal coordinates, adapted for a submanifold of codimension larger than one. First, we choose coordinates $y_\parallel$ on $\mathcal{D}$ and, for each $p \in \mathcal{D}$, a basis $\{n_i\}$ for the tangent space $T_p\mathcal{D}_\perp$ of the normal to $\mathcal{D}$ at p. Following the usual construction of Riemannian normal coordinates, we assign coordinates to a point q in some neighborhood of $p \in \mathcal{D}$ by constructing an affinely parametrized geodesic $\sigma : [0,1] \to \mathcal{X}$ which satisfies $\sigma(0) = p$ and $\sigma(1) = q$ and which has tangent vector $\sigma'(0) \in T_p\mathcal{D}_\perp$. The coordinates of q are then $y(q) = (y_\parallel(p), y_\perp) \in \mathbb{R}^{N_\mathcal{D}} \oplus \mathbb{R}^{N_\mathcal{X} - N_\mathcal{D}}$, where the i^{th} component of $y_\perp$ is given by the i^{th} component of $\sigma'(0)$ in the basis $\{n_i\}$. In a sufficiently small neighborhood around $\mathcal{D}$, we can find a unique basepoint $p \in \mathcal{D}$ and geodesic σ for every q.

One important aspect of this construction is that by rescaling the basis vectors $\{n_i\}$, we can rescale the components of $\sigma'(0)$.[2] This means we can rescale the $y_\perp$

[1] The form (6.19) restricts the slices $S_\perp(x_\mathcal{D})$ through S normal to $\mathcal{D}$ to be L_1 balls whose size is independent of $x_\mathcal{D}$. We make this restriction to simplify notation but the argument can straightforwardly be extended to arbitrary shapes of $S_\perp(x_\mathcal{D})$ by bounding it by an L_2 ball of radius $\delta/2$.

[2] Note that this does not change the parametrization of the geodesic, hence we still have $\sigma(0) = p$ and $\sigma(1) = q$.

coordinates arbitrarily and hence we can use this freedom to bound S in y coordinates by the same δ that appeared in (6.19),

$$S_y = \left\{ (y_\parallel, y_\perp) \in \mathbb{R}^{N_\mathcal{X}} \;\middle|\; y_\parallel \in \mathcal{D}_y,\ y^i_\perp \in \Big(-\frac{\delta}{2}, \frac{\delta}{2} \Big) \right\} . \tag{6.20}$$

Furthermore, in $g(\mathcal{Z})$, we can choose the basis $\{n_i\}$ orthogonal with respect to the (singular) metric γ and obtain in some neighborhood of $\mathcal{D} \cap g(\mathcal{Z})$

$$\gamma^{\mu\nu}(y) = \begin{pmatrix} \gamma^{-1}_\mathcal{D}(y) & & & \\ & \gamma^{-1}_{\perp_1} & & \\ & & \ddots & \\ & & & \gamma^{-1}_{\perp_{N_\mathcal{X} - N_\mathcal{D}}} \end{pmatrix}^{\mu\nu} . \tag{6.21}$$

Note that this form of the metric together with (6.20) implies in particular that S takes the product form mentioned in (6.9). In the following, we will show that for well-trained generative networks and thin data distributions (i.e. for small δ), $\gamma^{-1}_{\perp_i} \to 0$. To understand the consequences for the gradient ascent update step, consider (6.18) in y^μ coordinates

$$\begin{aligned} \gamma^{\alpha\beta} \frac{\partial f_t}{\partial x^\beta} &= \frac{\partial x^\alpha}{\partial y^\mu} \gamma^{\mu\nu} \frac{\partial f_t}{\partial y^\nu} \\ &= \frac{\partial x^\alpha}{\partial y^\mu_\parallel} (\gamma^{-1}_\mathcal{D})^{\mu\nu} \frac{\partial f_t}{\partial y^\nu_\parallel} + \frac{\partial x^\alpha}{\partial y^i_\perp} \gamma^{-1}_{\perp_i} \frac{\partial f_t}{\partial y^i_\perp} . \end{aligned} \tag{6.22}$$

For $\gamma^{-1}_{\perp_i} \to 0$ and $\frac{\partial x}{\partial y_\perp}$ bounded, the second term vanishes and we arrive at

$$\gamma^{\alpha\beta} \frac{\partial f_t}{\partial x^\beta} \to \frac{\partial x^\alpha}{\partial y^\mu_\parallel} (\gamma^{-1}_\mathcal{D})^{\mu\nu} \frac{\partial f_t}{\partial y^\nu_\parallel} \tag{6.23}$$

and hence the orthogonal directions in the update step (6.18), leading away from the data manifold $\mathcal{D}$, are suppressed. Therefore, (6.18) produces counterfactuals instead of adversarial examples.

6.3.2. Diffeomorphic counterfactuals

In this section, we show that for well-trained normalizing flows, the orthogonal components of the inverse metric $\gamma^{-1}_{\perp_i}$ vanish for thin data manifolds, as formalized in the following theorem.

Theorem 4. *For $\epsilon \in (0, 1)$ and g a normalizing flow with Kullback–Leibler divergence $\mathrm{KL}(p, q) < \epsilon$,*

$$\gamma^{-1}_{\perp_i} \to 0 \qquad \textit{as} \qquad \delta \to 0$$

for all $i \in \{1, \ldots, N_\mathcal{X} - N_\mathcal{D}\}$.

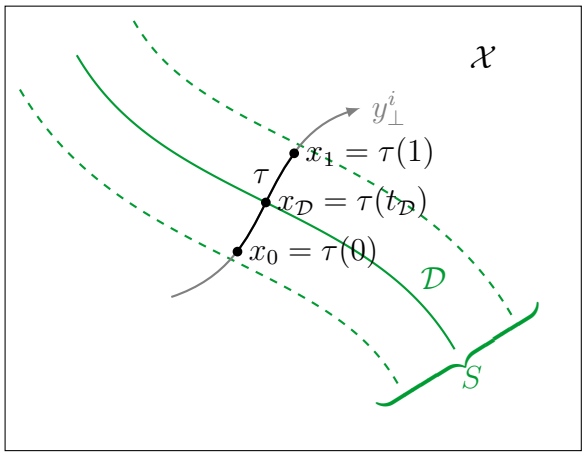


Fig. 6.6.: Construction of the curve τ used in Section 6.3.3.

The main argument of the formal proof given in Appendix C.1.1 proceeds as follows: First, we show that a small Kullback–Leibler divergence implies that most of the induced probability mass lies in the support of the data distribution,

$$\int_{S_x} q_x(x)\mathrm{d}x > 1 - \epsilon\,. \tag{6.24}$$

Next, we write q_x as the pull-back of the latent distribution q_z under the flow g using the familiar change-of-variables formula for normalizing flows. In the y^μ coordinates introduced above, the resulting integral then factorizes according to the block-diagonal structure (6.21) of the metric with integration domain $[-\delta/2, \delta/2]$ for the $y^i_\perp$ directions. As $\delta \to 0$, the bound (6.24) can only remain satisfied if the associated metric component $\gamma_{\perp_i}$ diverges, implying that $\gamma^{-1}_{\perp_i} \to 0$.

Following the steps at the end of Section 6.3.1, we see that this necessarily implies that the gradient ascent update (6.13) stays on the data manifold, since $\frac{\partial x}{\partial y_\perp}$ is constant (and therefore bounded) as $\delta \to 0$.

6.3.3. Approximately diffeomorphic counterfactuals

In Section 6.2.4, we introduced approximately diffeomorphic counterfactuals which can be obtained using VAEs or GANs. To derive a theorem similar to Theorem 4 for the case of approximately diffeomorphic counterfactuals, we will, however, need stronger assumptions since the generative models are in this case not bijective. In particular, we will assume that the generative model captures all of the data, i.e. that $\mathcal{D} \subset g(\mathcal{Z})$, implying that in y coordinates, although γ is singular for $N_\mathcal{Z} < N_\mathcal{X}$, the component $\gamma_\mathcal{D}$ is non-singular. Therefore, we split the $y_{\perp,i}$ directions into $N_\mathcal{X} - N_\mathcal{Z}$ singular directions and $N_\mathcal{Z} - N_\mathcal{D}$ non-singular directions. Since the inverse metric vanishes by definition in the singular directions, the theorem focuses on the non-singular directions and can then be stated as follows,

Theorem 5. *If $g : \mathcal{Z} \to \mathcal{X}$ is a generative model with $\mathcal{D} \subset g(\mathcal{Z})$ and image $g(\mathcal{Z})$ which extends in any non-singular orthogonal direction $y^i_\perp$ outside of $\mathcal{D}$,*

$$\gamma^{-1}_{\perp_i} \to 0$$

for $\delta \to 0$ for all non-singular orthogonal directions $y^i_\perp$.

The proof can be found in Appendix C.1.2 and proceeds as follows: First, we construct a curve $\tau : [0, 1] \to \mathcal{Z}$ which cuts through S along the $y^i_\perp$-coordinate line and lies completely in $g(\mathcal{Z})$, as illustrated in Figure 6.6. Then, the length $L(\tau)$ of this curve (with respect to γ) computed in y^μ-coordinates is, for small δ, approximately given by

$$L(\tau) \approx \sqrt{\gamma_{\perp_i}(x_\mathcal{D})}\,(x_{1,\perp}{}^i - x_{0,\perp}{}^i)\,. \tag{6.25}$$

Bounding the difference by δ and using that $L(\tau)$ is constant, yields the desired result. As in the case of Theorem 4 above, this implies again that the gradient ascent update (6.13) does not leave the data manifold as shown in (6.23).

6.4. Experiments

In this section, we show experimental results for our methods—diffeomorphic counterfactuals and approximately diffeomorphic counterfactuals—on different data sets, using various architectures for the generative models introduced in Section 6.1. For comparison we also construct adversarial examples on the same data sets.

For all experiments, we use the same setup: We require a pre-trained classifier f and start with a data point x from the test set that is predicted by the classifier f as belonging to the source class. We define the target class t and the target confidence Λ. To produce an adversarial example, we then update the original data point following the gradient in $\mathcal{X}$, $\frac{\partial f_t(x)}{\partial x}$, until we reach the desired target confidence. For the counterfactuals, we also require a pre-trained generator g. To produce a counterfactual we then first project the original data point into the latent space of the generative model g by applying the inverse generative model $g^{-1}(x) = z$, or an appropriate approximation (for GANs). We then update the original latent representation z following the gradient in $\mathcal{Z}$, $\frac{\partial (f_t \circ g)(z)}{\partial z}$, until we reach the desired target confidence.

We first illustrate results for diffeomorphic counterfactuals using a toy example in three-dimensional space. This allows us to directly visualize the data manifold and the trajectories of gradient ascent in $\mathcal{X}$ and $\mathcal{Z}$.

We then apply our diffeomorphic counterfactual method to four different image data sets, introduced in Section 2.2. We use MNIST, CelebA and CheXpert for classification tasks and the Mall data set for a regression task. We evaluate the results

qualitatively and quantitatively. Furthermore we discuss approximately diffeomorphic counterfactuals, which allow us to consider high resolution data.

For details on model configuration, training and hyperparameters we refer to Appendix C.2.

6.4.1. Diffeomorphic counterfactuals

6.4.1.1. Toy example

We consider a uniform data distribution along a one-dimensional manifold, a helix, that is embedded in three-dimensional space and train a normalizing flow with a RealNVP architecture [205]. We divide the data into two classes, corresponding to the upper and the lower half of the helix, and train a simple classifier.

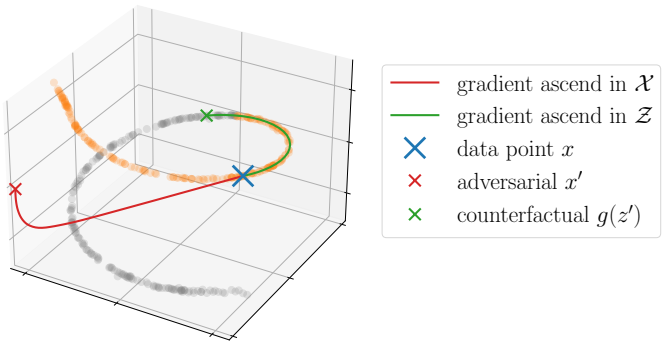


Fig. 6.7.: Gradient ascent in $\mathcal{X}$ leads to points that lie significantly off-manifold while gradient ascent in $\mathcal{Z}$ moves along the data manifold. The ground truth for different classes is depicted in orange (source class) and gray (target class).

Figure 6.7 demonstrates the application of our method on this basic setup. Starting from an original data point x on the helix, we apply gradient ascent in input space $\mathcal{X}$ using (6.12), and in the latent space of the flow $\mathcal{Z}$, using (6.13), respectively. We observe that gradient ascent in $\mathcal{X}$ leads to points that lie significantly off data manifold S. In contrast to that, the updates of gradient ascent in the latent space $\mathcal{Z}$ follow a trajectory along the data manifold resulting in a point on the helix with the desired target classification.

As, in this toy example, the data manifold can be described analytically, it is possible to calculate the distances to the data manifold for any points found via gradient ascent in $\mathcal{X}$ or $\mathcal{Z}$. For a quick, quantitative evaluation of our toy setup, we perform 1000 optimizations in the input space $\mathcal{X}$ and latent space $\mathcal{Z}$, respectively (all optimizations reach the desired target confidence), and calculate the Euclidean distances between the resulting points and the helix. The median distances when performing gradient ascent in $\mathcal{X}$ and in $\mathcal{Z}$ are 2.34 and 0.01 respectively (see also Section C.2.1 in the

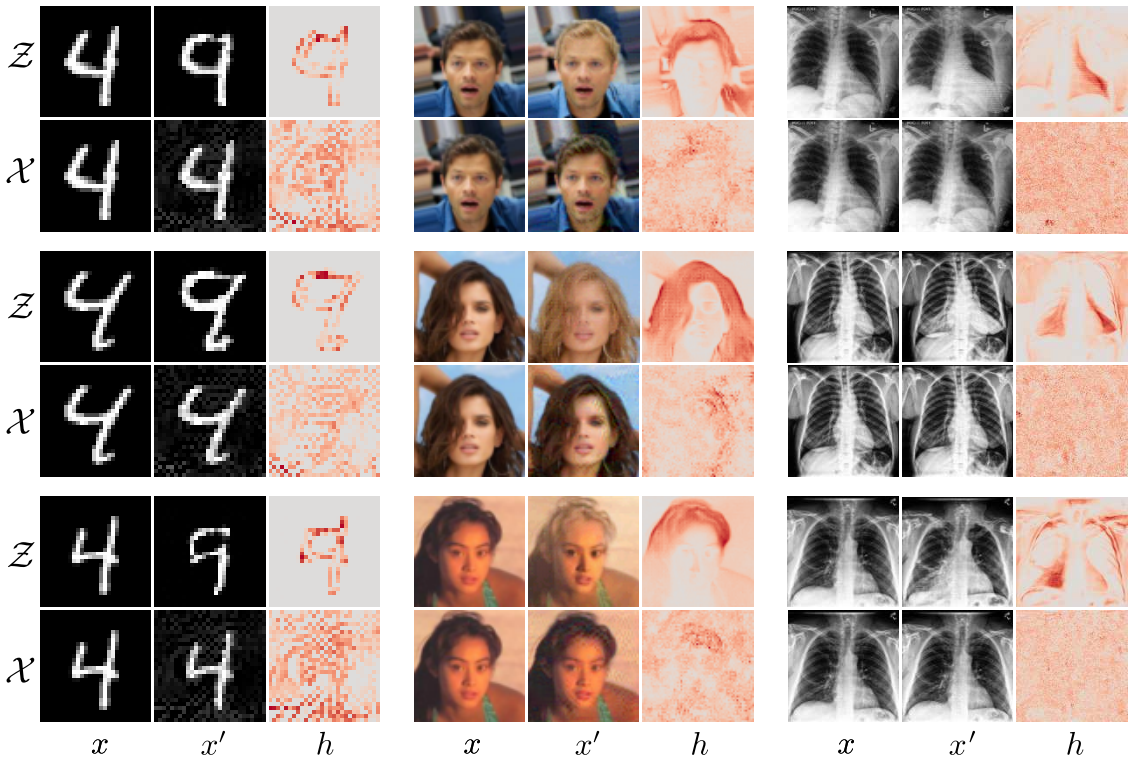


Fig. 6.8.: Counterfactuals for MNIST ('four' to 'nine'), CelebA ('not-blond' to 'blond'), and CheXpert ('healthy' to 'cardiomegaly'). Columns of each block show original image x, counterfactual x', and difference h for three selected data points. First row of each block is our diffeomorphic counterfactuals, i.e. obtained by gradient ascent in $\mathcal{Z}$ space. Second row of each block is standard gradient ascent in $\mathcal{X}$ space. Heatmaps h show the difference $|x - x'|$ summed over color channels.

appendix). This clearly illustrates the benefit of performing gradient ascent in the latent space $\mathcal{Z}$.

6.4.1.2. Image classification and regression

We now demonstrate how our method generates diffeomorphic counterfactuals for classification and regression tasks on different image data sets.

We train a ten-class CNN on MNIST (test accuracy of 99%). For CelebA and CheXpert, we train a binary CNN on the blond attribute (test accuracy of 94%) and the cardiomegaly attribute (test accuracy of 86%), respectively.

For the Mall data set we choose a regression task that estimates the number of pedestrians in the image. We train a U-Net [210] that outputs a probability map of the size of the image and a scalar regression value, which corresponds to the

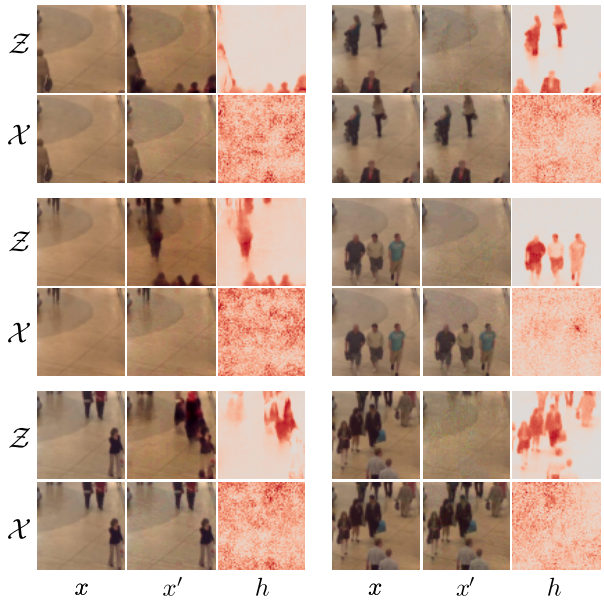


Fig. 6.9.: Counterfactuals for Mall ('few' to 'many') and Mall ('many' to 'few').

approximated number of pedestrians in the picture. Following the definitions by Ribera et al. [211], our trained U-Net reaches a RMSE for the head count of 0.63. When we run our gradient ascent algorithm, we aim to maximize, or minimize, merely the scalar regression value, i.e. the number of pedestrians.

For the generative models we choose a flow with RealNVP-type couplings [205] for MNIST and the Glow architecture [206] for CelebA, CheXpert and the Mall data set.

The generation of adversarial examples and counterfactuals then proceeds as follows: We start from original data points x of the classes 'four' for MNIST, 'not blond' for CelebA and 'healthy' for CheXpert. We select the classes 'nine', 'blond', and 'cardiomegaly' as targets t for MNIST, CelebA, and CheXpert, respectively, and take the confidence threshold to be $\Lambda = 0.99$. For the Mall data set, we maximize the regression value r (threshold at $r = 10$) if few pedestrians were identified in the original image x and minimize the regression value (threshold at $r = 0.01$) if many pedestrians were detected. We use Adam to optimize in $\mathcal{X}$ and $\mathcal{Z}$ until the confidence threshold Λ for the target class t, or the desired regression value, is reached.

We show some examples of successful optimizations in Figures 6.8 and 6.9. Our diffeomorphic counterfactuals indeed show semantically meaningful deformations, in particular when compared to the adversarial examples. The counterfactuals resemble images from the data set that have the target class as the ground truth label. At the same time the counterfactuals are similar to their respective source images with respect to features that are irrelevant for the differentiation between source and target class.

For digits from MNIST, the stroke width and the writing angle remain unchanged

in the counterfactuals while the gap in the upper part of the 'four' changes to the characteristic upper loop of the 'nine'.

For CelebA, the changes in the counterfactuals are focused on the hair area, as evident from the heatmaps h, while facial features and background stay (approximately) constant.

The counterfactuals for the CheXpert data set mostly brighten the pixels in the central region of the picture leading to the appearance of an enlarged heart. The other structures in the image remain mostly constant.

Also for pictures taken from the Mall data set, we observe that the counterfactuals remain close to the original images while showing structural changes. When maximizing the regression value, pedestrians are generated at the picture's edge or appear around darker areas in the original image. When minimizing pedestrians, we observe that the counterfactuals reproduce the darker parts of the floor and lines between the tiles.

In summary, the counterfactuals in Figures 6.8 and 6.9 reproduce semantic differences between classes that humans would deem important, suggesting that the predictive models base their decisions on the correct features.

6.4.1.3. Quantitative analysis

For a quantitative assessment of our counterfactuals we run our algorithm on a few hundred images per classification and regression task. We then use a variety of approaches to evaluate the resulting counterfacutals and adversarial examples, as detailed in the following.

Oracle As counterfactuals should resemble actual data points from the target class while adversarial examples do not, we would expect counterfactuals to generalize better to other, independently trained prediction models.

We therefore train a 10-class support vector machine (SVM) on MNIST (test accuracy of 92%) and binary SVMs on CelebA (test accuracy of 85%) and CheXpert (test accuracy of 70%). The counterfactuals found by performing gradient ascent in $\mathcal{Z}$ generalize significantly better to these simple models suggesting that they indeed use semantically more relevant deformations than conventional adversarial examples produced by gradient ascent in $\mathcal{X}$.

For the Mall data set, we train a slightly larger U-Net (RMSE for head count 0.72) and calculate regression values for the original images, the adversarial examples found in $\mathcal{X}$ and the counterfactuals found in $\mathcal{Z}$. As expected, the regression values for the counterfactuals are significantly closer to the target values ($r = 10$ when maximizing and $r = 0.01$ when minimizing) than those of original images and adversarial examples.

Figure 6.10 summarizes these findings.

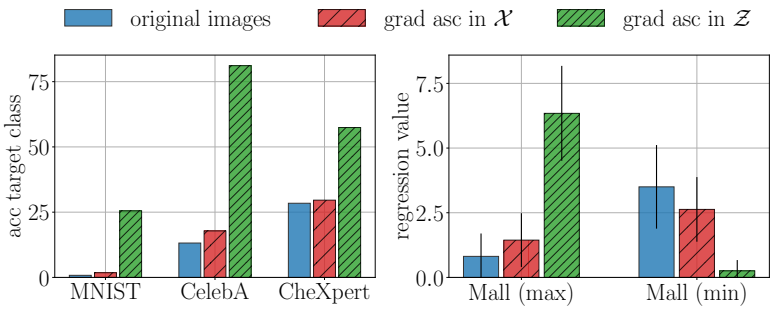


Fig. 6.10.: Left: accuracy with respect to the target class k generalizes better to SVM for diffeomorphic counterfactuals. **Right:** regression values for oracle are closer to target values for $\mathcal{Z}$-based counterfactuals (bars show means and errors denote one standard deviation).

Nearest neighbours We expect valid counterfactuals to look like images from the target class. If we consider nearest neighbours to these counterfactuals, we would also expect the nearest neighbours to belong primarily to the target class. This is in contrast to the nearest neighbours for the original images and the adversarial examples, which we expect to belong mostly to the source class.

To verify these expectations, we compare the original images and the images modified in $\mathcal{X}$ and $\mathcal{Z}$ with data from the data set. We find the k-nearest neighbours (with respect to the Euclidean norm) and their respective ground truth classification or regression value. For MNIST, CelebA and CheXpert, we then check what percentage of the nearest neighbours was classified as the target class. For Mall, we check the average number of pedestrians present in the nearest neighbours. Figure 6.11 shows that the ten nearest neighbours of the diffeomorphic counterfactuals for MNIST, CelebA and CheXpert share the target classification more often than the nearest neighbours for the original images or the adversarial examples, confirming our assumptions. For the Mall data set the three nearest neighbours of each counterfactual have, on average, regression values that more closely match the target regression value ($r = 10$ when maximizing and $r = 0.01$ when minimizing).

IM1 and IM2 Van Looveren and Klaise [212] propose two metrics to test interpretability: the first metric, IM1, is defined by

$$\text{IM1} = \frac{||x' - \text{AE}_t(x')||}{||x' - \text{AE}_{c_0}(x')|| + \epsilon}, \tag{6.26}$$

where AE_{c_0} and AE_t are two autoencoders which were each trained on data from only one class (original class c_0 and target class t, respectively) and ϵ is a small

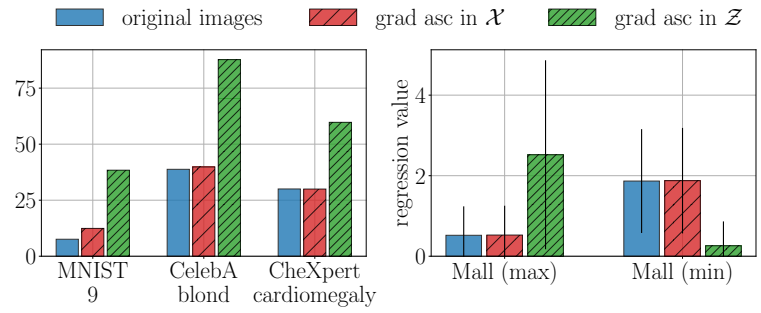


Fig. 6.11.: **Left:** ground truth class for the ten nearest neighbours (NNs) matches the target value ('9', 'blond' and 'cardiomegaly') more often for the counterfactuals found in $\mathcal{Z}$. **Right:** ground truth pedestrian counts averaged over the three nearest neighbours are closer to target values for diffeomorphic counterfactuals. Bars show means and errors denote one standard deviation.

positive value. The second metric, IM2, is defined by

$$\text{IM2} = \frac{||\text{AE}_t(x') - \text{AE}(x')||}{||x'||_1 + \epsilon}\,, \tag{6.27}$$

where AE is an autoencoder trained on all classes.

IM1 and especially IM2 have been repeatedly critisized [213–215]. For IM2, one divides by the one-norm $||x'||_1$ of the modified image. This value is large if the image has more bright pixels. Consequently, images with brighter pixels will tend to have a smaller IM2, even though they might not be more interpretable. We therefore limit our evaluation to IM1. In Table 6.1, we show mean and standard deviation for the

data set	images	IM1
MNIST	original	2.250 ± 0.711
	gradient ascent in $\mathcal{X}$	1.603 ± 0.317
	gradient ascent in $\mathcal{Z}$	1.056 ± 0.233
CelebA	original	1.160 ± 0.303
	gradient ascent in $\mathcal{X}$	1.144 ± 0.287
	gradient ascent in $\mathcal{Z}$	0.807 ± 0.222

Tab. 6.1.: Interpretability metric IM1 values for MNIST and CelebA calculated for original images, adversarial examples and counterfactuals. Low values mean better interpretability. We show mean and standard deviation.

interpretability metric IM1 for the MNIST and CelebA data sets. We calculate the values for the original images, the adversarial examples, produced by gradient ascent in $\mathcal{X}$ space, and the diffeomorphic counterfactuals, produced by gradient ascent in $\mathcal{Z}$ space. A low value for IM1 means the image is better represented by an autoencoder

trained on only the target class. Diffeomorphic counterfactuals achieve a lower IM1 value than the adversarial examples, suggesting they are more interpretable.

Similarity to original images Counterfactuals are usually required to be minimal, that is they should be the closest point to the original data point, that lies on the data manifold and reaches the desired confidence Λ with respect to the target class t. We do not encourage similarity explicitly by minimizing some distance function between the counterfactual and the original image. Other approaches penalize, for example, the Euclidean distance in $\mathcal{X}$ between the counterfactual and the original input during the search process. We argue, that the relevant distance is to be computed by the induced metric on the data manifold S or, equivalently, by the flat metric in the latent space $\mathcal{Z}$. Although we do not optimize for resemblance to the original input, our counterfactuals still preserve high similarity to the respective source images. We confirm this by calculating the Euclidean distances in $\mathcal{X}$ and $\mathcal{Z}$ between counterfactuals and all images of the source class.

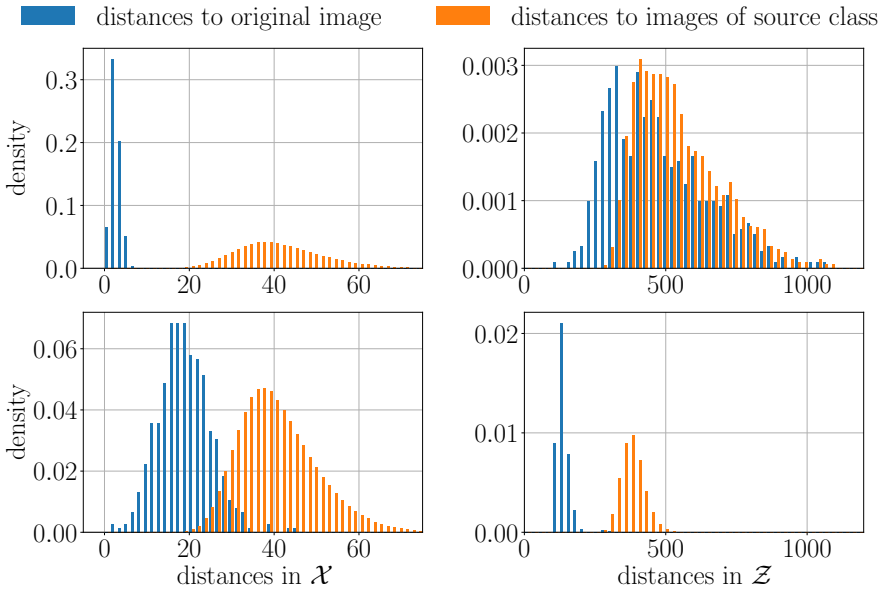


Fig. 6.12.: Euclidean distances in $\mathcal{X}$ and $\mathcal{Z}$ for adversarial examples (first row) and counterfactuals (second row) for the CelebA dataset. Counterfactuals lie closer to their respective source image than adversarial examples when measured in $\mathcal{Z}$, i.e. along the data manifold.

The average Euclidean norm between counterfactuals and the respective source images is significantly lower than the average Euclidean norm between counterfactuals and all images of the source class. For adversarial examples, we expect the Euclidean distances in $\mathcal{X}$ to the respective source image to be very small while the Euclidean distances in $\mathcal{Z}$ should be larger. Figure 6.12 illustrates this by presenting the distribution of distances in $\mathcal{X}$ and $\mathcal{Z}$ between counterfactuals or adversarials and their

respective source images as well as distances between counterfactuals or adversarials and all images of the source class for the CelebA data set.

We refer to the Appendix C.2.6 for graphs for the other data sets.

In Table 6.2, and Table 6.3 we show the averaged Euclidean norms of the distances in $\mathcal{X}$ and $\mathcal{Z}$ for counterfactuals and adversarials respectively for all the considered data sets, confirming our expectations.

data set	img	L^2 source image	L^2 source class
MNIST	in $\mathcal{X}$	2.54 ± 0.61	8.77 ± 1.25
	in $\mathcal{Z}$	4.82 ± 1.20	9.27 ± 1.32
CelebA	in $\mathcal{X}$	2.84 ± 1.15	41.13 ± 10.03
	in $\mathcal{Z}$	19.10 ± 6.08	40.79 ± 9.19
CheXpert	in $\mathcal{X}$	1.59 ± 0.46	33.68 ± 6.49
	in $\mathcal{Z}$	13.69 ± 4.54	34.71 ± 6.51
Mall (min)	in $\mathcal{X}$	1.34 ± 0.39	19.11 ± 2.67
	in $\mathcal{Z}$	10.98 ± 3.66	17.39 ± 3.10
Mall (max)	in $\mathcal{X}$	1.33 ± 0.17	9.31 ± 3.40
	in $\mathcal{Z}$	15.90 ± 5.28	19.35 ± 6.36

Tab. 6.2.: Euclidean norms L^2 in $\mathcal{X}$ for adversarial examples found via gradient ascent in $\mathcal{X}$ and counterfactuals found via gradient ascent in $\mathcal{Z}$. We show mean and standard deviation.

data set	img	L^2 source image	L^2 source class
MNIST	in $\mathcal{X}$	41.86 ± 2.00	42.12 ± 0.74
	in $\mathcal{Z}$	35.26 ± 4.68	39.91 ± 1.61
CelebA	in $\mathcal{X}$	473.72 ± 171.49	542.21 ± 149.63
	in $\mathcal{Z}$	138.35 ± 23.43	380.24 ± 41.23
CheXpert	in $\mathcal{X}$	355.50 ± 114.23	539.18 ± 88.40
	in $\mathcal{Z}$	64.90 ± 25.33	400.31 ± 48.49
Mall (min)	in $\mathcal{X}$	160.53 ± 33.67	199.45 ± 28.56
	in $\mathcal{Z}$	78.56 ± 11.84	180.39 ± 15.78
Mall (max)	in $\mathcal{X}$	142.50 ± 7.61	153.96 ± 8.80
	in $\mathcal{Z}$	116.85 ± 27.50	161.64 ± 23.99

Tab. 6.3.: Euclidean norms L^2 in $\mathcal{Z}$ for adversarial examples found via gradient ascent in $\mathcal{X}$ and counterfactuals found via gradient ascent in $\mathcal{Z}$. We show mean and standard deviation.

6.4.1.4. Tangent spaces

A non-trivial consequence of our theoretical insights is that we can infer the tangent space of each point on the data manifold from our flow g. Specifically, we perform

a singular value decomposition of the Jacobian $\frac{\partial g}{\partial z} = U\,\Sigma\,V$ and rewrite the inverse induced metric as

$$\gamma^{-1} = \frac{\partial g}{\partial z}\frac{\partial g}{\partial z}^T = U\,\Sigma^2\,U^T\,. \tag{6.28}$$

As our theoretical analysis in Section 6.3 showed, for data concentrated on an $N_{\mathcal{D}}$-dimensional data manifold $\mathcal{D}$ in an $N_{\mathcal{X}}$-dimensional embedding space $\mathcal{X}$, the inverse induced metric γ^{-1} has $N_{\mathcal{X}} - N_{\mathcal{D}}$ small eigenvalues. Furthermore, the eigenvectors corresponding to the large eigenvalues will approximately span the tangent space of the data manifold.

For our toy example from Section 6.4.1.1, we can directly show the parallelepiped at each point, spanned by the three eigenvectors in three-dimensional space. Figure 6.13 (left) indeed shows that the parallelepipeds are significantly contracted in two of the three dimensions making them appear as lines. For the high-dimensional image data sets, which are discussed in Section 6.4.1.2, we show the sorted eigenvalues, averaged over 100 random data points per data set in Figure 6.13 (right). These experiments confirm the theoretical expectation that the eigenvectors belonging to the large eigenvalues indeed span the tangent space of the manifold.

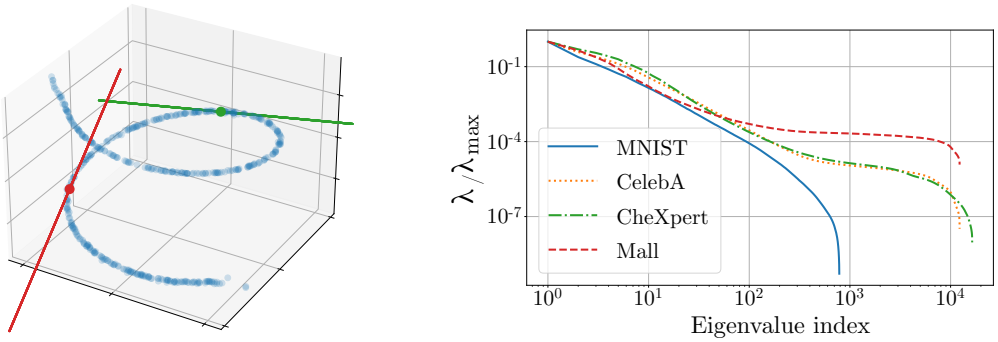


Fig. 6.13.: **Left:** As expected from the theoretical analysis, the parallelepiped spanned by all three eigenvectors of the inverse induced metric scaled by the corresponding eigenvalues is, to good approximation, one-dimensional, i.e. of the same dimension as the data manifold, and tangential to it. **Right:** The Jacobians of the trained flows have a low number of large and a large number of small eigenvalues, suggesting that the images lie approximately on a low-dimensional manifold. Both axes are scaled logarithmically.

6.4.2. Approximately diffeomorphic counterfactuals

In this section, we present our experimental analysis of approximately diffeomorphic counterfactuals for which we use VAEs and GANs. As explained in Section 6.2.4, an important downside of approximately diffeomorphic counterfactuls is that the

latent representation z of the original image x is usually lossy, i.e. $g(z) \neq x$, since the diffeomorphism is only approximate and not exact. An advantage of this approximation is that our method can be scaled to data of very high-dimensionality. Both of these statements will be demonstrated experimentally in the following. We use the same classifiers as before in Section 6.4.1.2.

MNIST and CelebA We use a simple convolutional VAE (cVAE) for the MNIST data set and find counterfactuals using gradient ascent in the latent space of the cVAE. Results are shown in Figure 6.14 in the left most block. The encoded and decoded images $\tilde{x}$ (second column of the block) appear slightly fuzzy but reproduce most characteristics of the original images. Evidently, approximately diffeomorphic counterfactuals, found by gradient ascent in the latent space of the cVAE, replicate features irrelevant for classification, such as stroke width and writing angle while structurally modifying the image so that it resembles an image of the digit nine.

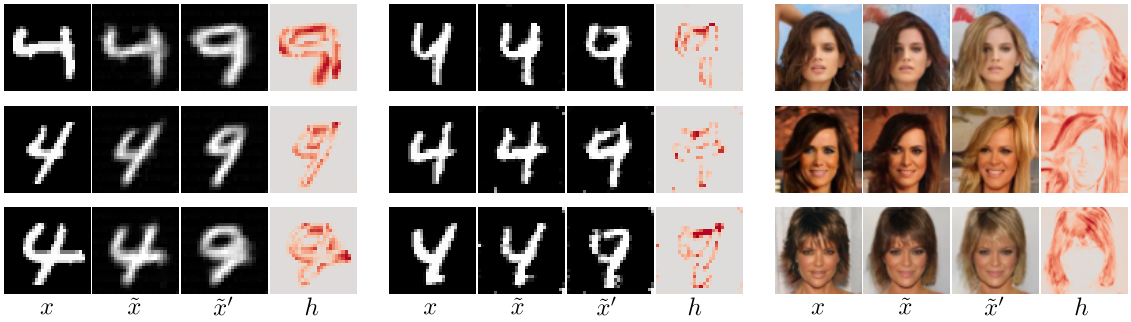


Fig. 6.14.: Counterfactuals for cVAE on MNIST (left block), dcGAN on MNIST (middle block) and pGAN on CelebA (right block). Columns of each block show original image, decoded latent representation of original, counterfactual and absolute difference $|\tilde{x} - \tilde{x}'|$ summed over color channels.

We furthermore apply our method to a simple convolutional GAN (dcGAN) [216] for MNIST and a progressive GAN (pGAN) [150] for CelebA. As discussed in Section 6.2.4, GANs do not require an encoder during the training process and we apply GAN inversion methods to find an encoding of the source image. Specifically, for MNIST and CelebA, as these are relatively low-dimensional data sets, we find a fitting latent representation z by minimizing the Euclidean norm between the decoded latent representation $g(z)$ and the original image x. Results are shown in Figure 6.14 in the middle and right block.

The dcGAN on MNIST produces some random pixel artifacts, but the generated images are sharper than those produced by the cVAE.

For the CelebA images generated with pGAN, we see that the decoded optimized latent representation of the original image deviates slightly from the original. This is especially visible if the composition is not typical (arm is not properly reproduced

in the first row) or the background is highly structured (second row). For the approximately diffeomorphic counterfactuals, we observe even larger changes in the background. This may be attributed to the imperfect inversion process and the quality of the pGAN, i.e. the fact that the diffomorphism is only approximate and not exact.

CelebA-HQ To demonstrate the scalability of approximately diffeomorphic explanations to very high-dimensional data, we use a StyleGAN [3, 4], pre-trained on the CelebA-HQ data set, as a generator for images of resolution 1024×1024. We use the HyperStyle [217] GAN-inversion techniques to find the initial latent representation of a given source image. In order to use the same classifier as before, we downscale the images to 64×64 resolution before using them as input to the classifier. As illustrated by Figure 6.15, approximately diffeomorphic counterfactuals lead to semantically meaningful and interpretable results even on this very high-dimensional data set.

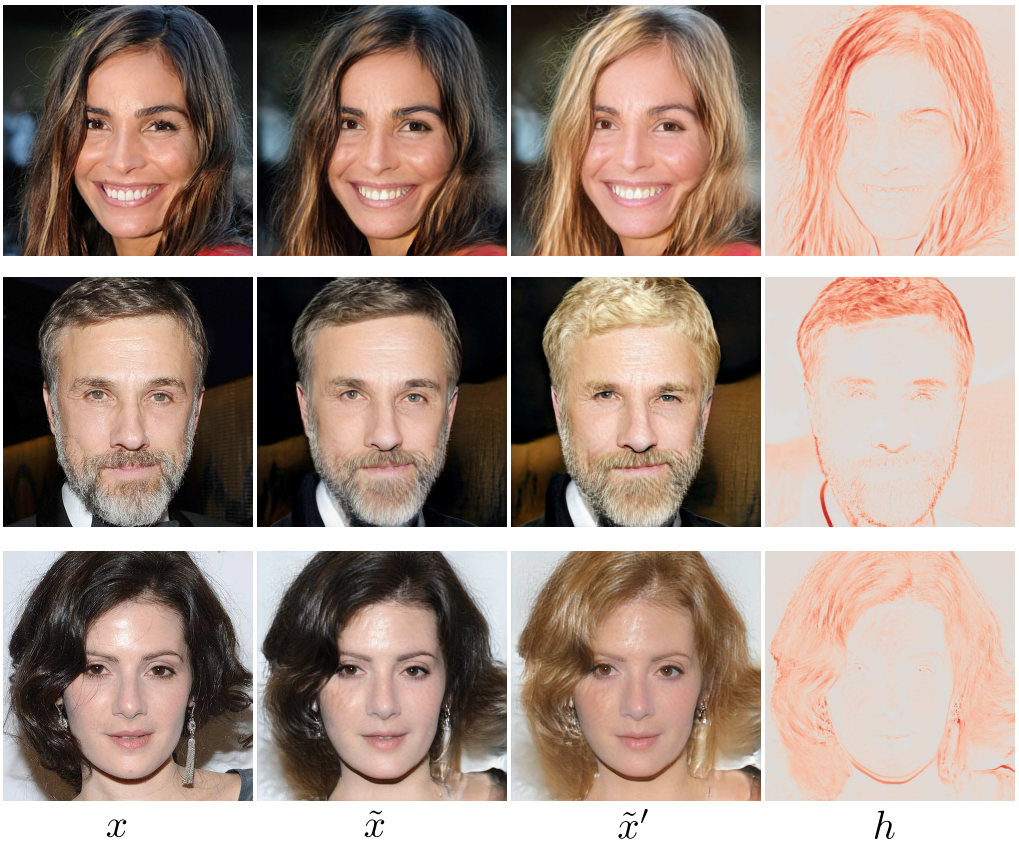


Fig. 6.15.: Counterfactuals generated with HyperStyle and Celeba-HQ. Columns show original, decoded latent representation, counterfactual and absolute difference $|\tilde{x} - \tilde{x}'|$ summed over color channels.

6.5. Related work

In this section, we compare our approach with existing methods. Our approach is closest in spirit to the one taken by Joshi et al. [126], who introduce an algorithm that applies gradient ascent in the latent space of a generative model (they mention VAEs and GANs), while minimizing the difference between original and modified data point. The application concentrates on recourse for tabular data. The examples shown for image data are limited to a VAE (which results in relatively low quality counterfactuals) and lack quantitative evaluation.

Our method is different to most other methods in a few key aspects:

- We introduce both diffeomorphic and approximately diffeomorphic counterfactuals and theoretically analyse them in a unified manner which allows us to compare the relative strengths and weaknesses of these approaches.
- Our approach is fully atomized and relies on gradients for the target class with respect to the latent representation of the image.
- We do not require additional hyperparamters, beyond those also required for finding adversarial examples.
- Our method is very modular, as we can combine pre-trained classifiers and generative models and do not require retraining for any particular architecture.
- We consider a variety of quantitative metrics to evaluate the quality of our generated counterfactuals.

A comparatively small number of publications consider normalizing flows, which started to gain attention relatively recently, in the context of generating counterfactuals.

Hvilshøj et al. [218] rely on classifier independent linear interpolation between two class centres in the base space of a flow. Sixt et al. [219] train a linear binary classifier directly in the base space of the flow. Adding the weight vector corresponding to the target class to the base space representation and projecting back to image space then produces a counterfactual with semantically changed features.

Other works use VAEs to generate counterfactuals.

Dhurandhar et al. [125] use elastic net regularization to keep the perturbation δ to the original data small and sparse. Furthermore, they use an autoencoder to minimize the reconstruction loss of the modified image and thus make sure the counterfactual lies on the data manifold. This approach was expanded by adding a prototype loss [212]. Both approaches test their algorithm on tabular data and MNIST. Kim et al. [35] specifically train a "Disentangled Causal Effect Variational Autoencoder" and then generate counterfactuals conditioned on the original image and the label they aim to change.

A number of approaches use GANs to generate counterfactuals.

Zhao et al. [138] find "natural adversarial examples" by perturbing the latent representation of a Wasserstein GAN using exhaustive search or continuous relaxation until they achieve a desired target classification.

Chang et al. [220] use a conditional GAN for infilling regions that were previously removed from the original image. Their proposed algorithm aims to find an infilling mask which maximizes or minimizes the classification confidence while penalizing the size of the region that is replaced in the original.

In some works [221–223], the classifier is incorporated in the training process of the GAN. After the training, the GAN generates counterfactuals without querying the classifier.

Lius et al. [224] use a GAN specifically trained for editing that they condition on the original query image and the desired attributes. They apply gradient descent to find attributes that cause the GAN to generate an image that the classifier predicts as the target class, while at the same time enforcing the image to be close to the original.

Similarly to [219], Shen et al. [225] train a linear SVM in the latent space of a GAN using generated images. They can then modify the latent representation of an inverted image linearly along the normal directions of the learned SVMs.

6.6. Limitations

Dependence on generative model The quality of the counterfactuals obtained with our method is largely dependent on the capacity of the underlying generative model. Challenges with generative modeling, for example mode dropping, where parts of the data manifold are not captured by the generative model, therefore also impact the counterfactuals found with our method. As the the capacity of the generative model is dependent on the underlying data set, we can see limitations for our method when features of the original starting data point are rarely present in combination with features, that are correlated in the training data with the target class. For example: our method has difficulties finding high-quality counterfactuals for the target class blond on the CelebA data set when starting from images that show people with hats, people that are bald or people that have dark skin. We show a few of these examples in Figure C.7.

Due to the dependence of our counterfactuals on the predictive model and the generative model, it might not always be clear if a correlation observed in a counterfactual is due to the predictive model using the correlated features for the classification, or due to the generative model combining the features in the approximated data distribution, or both. For example we sometimes observe increased makeup or lighter

skin tone in counterfactuals for the CelebA data set for the target class blond (see Figure C.8).

Counterfactuals for different targets Our main experiments focus on a few distinct classes. For some target classes the transition between original image and counterfactual is quite easily understood and intermediate images seem likely under the data distribution. For example when going from a picture of a person with brown hair to a picture of the same person with blond hair, we can imagine very well how intermediate pictures might look (see also Figure C.3). In contrast, some transitions between pictures of two different classes do not seem continuous and might have to traverse low probability areas. For example, going from a person not wearing a hat to a person wearing a hat seems like the transition should be quite abrupt. In addition to this, samples of images with specific attributes can be more or less likely in the training distribution and therefore more or less easily captured by the generative model. For some queries our approach might therefore work better than for others. We show counterfactuals for different queries in Appendix C.3.3.

Simple loss function Our approach includes a very basic loss function with only the hyperparameters of the optimizer (for example the learning rate) to tune. Other works incorporate regularizers to achieve sparse counterfactuals or high similarity to the source image. Our method can be seen as a baseline, which can be adapted for specific use cases. We do not explore additional requirements, such as actionability or proximity to another target class, beyond this simple loss function.

Evaluation of resulting explanations The quantitative assessment of counterfactuals is still an active research area, a summary of quantitative measures can be found in [213].

We evaluate our counterfactuals mainly with respect to two criteria: how they compare to other images of the target class and how close they lie to the original image or images from the source class, respectively. We do so by investigating how counterfactuals generalize to other classifiers, by measuring their distances to data points in $\mathcal{X}$ and $\mathcal{Z}$, and by calculating the IM1 metric. We also do visual assessments ourselves. As reference points we use our generated adversarial examples.

Other works on counterfactual explanations [33, 138, 219, 223] undertake user studies to determining the interpretability of the generated counterfactuals. One could extend our evaluation to a similar user study.

One could also evaluate the resulting counterfactuals using additional measures like the Fréchet Inception Distance (FID) score [226], applied in [35, 222, 227] or substitutability, applied in [35, 221].

Furthermore, an evaluation of the resulting *heatmaps*, that highlight the changed areas between counterfactuals and original images, using pixel flipping and related metrics [26, 164, 221, 228, 229] or evaluating the counterfactuals using the "Remove And Retrain" (ROAR) algorithm [180] could be interesting.

Our quantitative analysis is limited to our main experiments using normalizing flows as generative models. We do not provide quantitative results for counterfactuals generated with GANs or VAEs. We also do not compare our method directly with other methods that generate counterfactuals. A larger survey that compares several approaches to generating counterfactuals on different evaluation metrics might be very valuable for the ML community but is beyond the scope of this thesis.

Application beyond images Our experiments focus on different image data sets, as for images the differences between counterfactuals and adversarial examples are especially striking and easy to spot for a human. Another interesting application domain for our method would be natural language processing, especially audio signals. Audio signals (unlike written text) exist on a continuum, similar to pixel values for images. Small perturbations of adversarial attacks can therefore change the prediction of audio input without being perceptible for humans while valid counterfactuals should be clearly identifiable. We leave experiments on data beyond images for future work.

6.7. Summary

In this chapter, we proposed theoretically rigorous yet practical methods to generate counterfactuals for both classification as well as regression tasks, namely exact and approximately diffeomorphic counterfactuals. The exact diffeomorphic counterfactuals are obtained by following gradient ascent in the base space of a normalizing flow. While approximate diffeomorphisms are obtained with the help of either generative adversarial networks or variational autoencoders. Our theoretical analysis, using Riemannian differential geometry, shows that for well-trained models, our counterfactuals necessarily stay on the data manifold during the search process and consequently exhibit semantic features corresponding to the target class. Approximately diffeomorphic counterfactuals come with the risk of information loss but allow excellent scalability to higher dimensional data. Our theoretical findings are backed by experiments which both quantitatively and qualitatively demonstrate the performance of our method on different classification as well as regression tasks and for numerous data sets.

The application of our counterfactual explanation method is straightforward and requires no retraining, so that it can be readily applied to investigate common problems in deep learning like identifying biases for classifiers or training data or scrutinizing falsely classified examples.

7. Conclusion and outlook

7.1. Conclusion

Machine learning models, especially deep neural networks, are potent predictors which have cracked many long-standing challenges in scientific research and industrial applications. However, the way these models obtain their decisions remains incomprehensible to humans. As relevant information might hide in the decision process itself, not having access to it restricts the potential for discoveries. Furthermore, machine learning models might base their predictions on spurious correlations in the data, which can go undetected during development but severely limits their performance in practice.

The field of explainable artificial intelligence aims to solve these limitations by allowing us to peek inside the black box of deep neural networks. Explanation methods have already proven helpful in delivering insight into the reasoning processes of neural networks and application areas are growing steadily. However, some aspects of these methods are not adequately understood. This is problematic as we might not be able to trust such explanations.

In this thesis, we presented steps towards a unified geometrical understanding of explainability. We applied tools from differential geometry to discover limitations of explanation methods and understand the underlying reasons for certain behaviors. We then used our insights to develop robust attribution methods and interpretable counterfactual explanations.

In Chapter 4, we showed that many popular attribution methods can be arbitrarily manipulated by adding imperceptible perturbations to the input. Analyzing this surprising property theoretically led us to identify the high curvature of the neural network's output manifold as the source of the explanation's susceptibility to manipulation.

Awareness of shortcomings and unexpected properties of explanations is highly critical in scenarios where crucial decisions are based on the explanation, in addition to the prediction, of a neural network. Especially in medicine or law enforcement, it is paramount to know the limits of explanation methods in order to apply them effectively. Our discovery of how easily many popular attribution maps can be arbitrarily manipulated is therefore a call for caution to not blindly trust any explanation one might encounter.

We proceeded by seeking to mitigate this undesired behavior from different angles. We introduced a new explanation method, β-smoothing, which smooths out the kinks with diverging curvature inflicted by the popular ReLU activation function. This modification provably reduced the upper bound on the network curvature and led to explanations that are less noisy and more robust against adversarial perturbations. Furthermore, β-smoothing could be seen as a meta explanation method applicable to all attribution methods we introduced in Section 2.1.2.

In Chapter 5, we tackled the question of robustness from a different angle—by investigating how we can modify the training procedure so that the emerging neural networks have robust explanations by default. We expanded on the theoretical results of the preceding chapter and identified three approaches that limit a neural network's curvature in different manners: regularizing the weight norms, introducing smooth activation functions, and directly minimizing the Hessian norm. We demonstrated the effectiveness of these methods with numerous experiments.

Mitigating the susceptibility of explanations to manipulation and increasing their robustness guarantees their dependable results during use and thus bolsters confidence in explanation methods. Robustness against adversarial attacks is of especially great importance in safety-critical applications. The rise of legal requirements for machine learning systems, such as the 'right to explanation' in the European Union general data protection regulation, also calls for reliable explanations, as our work advocates.

Attribution methods have become popular due to their straightforward implementation and rapid results. Depending on the application and the audience, other types of explanations may offer additional insights or can be easier to understand. We therefore expanded our research to the field of counterfactual explanations.

Counterfactual explanations are very natural to interpret, which makes them an excellent choice for applications where laypeople desire explanations for the decisions of machine learning systems. One of the main characteristics of counterfactuals, as opposed to adversarial examples, is that they lie on the data manifold. Until recently, the problem of achieving good approximations of data manifolds embedded in high-dimensional spaces was still an open research question. Lately, the field of generative modeling has made significant progress towards answering this question.

In Chapter 6 we focused on counterfactual explanations. Using concepts from differential geometry, we showed how applying gradient ascent optimization of the target class in the latent space of a well-trained generative model provably moves along the data manifold. Based on these theoretical insights, we introduced an algorithm that produces counterfactuals with semantic features that resemble the target class but are otherwise very similar to the original query image. This is in contrast to gradient ascent in data space, which quickly leads off the manifold and results in adversarial examples which are indistinguishable from the original image but are predicted to be of the target class.

Our results are part of pioneering work leveraging generative models' capacity to

find high-quality counterfactuals. Our method is notably modular and easy to use since it requires no retraining. For these reasons, we expect that our method will be widely adopted.

In conclusion, we have tackled various significant limitations of explainable artificial intelligence and showed how to overcome them. Examining the problems from the viewpoint of differential geometry has enabled us to make precise theoretical statements about the properties of explanations. These abstract considerations have given us the necessary insights to develop efficient methods to boost the robustness of attribution methods and introduce an elegant algorithm to find interpretable counterfactual explanations. Our theoretical findings are supported by numerous experiments and thorough quantitative evaluations thereof.

7.2. Outlook

Research on the robustness of explanations will continue to play a major role in the development and application of explanation methods. Previous work [17, 18, 127] and our contributions [37, 38] have sparked significant interest in the unexpected behaviors of many popular explanation methods. The full extent of how concerning such manipulations of explanations are is yet to be determined. A pressing direction for future research is, for example, to find out how vulnerable explanations are to black box attacks, i.e., when there is no access to the model's gradient or architecture. This thesis already offers partial answers to this question, as we show that random input perturbations can affect the explanation and discuss that adversarially perturbed inputs generalize partially to different explanation methods but not to different architectures. Recent work [163] also found that averaging over the explanation maps of different methods increases robustness to adversarial perturbations. However, more detailed studies are required as an arms race between defense and offense concerning explanation manipulation could arise, analogous to conventional adversarial attacks [230].

Investigating analogies between our attacks and conventional adversarial attacks is a promising line of research in its own right, especially as there are similarities between the robustness measures, as proposed in Chapter 5, and efforts to improve the robustness of neural networks against adversarial attacks [231, 232]. Another research direction, that could lead to defense mechanisms, is to study the noise structure of the adversarial perturbations. These patterns might be easier to expose than for conventional adversarial attacks, as the targets for explanation manipulation are usually high dimensional and seem to correlate with the added perturbations (see Figure 4.3 and A.4).

Though in this thesis we restricted our experiments to image data, we expect the presented adversarial attack to generalize to audio and other high-dimensional continuous data without many algorithm adaptations. However, manipulating

explanations based on text could be challenging due to the discrete nature of written words. If this was confirmed, text-based explanations for images could be an excellent choice in settings where robustness against adversarial manipulation is paramount.

As our work showcases, we can counteract the manipulability of explanations by introducing robust explanation methods, like β-smoothing, or by making the network intrinsically less susceptible to input perturbations. Hybrid approaches might offer additional advantages but have gained little attention so far.

Our experimental analysis in Section 5.2.4 revealed that the effect of our robustness enhancing measures differs depending on which network architecture they are applied to. A more extensive study of how the geometry of decision surfaces differs between architectures could unveil further influences on explanation robustness. For more experimental evaluation approaches it would be advantageous to develop a unified framework for testing the robustness of explanations. First promising projects [233] are already heading in this direction.

Another research direction could investigate connections between robustness and related evaluation measures for explanations. For example, recent work [179] has aimed to quantify the uncertainty for attribution maps. It would be interesting to explore how uncertainty in explanations relates to the robustness of neural networks against manipulations of the explanation.

Research into counterfactuals for high dimensional data is only beginning to thrive and offers many exciting directions to explore. With the rise of generative models, many works have studied the latent space of these models [234–236]. Our approach to counterfactuals relates generative models to predictive models. This connection could offer interesting new directions when analyzing latent spaces. Future work could, for example, study how decision boundaries of predictive models translate into latent spaces or examine the trajectories taken by our algorithm when generating counterfactuals.

Counterfactuals can, in theory, be very diverse since multiple, quite different alternative inputs could cause a similar change in prediction. Current approaches guide the search for counterfactuals by adding additional restrictions to the loss function. Our approach offers the possibility to access directions in latent space directly. One could implement this by only calculating the gradients for specific dimensions of disentangled latent space vectors. Some generative models also have latent encodings on different structural levels (for example, the Glow architecture for normalizing flows [206] or the structure induced by progressively growing GANs [150]). For images, these different levels of latent encodings could correspond to low-level features, such as object compositions, and high-level structures, such as detailed hair structure in the CelebA data set. One could then imagine low-level and high-level counterfactuals, respectively.

We have used latent spaces of normalizing flows, GANs, and VAEs. Interesting extensions to our work could explore adaptations of our approach to generative

models without clear latent representations, such as diffusion models [237, 238] and autoregressive models [239, 240].

Our algorithm uses the gradients calculated through the generative and predictive models. In some cases, access to the predictive model might not be possible. It would be interesting to explore if high-quality counterfactuals can still be obtained if the gradient of the predictive model could only be estimated.

While we have covered methodology and theoretical justifications for diffeomorphic and approximately diffeomorphic counterfactuals we have not exhaustively explored possible applications of our method. The modularity of our proposed algorithm makes it an excellent choice for applications in medicine and industrial settings as well as for scientific research. Similar to recent approaches to detect spurious correlations with attribution methods [15, 30], counterfactuals can be used to detect such correlations in data. As counterfactuals are alternative inputs one can furthemore use these generated data samples for active learning approaches. In combination with efficient methods for human-in-the-loop strategies, our counterfactuals could help improve model accuracy. For example in medical diagnostics, a doctor could verify if the prediction model is working as intended by examining several counterfactuals. If these counterfactuals reveal that the prediction was based on dubious features, they could be labeled by the doctor and be reintroduced into a fine-tuning step. We can imagine similar procedures in industrial settings, where counterfactuals could improve the accuracy of predictive maintenance or propose actionable steps to fix a faulty system.

With recent progress in natural language processing and impressive gains in performance of text generation [2], our method to produce counterfactuals might be extended to text-based data. They could greatly facilitate the editing process, as changing a text's tone or complexity could be easily obtained if a suitable prediction model was applied. Applications involving audio data could also profit from this as our method can be used to modify recordings to be more acoustically comprehensible or change the sentiment.

We also expect our work to influence research in explainability to increasingly include theoretical considerations. We have demonstrated the value of a geometrical perspective on explainability for attribution methods and counterfactual explanations. Theoretical approaches to deepen the understanding of even more approaches to explainability, for example, concept-based explanations, could prove valuable.

In summary, this thesis has developed a theoretical understanding of general explanation methods using differential geometry. These theoretical insights allowed us to shed light on various pathological properties of explanations, such as their surprising susceptibility to manipulations. Furthermore, our theoretical insights allowed us to propose practical methods to improve the robustness of attribution methods as well as to generate high-quality counterfactuals in a provably information-preserving (diffeomorphic) fashion.

Bibliography

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, pages 4171–4186. Association for Computational Linguistics, 2019.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

[3] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019.

[4] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8110–8119, 2020.

[5] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[6] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[7] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[8] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[10] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. 2015. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[11] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*. Springer, 2019.

[12] Fabian Horst, Sebastian Lapuschkin, Wojciech Samek, Klaus-Robert Müller, and Wolfgang I Schöllhorn. Explaining the unique nature of individual gait patterns with deep learning. *Scientific reports*, 9(1):2391, 2019.

[13] Frederick Klauschen, K-R Müller, Alexander Binder, Michael Bockmayr, M Hägele, P Seegerer, Stephan Wienert, Giancarlo Pruneri, S De Maria, S Badve, et al. Scoring of tumor-infiltrating lymphocytes: From visual estimation to machine learning. In *Seminars in cancer biology*, volume 52, pages 151–157. Elsevier, 2018.

[14] Xiaoting Zhong, Brian Gallagher, Shusen Liu, Bhavya Kailkhura, Anna Hiszpanski, and T. Yong-Jin Han. Explainable machine learning in materials science. *npj Computational Materials*, 8(1):204, Sep 2022.

[15] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1):1096, 2019.

[16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, pages 1135–1144, 2016.

[17] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of Neural Networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.

[18] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 9525–9536, 2018.

[19] Timo Freiesleben. The intriguing relation between counterfactual explanations and adversarial examples. *Minds and Machines*, 32:77–109, 2021.

[20] Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. Exploring counterfactual explanations through the lens of adversarial examples: A theoretical and empirical analysis. In *International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 4574–4594. PMLR, 2022.

[21] 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.

[22] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing (NIPS)*, volume 30, pages 4765–4774. Curran Associates, Inc., 2017.

[23] 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(61):1803–1831, 2010.

[24] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[25] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the International Conference on Machine Learning, (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 2017.

[26] 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):1–46, 07 2015.

[27] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the International Conference on Machine Learning, (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017.

[28] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. In *Proceedings of the International Conference on Machine Learning (ICML), Workshop on Visualization for Deep Learning*, volume 70 of *Proceedings of Machine Learning Research*. PMLR, 2017.

[29] Miriam Hägele, Philipp Seegerer, Sebastian Lapuschkin, Michael Bockmayr, Wojciech Samek, Frederick Klauschen, Klaus-Robert Müller, and Alexander Binder. Resolving challenges in deep learning-based analyses of histopathological images using explanation methods. *Scientific Reports*, 10(1):6423, 2020.

[30] Christopher J Anders, Leander Weber, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. Finding and removing Clever Hans: Using explanation methods to debug and improve deep models. *Information Fusion*, 77:261–295, 2022.

[31] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31:841, 2017.

[32] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual Fairness. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pages 4066–4076, 2017.

[33] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual visual explanations. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2376–2384. PMLR, 2019.

[34] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[35] Hyemi Kim, Seungjae Shin, JoonHo Jang, Kyungwoo Song, Weonyoung Joo, Wanmo Kang, and Il-Chul Moon. Counterfactual fairness with disentangled causal effect variational autoencoder. *Association for the Advancement of Artificial Intelligence (AAAI)*, 35(9):8128–8136, 2021.

[36] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[37] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 13567–13578, 2019.

[38] Ann-Kathrin Dombrowski, Christopher J Anders, Klaus-Robert Müller, and Pan Kessel. Towards robust explanations for deep neural networks. *Pattern Recognition*, 121:108194, 2022.

[39] Ann-Kathrin Dombrowski, Jan E Gerken, and Pan Kessel. Diffeomorphic explanations with normalizing flows. In *Proceedings of the International Conference on Machine Learning (ICML), Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

[40] Ann-Kathrin Dombrowski, Jan E Gerken, Klaus-Robert Müller, and Pan Kessel. Diffeomorphic counterfactuals with generative models. *arXiv preprint arXiv:2206.05075*, 2022.

[41] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[42] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[44] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. NVIDIA Tesla: A unified graphics and computing architecture. *IEEE Micro*, 28(2):39–55, 2008.

[45] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the Annual International Symposium on Computer Architecture*, pages 1–12, 2017.

[46] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. *arXiv preprint arXiv:1603.04467*, 2015.

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing (NeurIPS)*, volume 32, pages 8024–8035, 2019.

[48] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[49] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[50] Andreas Holzinger, Randy Goebel, Ruth Fong, Taesup Moon, Klaus-Robert Müller, and Wojciech Samek. xxAI-Beyond Explainable Artificial Intelligence. In *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*, volume 13200 of *Lecture Notes in Computer Science*, pages 3–10. Springer, 2022.

[51] Q Vera Liao, Daniel Gruen, and Sarah Miller. Questioning the AI: informing

design practices for explainable AI user experiences. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 1–15, 2020.

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

[53] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6:52138–52160, 2018.

[54] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Workshop on explainable AI (XAI)*, volume 8, pages 8–13, 2017.

[55] 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 *Proceedings of the International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. IEEE, 2018.

[56] Christoph Molnar. *Interpretable Machine Learning*. Lulu.com, 2020. `https://christophm.github.io/interpretable-ml-book/`.

[57] David Gunning and David Aha. DARPA's explainable artificial intelligence (XAI) program. *AI Magazine*, 40(2):44–58, 2019.

[58] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. XAI - Explainable Artificial Intelligence. *Science Robotics*, 4(37):eaay7120, 2019.

[59] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[60] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable AI: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

[61] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. iNNvestigate Neural Networks! *Journal of Machine Learning Research*, 20(93):1–8, 2019.

[62] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques. *arXiv preprint arXiv:1909.03012*, 2019.

[63] 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. *arXiv preprint arXiv:2009.07896*, 2020.

[64] Raphael Meudec. tf-explain: Interpretability for Tensorflow 2.0, 2021. Software, available at `https://github.com/sicara/tf-explain`.

[65] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, page 1135–1144. Association for Computing Machinery, 2016.

[66] Jiamei Sun, Sebastian Lapuschkin, Wojciech Samek, Yunqing Zhao, Ngai-Man Cheung, and Alexander Binder. Explanation-guided training for cross-domain few-shot classification. In *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR)*, pages 7609–7616, 2021.

[67] Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115:107899, 2021.

[68] Guang Yang, Qinghao Ye, and Jun Xia. Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond. *Information Fusion*, 77:29–52, 2022.

[69] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (XAI): Toward Medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11):4793–4813, 2020.

[70] Julia Amann, Alessandro Blasimme, Effy Vayena, Dietmar Frey, and Vince I Madai. Explainability for artificial intelligence in healthcare: a multidisciplinary perspective. *BMC Medical Informatics and Decision Making*, 20(1):1–9, 2020.

[71] Yiming Zhang, Ying Weng, and Jonathan Lund. Applications of explainable artificial intelligence in diagnosis and surgery. *Diagnostics*, 12(2):237, 2022.

[72] Ouren Kuiper, Martin van den Berg, Joost van der Burgt, and Stefan Leijnen. Exploring Explainable AI in the Financial Sector: Perspectives of Banks and Supervisory Authorities. In *Proceedings of the Benelux Conference on Artificial Intelligence (BNAIC)*, pages 105–119. Springer, 2021.

[73] Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. Explainable machine learning in credit risk management. *Computational Economics*, 57(1):203–216, 2021.

[74] Philippe Bracke, Anupam Datta, Carsten Jung, and Shayak Sen. Machine learning explainability in finance: an application to default risk analysis. *Bank of England Working Paper*, 2019.

[75] Sebastian Fritz-Morgenthal, Bernhard Hein, and Jochen Papenbrock. Financial Risk Management and Explainable, Trustworthy, Responsible AI. *Frontiers in Artificial Intelligence*, 5(779799):1–14, 2022.

[76] Ian Sample. Computer says no: why making AIs fair, accountable and transparent is crucial. *The Guardian*, 5:1–15, 2017.

[77] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 28 of *Proceedings of Machine Learning Research*, pages 325–333. PMLR, 2013.

[78] Bryce Goodman and Seth Flaxman. European Union regulations on algorithmic decision-making and a "Right to Explanation". *AI Magazine*, 38(3):50–57, 2017.

[79] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017.

[80] Andrew Selbst and Julia Powles. "Meaningful Information" and the Right to Explanation. In *Proceedings of the Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 48–48. PMLR, 2018.

[81] Sandra Wachter. Normative challenges of identification in the Internet of Things: Privacy, profiling, discrimination, and the GDPR. *Computer Law & Security Review*, 34(3):436–449, 2018.

[82] Ashley Deeks. The judicial demand for explainable artificial intelligence. *Columbia Law Review*, 119(7):1829–1850, 2019.

[83] Arzam Kotriwala, Benjamin Klöpper, Marcel Dix, Gayathri Gopalakrishnan, Dawid Ziobro, and Andreas Potschka. XAI for Operations in the Process Industry-Applications, Theses, and Research Directions. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, pages 1–12, 2021.

[84] SG Zeldam. Automated failure diagnosis in aviation maintenance using explainable artificial intelligence (XAI). Master's thesis, University of Twente, 2018.

[85] Athar Kharal. Explainable artificial intelligence based fault diagnosis and insight harvesting for steel plates manufacturing. *arXiv preprint arXiv:2008.04448*, 2020.

[86] Bahrudin Hrnjica and Selver Softic. Explainable AI in manufacturing: a predictive maintenance case study. In *Proceedings of the International Conference on Advances in Production Management Systems (APMS)*, pages 66–73. Springer, 2020.

[87] Martin W Hoffmann, Rainer Drath, and Christopher Ganz. Proposal for requirements on industrial AI solutions. In *Machine Learning for Cyber Physical Systems*, pages 63–72. Springer, 2021.

[88] John Ashworth Nelder and Robert WM Wedderburn. Generalized Linear Models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

[89] Peter McCullagh and John Ashworth Nelder. *Generalized Linear Models*, volume 37 of *Monographs on Statistics and Applied Probability*. Chapman & Hall / CRC, 1989.

[90] Trevor Hastie and Robert Tibshirani. Generalized additive models: Some applications. *Journal of the American Statistical Association*, 82(398):371–386, 1987.

[91] J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.

[92] Ryan J Urbanowicz and Jason H Moore. Learning classifier systems: a complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*, 2009:736398, 2009.

[93] Chengqi Zhang and Shichao Zhang. *Association Rule Mining, Models and Algorithms*, volume 2307 of *Lecture Notes in Computer Science*. Springer, 2002.

[94] Leandro Nunes De Castro, Leandro Nunes Castro, and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[95] David D Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 4–15. Springer, 1998.

[96] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[97] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! Criticism for interpretability. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29, pages 2280–2288, 2016.

[98] Karthik S Gurumoorthy, Amit Dhurandhar, Guillermo Cecchi, and Charu Aggarwal. Efficient data representation by selecting prototypes with importance weights. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 260–269. IEEE, 2019.

[99] 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 *Proceedings*

of the International Conference on Machine Learning (ICML), volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677. PMLR, 2018.

[100] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348. PMLR, 2020.

[101] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

[102] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[103] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[104] Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015.

[105] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Series in Statistics. Springer, 2009.

[106] Jerome H Friedman and Bogdan E Popescu. Predictive learning via rule ensembles. *The annals of applied statistics*, 2(3):916–954, 2008.

[107] Daniel W Apley and Jingyu Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(4):1059–1086, 2020.

[108] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[109] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29, pages 3387–3395, 2016.

[110] R Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18, 1977.

[111] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 2017.

[112] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 9277–9286, 2019.

[113] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 818–833, 2014.

[114] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3429–3437, 2017.

[115] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized Input Sampling for Explanation of Black-box Models. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 151–164, 2018.

[116] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 1792–1801. PMLR, 2018.

[117] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 1527–1535, 2018.

[118] Alvin E Roth. *The Shapley value: Essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[119] 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 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[120] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for Simplicity: The All Convolutional Net. In *Proceedings of the International Conference on Learning Representations (ICLR), Workshop contribution*, 2015.

[121] 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/CVF International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[122] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2956–2964, 2015.

[123] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene CNNs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[124] Piotr Dabkowski and Yarin Gal. Real Time Image Saliency for Black Box Classifiers. In *Advances in Neural Information Processing (NIPS)*, volume 30, pages 6967–6976. Curran Associates, Inc., 2017.

[125] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 590–601, 2018.

[126] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. In *Proceedings of the International Conference on Machine Learning (ICML), Safe Machine Learning Workshop*, 2019.

[127] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (Un)reliability of Saliency Methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 267–280. Springer, 2019.

[128] Marco Ancona, Enea Ceolini, Cengiz Oztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[129] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise Relevance Propagation: An Overview. In *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 193–209. Springer, 2019.

[130] Stefan Haufe, Frank Meinecke, Kai Görgen, Sven Dähne, John-Dylan Haynes, Benjamin Blankertz, and Felix Bießmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *Neuroimage*, 87:96–110, 2014.

[131] Milda Pocevičiūtė, Gabriel Eilertsen, and Claes Lundström. Survey of XAI in digital pathology. In *Artificial intelligence and machine learning for digital pathology*, volume 12090 of *Lecture Notes in Computer Science*, pages 56–88. Springer, 2020.

[132] Ilia Stepin, Jose M Alonso, Alejandro Catala, and Martín Pereira-Fariña. A

survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9:11974–12001, 2021.

[133] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (ACM FAccT)*, pages 607–617, 2020.

[134] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 156–163, 2017.

[135] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *Proceedings of the International Conference on Learning Representations, (ICLR)*, 2019.

[136] Simran Kaur, Jeremy Cohen, and Zachary C Lipton. Are perceptually-aligned gradients a general property of robust classifiers? *arXiv preprint arXiv:1910.08640*, 2019.

[137] Kieran Browne and Ben Swift. Semantics and explanation: why counterfactual explanations produce adversarial examples in deep neural networks. *arXiv preprint arXiv:2012.10076*, 2020.

[138] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[139] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017. `https://distill.pub/2017/feature-visualization`.

[140] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015.

[141] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196, 2015.

[142] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog*, 2015.

[143] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. In *Proceedings of the International Conference on Machine Learning (ICML), Visualization for Deep Learning Workshop*, 2016.

[144] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[145] Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[146] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. Dataset, available at `https://www.cs.toronto.edu/~kriz/cifar.html`.

[147] 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.

[148] Jeremy Howard. ImageNette, 2020. Dataset, available at `https://github.com/fastai/imagenette/`.

[149] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[150] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[151] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.

[152] Ke Chen, Chen Change Loy, Shaogang Gong, and Tony Xiang. Feature mining for localised crowd counting. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 21.1–21.11, 2012.

[153] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[154] John A Thorpe. *Elementary topics in differential geometry*. Undergraduate Texts in Mathematics. Springer, 2012.

[155] Loring W Tu. *Differential geometry: connections, curvature, and characteristic classes*, volume 275 of *Graduate Texts in Mathematics*. Springer, 2017.

[156] John M Lee. *Riemannian manifolds: an introduction to curvature*, volume 176 of *Graduate Texts in Mathematics*. Springer, 2006.

[157] Mike Bostock, Philippe Rivière, Chris Uehlinger, and Noah Veltman. Geographic projections, spherical shapes and spherical trigonometry (d3-geo). `https://github.com/d3/d3-geo`.

[158] Christopher Anders, Plamen Pasliev, Ann-Kathrin Dombrowski, Klaus-Robert Müller, and Pan Kessel. Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 314–323. PMLR, 2020.

[159] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[160] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, volume 25, pages 1097–1105. Curran Associates Inc., 2012.

[161] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[162] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[163] Laura Rieger and Lars Kai Hansen. A simple defense against adversarial attacks on heatmap explanations. In *Proceedings of the International Conference on Machine Learning (ICML), Workshop on Human Interpretability in Machine Learning*, 2020.

[164] Wojciech Samek, Alexander Binder, Gregoire 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*, 28:2660–2673, 11 2017.

[165] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[166] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. Interpretable deep learning under fire. In *Security Symposium (USENIX))*, volume 29, pages 1659–1676, 2020.

[167] Akshayvarun Subramanya, Vipin Pillai, and Hamed Pirsiavash. Towards

hiding adversarial examples from network interpretation. *arXiv preprint arXiv:1812.02843*, 2018.

[168] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Suggala, David I Inouye, and Pradeep K Ravikumar. On the (in) fidelity and sensitivity of explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 10967–10978, 2019.

[169] Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling Neural Network Interpretations via Adversarial Model Manipulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 2921–2932, 2019.

[170] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[171] Botty Dimanov, Umang Bhatt, Mateja Jamnik, and Adrian Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In *European Association for Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2473–2480, 2020.

[172] Tom Viering, Ziqi Wang, Marco Loog, and Elmar Eisemann. How to manipulate CNNs to make them lie: the GradCAM case. In *Proceedings of the British Machine Vision Conference (BMVC), Workshop on Interpretable and Explainable Machine Vision*, 2019.

[173] Himabindu Lakkaraju and Osbert Bastani. "How do I fool you?" Manipulating User Trust via Misleading Black Box Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 79–85, 2020.

[174] Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3719–3728, 2018.

[175] 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*, pages 247–258. Association for Computational Linguistics, November 2020.

[176] Mengdi Huai, Jianhui Sun, Renqin Cai, Liuyi Yao, and Aidong Zhang. Malicious attacks against deep reinforcement learning interpretations. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining (ACM SIGKDD)*, pages 472–482, 2020.

[177] Sushant Agarwal, Shahin Jabbari, Chirag Agarwal, Sohini Upadhyay, Steven Wu, and Himabindu Lakkaraju. Towards the unification and robustness of

perturbation and gradient based explanations. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 110–119. PMLR, 18–24 Jul 2021.

[178] Dylan Slack, Anna Hilgard, Sameer Singh, and Himabindu Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 9391–9404. Curran Associates, Inc., 2021.

[179] Kirill Bykov, Marina M-C Höhne, Klaus-Robert Müller, Shinichi Nakajima, and Marius Kloft. How Much Can I Trust You? — Quantifying Uncertainties in Explaining Neural Networks. *arXiv preprint arXiv:2006.09000*, 2020.

[180] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A Benchmark for Interpretability Methods in Deep Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 9734–9745, 2019.

[181] Anders Krogh and John A Hertz. A Simple Weight Decay Can Improve Generalization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 5, pages 950–957, 1992.

[182] Andreas S Weigend, David E Rumelhart, and Bernardo A Huberman. Generalization by Weight-Elimination with Application to Forecasting. In *Advances in Neural Information Processing Systems (NIPS)*, volume 4, pages 875–882, 1991.

[183] Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 1, pages 177–185, 1988.

[184] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9078–9086, 2019.

[185] Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

[186] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29, pages 2613–2621, 2016.

[187] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1778–1787, 2018.

[188] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 1831–1841, 2019.

[189] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security*, pages 39–49, 2017.

[190] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020.

[191] Zifan Wang, Haofan Wang, Shakul Ramkumar, Piotr Mardziel, Matt Fredrikson, and Anupam Datta. Smoothed geometry for robust attribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 13623–13634, 2020.

[192] Himabindu Lakkaraju, Nino Arsov, and Osbert Bastani. Robust and stable black box explanations. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 5628–5638. PMLR, 2020.

[193] Badri Patro, Shivansh Patel, and Vinay Namboodiri. Robust explanations for visual question answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1577–1586, March 2020.

[194] Guohang Zeng, Yousef Kowsar, Sarah Erfani, and James Bailey. Generating deep networks explanations with robust attribution alignment. In *Proceedings of the Asian Conference on Machine Learning (ACML)*, volume 157 of *Proceedings of Machine Learning Research*, pages 753–768. PMLR, 2021.

[195] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations, (ICLR)*, 2014.

[196] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taïga, Francesco Visin, David Vázquez, and Aaron C. Courville. Pixelvae: A latent variable model for natural images. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[197] Huaibo Huang, Ran He, Zhenan Sun, and Tieniu Tan. IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 52–63, 2018.

[198] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 19667–19679, 2020.

[199] Jakob D Drachmann Havtorn, Jes Frellsen, Soren Hauberg, and Lars Maaløe. Hierarchical vaes know what they don't know. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 4117–4128. PMLR, 2021.

[200] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, volume 27, pages 2672–2680, 2014.

[201] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2794–2802, 2017.

[202] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1857–1865. PMLR, 2017.

[203] Amjad Almahairi, Sai Rajeshwar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 195–204. PMLR, 2018.

[204] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538. PMLR, Jul 2015.

[205] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using RealNVP. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[206] Durk P Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 10215–10224. Curran Associates, Inc., 2018.

[207] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[208] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 7511–7522, 2019.

[209] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. GAN Inversion: A Survey. *arXiv preprint arXiv:2101.05278*, 2021.

[210] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 12261 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.

[211] Javier Ribera, David Guera, Yuhao Chen, and Edward J Delp. Locating objects without bounding boxes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6479–6489, 2019.

[212] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 650–665. Springer, 2021.

[213] Frederik Hvilshøj, Alexandros Iosifidis, and Ira Assent. On quantitative evaluations of counterfactuals. *arXiv preprint arXiv:2111.00177*, 2021.

[214] Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. In *Advances in Neural Information Processing Systems (NeurIPS), CausalML: Machine Learning and Causal Inference for Improved Decision Making Workshop*, 2019.

[215] Lisa Schut, Oscar Key, Rory Mc Grath, Luca Costabello, Bogdan Sacaleanu, Yarin Gal, et al. Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130 of *Proceedings of Machine Learning Research*, pages 1756–1764. PMLR, 2021.

[216] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[217] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit H Bermano. HyperStyle: StyleGAN Inversion with HyperNetworks for Real Image Editing. *arXiv preprint arXiv:2111.15666*, 2021.

[218] Frederik Hvilshøj, Alexandros Iosifidis, and Ira Assent. ECINN: Efficient Counterfactuals from Invertible Neural Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2021.

[219] Leon Sixt, Martin Schuessler, Philipp Weiß, and Tim Landgraf. Interpretability

Through Invertibility: A Deep Convolutional Network With Ideal Counterfactuals And Isosurfaces, 2021. https://openreview.net/pdf?id=8YFhXYe1Ps.

[220] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[221] Pouya Samangouei, Ardavan Saeedi, Liam Nakagawa, and Nathan Silberman. ExplainGAN: Model Explanation via Decision Boundary Crossing Transformations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–681, 2018.

[222] Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. Explanation by Progressive Exaggeration. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[223] Oran Lang, Yossi Gandelsman, Michal Yarom, Yoav Wald, Gal Elidan, Avinatan Hassidim, William T Freeman, Phillip Isola, Amir Globerson, Michal Irani, et al. Explaining in Style: Training a GAN to explain a classifier in StyleSpace. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 693–702, 2021.

[224] Shusen Liu, Bhavya Kailkhura, Donald Loveland, and Yong Han. Generative Counterfactual Introspection for Explainable Deep Learning. In *Proceedings of the Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5, 2019.

[225] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. InterFaceGAN: Interpreting the disentangled face representation learned by GANs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 44(4):2004–2018, 2022.

[226] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pages 6626–6637, 2017.

[227] Robin Rombach, Patrick Esser, and Björn Ommer. Making sense of CNNs: Interpreting deep representations & their invariances with INNs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 12362 of *Lecture Notes in Computer Science*, pages 647–664. Springer, 2020.

[228] Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 159–168. Association for Computational Linguistics, September 2017.

[229] Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and

Alun Preece. Sanity Checks for Saliency Metrics. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6021–6029, Apr. 2020.

[230] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[231] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 6541–6550. Curran Associates, Inc., 2018.

[232] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pages 2266–2276. Curran Associates, Inc., 2017.

[233] Anna Hedström, Leander Weber, Dilyara Bareeva, Franz Motzkus, Wojciech Samek, Sebastian Lapuschkin, and Marina M.-C. Höhne. Quantus: An Explainable AI Toolkit for Responsible Evaluation of Neural Network Explanations. *arXiv preprint arXiv:2202.06861*, 2022.

[234] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9243–9252, 2020.

[235] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[236] Wei Jiang, Richard Lee Davis, Kevin Gonyop Kim, and Pierre Dillenbourg. GANs for All: Supporting Fun and Intuitive Exploration of GAN Latent Spaces. In *Advances in Neural Information Processing Systems (NeurIPS), Competitions and Demonstrations Track*, volume 176, pages 292–296. PMLR, 2022.

[237] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265. PMLR, 2015.

[238] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020.

[239] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the International Conference on*

Machine Learning (ICML), volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756. PMLR, 2016.

[240] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

A. Appendices for Chapter 4

A.1. Proofs

In this section, we collect the proofs of the theorems stated in Chapter 4.

A.1.1. Theorem 1

Theorem 1. *Let $f : \mathbb{R}^N \to \mathbb{R}$ be a network with softplus$_\beta$ non-linearities and $\mathcal{U}_\epsilon(p) = \{x \in \mathbb{R}^N; \|x - p\| < \epsilon\}$ an environment of a point $p \in S$ such that $\mathcal{U}_\epsilon(p) \cap S$ is fully connected. Let f have bounded derivatives $\|\nabla f(x)\| \geq c$ for all $x \in \mathcal{U}_\epsilon(p) \cap S$. It then follows for all $p_0 \in \mathcal{U}_\epsilon(p) \cap S$ that*

$$\|h(p) - h(p_0)\| \leq |\lambda_{max}|\, d_g(p, p_0) \leq \beta\, C\, d_g(p, p_0), \tag{A.1}$$

where λ_{max} is the principal curvature with the largest absolute value for any point in $\mathcal{U}_\epsilon(p) \cap S$ and the constant $C > 0$ depends on the weights of the neural network.

Proof: This proof will proceed in four steps. We will first bound the Frobenius norm of the Hessian of the network f. From this, we will deduce an upper bound on the Frobenius norm of the second fundamental form. This in turn will allow us to bound the largest principal curvature $|\lambda_{\max}| = \max\{|\lambda_1| \dots |\lambda_{d-1}|\}$. Finally, we will bound the maximal and minimal change in explanation.

Step 1: Let $\text{softplus}^{(l)}(x) = \text{softplus}(W^{(l)}x)$ where $W^{(l)}$ are the weights of layer l.[1] We note that

$$\partial_k \text{softplus}(\sum_j W_{ij}x_j) = W_{ik}\, \sigma(\sum_j W_{ij}x_j) \tag{A.2}$$

$$\partial_l \sigma(\sum_j W_{ij}x_j) = \beta\, W_{il}\, g(\sum_j W_{ij}x_j)) \tag{A.3}$$

where

$$\sigma(x) = \frac{1}{(1 + e^{-\beta x})}\,, \qquad g(x) = \frac{1}{(e^{\beta x/2} + e^{-\beta x/2})^2}\,. \tag{A.4}$$

[1] We do not make the dependence of softplus on its β parameter explicit and ignore biases to ease notation.

The activation at layer L is then given by

$$a^{(L)}(x) = (\text{softplus}^{(L)} \circ \cdots \circ \text{softplus}^{(1)})(x) \tag{A.5}$$

Its derivative $\partial_k a_i^{(L)}$ is equal to

$$\sum_{s_2 \dots s_L} W_{is_L}^{(L)} \sigma \left(\sum_j W_{ij}^{(L)} a_j^{(L-1)} \right) W_{s_L s_{L-1}}^{(L-1)} \sigma \left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)} \right) \dots W_{s_2 k}^{(1)} \sigma \left(\sum_j W_{s_2 j}^{(1)} x_j \right)$$

As $|\sigma(\bullet)| \leq 1$ we therefore obtain

$$\left\| \nabla a^{(L)} \right\| \leq \prod_{l=1}^{L} \left\| W^{(l)} \right\|_F \tag{A.6}$$

Deriving the expression for $\partial_k a_i^{(L)}$ again, we obtain

$$\begin{aligned}
\partial_l \partial_k a_i^{(L)} = & \sum_m \sum_{s_2 \dots s_L} \{ \\
& W_{is_L}^{(L)} \sigma \left(\sum_j W_{ij}^{(L)} a_j^{(L-1)} \right) W_{s_L s_{L-1}}^{(L-1)} \sigma \left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)} \right) \\
& \dots \beta \sum_{\hat{s}_m} W_{s_{m+1} \hat{s}_m}^{(m)} W_{s_{m+1} s_m}^{(m)} g \left(\sum_j W_{s_{m+1} j}^{(m)} a_j^{(m-1)}(x) \right) \partial_l a_{\hat{s}_m}^{(m-1)}(x) \\
& \dots W_{s_2 k}^{(1)} \sigma \left(\sum_j W_{s_2 j}^{(1)} x_j \right) \}
\end{aligned}$$

We now restrict to the case for which the index i only takes a single value and use $|\sigma(\bullet)| \leq 1$ and $|g(\bullet)| \leq \frac{1}{4}$. The Hessian $H_{ij} = \partial_i \partial_j a^L(x)$ is then bounded by

$$\|H\|_F \leq \beta \tilde{C} \tag{A.7}$$

where the constant is given by

$$\tilde{C} = \sum_m \left\| W^{(L)} \right\|_F \left\| W^{(L-1)} \right\|_F \dots \left\| W^{(m)} \right\|_F^2 \dots \left\| W^{(1)} \right\|_F^2 . \tag{A.8}$$

Step 2: Let $\{e_1, \dots, e_{d-1}\}$ be a basis of the tangent space $T_p S$. Then the second fundamental form for the hypersurface $f(x) = c$ at point p is given by

$$\mathcal{L}(e_i, e_j) = -\langle D_{e_i} n(p), e_j \rangle \tag{A.9}$$

$$= -\langle D_{e_i} \frac{\nabla f(p)}{\|\nabla f(p)\|}, e_j \rangle \tag{A.10}$$

$$= -\frac{1}{\|\nabla f(p)\|} \langle H[f] e_i, e_j \rangle + (\dots) \langle \nabla f(p), e_j \rangle \tag{A.11}$$

We now use the fact that $\langle \nabla f(p), e_j \rangle = 0$, i.e. the gradient of f is normal to the tangent space. This property was explained in the main text. This allows us to deduce that

$$\mathcal{L}(e_i, e_j) = -\frac{1}{\|\nabla f(p)\|} H[f]_{ij} \,. \tag{A.12}$$

Step 3: The Frobenius norm of the second fundamental form (considered as a matrix in the sense of step 2) can be written as

$$\|\mathcal{L}\|_F = \sqrt{\lambda_1^2 + \cdots + \lambda_{d-1}^2} \,, \tag{A.13}$$

where λ_i are the principal curvatures. This property follows from the fact that the second fundamental form is symmetric and can therefore be diagonalized with real eigenvectors, e.g. the principal curvatures. Using the fact that the derivative of the network is bounded from below, $\|\nabla f(p)\| \geq c$, we obtain

$$|\lambda_{\max}| \leq \beta \, \frac{\tilde{C}}{c} \,. \tag{A.14}$$

Step 4: For $p, p_0 \in \mathcal{U}_\epsilon(p) \cap S$, we choose a curve τ with $\tau(t_0) = p_0$ and $\tau(t) = p$. Furthermore, we use the notation $u(t) = \tau'(t)$. It then follows that

$$n(p) - n(p_0) = \int_{t_0}^{t} \frac{d}{dt} \left(n(\tau(t)) \right) \, \mathrm{d}t = \int_{t_0}^{t} D_{u(t)} n(\tau(t)) \, \mathrm{d}t \tag{A.15}$$

Using the fact that $D_{u(t)} n(\tau(t)) \in T_{\tau(t)} S$ and choosing an orthonormal basis $e_i(t)$ for the tangent spaces, we obtain

$$\int_{t_0}^{t} D_{u(t)} n(\tau(t)) \, \mathrm{d}t = \int_{t_0}^{t} \sum_j \langle e_j(t), D_{u(t)} n(\tau(t)) \rangle \, e_j(t) \, \mathrm{d}t \tag{A.16}$$

$$= \int_{t_0}^{t} \sum_j \mathcal{L}(e_j(t), u(t)) \, e_j(t) \, \mathrm{d}t \,. \tag{A.17}$$

The second fundamental form $\mathcal{L}$ is bilinear and therefore

$$\int_{t_0}^{t} \sum_i \mathcal{L}(e_j(t), u(t)) \, e_j(t) \, \mathrm{d}t = \int_{t_0}^{t} \sum_{i,j} \mathcal{L}(e_j(t), e_i(t)) \, u^i(t) \, e_j(t) \, \mathrm{d}t \,. \tag{A.18}$$

We now use the notation $\mathcal{L}_{ij}(t) = \mathcal{L}(e_j(t), e_i(t))$ and choose its eigenbasis for $e_i(t)$. We then obtain for the difference in the unit normals:

$$n(p) - n(p_0) = \int_{t_0}^{t} \sum_i \lambda_i(t) \, u^i(t) \, e_i(t) \, \mathrm{d}t \,, \tag{A.19}$$

where $\lambda_i(t)$ denote the principal curvatures at $\tau(t)$. By orthonormality of the eigenbasis, it can be easily checked that

$$\langle \sum_i \lambda_i(t)\, u^i(t)\, e_i(t), \sum_j \lambda_j(t)\, u^j(t)\, e_j(t) \rangle \leq |\lambda_{\max}|^2 \sum_i u^i(t)^2$$
$$\Rightarrow \left\| \sum_i \lambda_i(t)\, u^i(t)\, e_i(t) \right\| \leq |\lambda_{\max}|\, \|u(t)\|$$

Using this relation and the triangle inequality, we then obtain by taking the norm on both sides of (A.19):

$$\|n(p) - n(p_0)\| \leq |\lambda_{\max}| \int_{t_0}^{t} \|\tau'(t)\|\, \mathrm{d}t\,. \tag{A.20}$$

This inequality holds for any curve connecting p and p_0 but the tightest bound follows by choosing τ to be the shortest possible path in $\mathcal{U}_\epsilon(p) \cap S$ with length $\int_{t_0}^{t} \|\tau'(t)\|\, dt$, i.e. the geodesic distance $d_g(p, p_0)$ on $\mathcal{U}_\epsilon(p) \cap S$. The second inequality of the theorem is obtained by the upper bound on the largest principal curvature $\lambda_{\max}$ derived above, i.e. (A.14).

A.1.2. Theorem 2

Theorem 2. *For one layer neural networks $f(x) = ReLU(w^T x)$ and $f_\beta(x) = softplus_\beta(w^T x)$, it holds that*

$$\mathbb{E}_{\epsilon \sim p_\beta} \left[\nabla f(x - \epsilon) \right] = \nabla f_{\frac{\beta}{\|w\|}}(x)\,, \tag{A.21}$$

where $p_\beta(\epsilon) = \frac{\beta}{(e^{\beta\epsilon/2} + e^{-\beta\epsilon/2})^2}$.

Proof: We first show that

$$\mathrm{softplus}_\beta(x) = \mathbb{E}_{\epsilon \sim p_\beta} \left[\mathrm{ReLU}(x - \epsilon) \right]\,, \tag{A.22}$$

for a scalar input x. This follows by defining $p(\epsilon)$ implicitly as

$$\mathrm{softplus}_\beta(x) = \int_{-\infty}^{+\infty} p(\epsilon)\, \mathrm{ReLU}(x - \epsilon)\, \mathrm{d}\epsilon\,. \tag{A.23}$$

Differentiating both sides of this equation with respect to x results in

$$\sigma_\beta(x) = \int_{-\infty}^{+\infty} p(\epsilon)\, \Theta(x - \epsilon)\, \mathrm{d}\epsilon = \int_{-\infty}^{x} p(\epsilon)\, \mathrm{d}\epsilon\,, \tag{A.24}$$

where $\Theta(x) = \mathbb{I}(x > 0)$ is the Heaviside step function and $\sigma_\beta(x) = \frac{1}{(1 + e^{-\beta x})}$. Differentiating both sides with respect to x again results in

$$p_\beta(x) = p(x)\,. \tag{A.25}$$

Therefore, (A.22) holds. For a vector input $\vec{x}$, we define the distribution of its perturbation $\vec{\epsilon}$ by

$$p_\beta(\vec{\epsilon}) = \prod_i p_\beta(\epsilon_i)\,, \tag{A.26}$$

where ϵ_i denotes the components of $\vec{\epsilon}$. We will suppress any arrows denoting vector-valued variables in the following in order to ease notation. We choose an orthogonal basis such that

$$\epsilon = \epsilon_p \hat{w} + \sum_i \epsilon_o^{(i)} \hat{w}_o^{(i)} \qquad \text{with} \qquad \hat{w} \cdot \hat{w}_o^{(i)} = 0 \qquad \text{and} \qquad w = \|w\|\,\hat{w}\,. \tag{A.27}$$

This allows us to rewrite

$$\begin{aligned}\mathbb{E}_{\epsilon\sim p_\beta}\left[\mathrm{ReLU}(w^T(x-\epsilon))\right] &= \mathbb{E}_{\epsilon\sim p_\beta}\left[\mathrm{ReLU}(w^T x - \|w\|\,\epsilon_p))\right] \\ &= \int p_\beta(\epsilon_p)\left(\mathrm{ReLU}(w^T x - \|w\|\,\epsilon_p)\right)\mathrm{d}\epsilon_p\end{aligned}$$

By changing the integration variable to $\tilde{\epsilon} = \|w\|\,\epsilon_p$ and using (A.22), we obtain

$$\mathbb{E}_{\tilde{\epsilon}\sim p_\beta}\left[\mathrm{ReLU}(w^T x - \tilde{\epsilon})\right] = \mathrm{softplus}_{\frac{\beta}{\|w\|}}(w^T x)\,, \tag{A.28}$$

The theorem then follows by deriving both sides of the equation with respect to x.

A.2. Details on experiments

We summarize the choices for the maximum number of iterations and the learning rate, used in our quantitative analysis in Table A.1. The α parameter, introduced in Equation 4.1, that determines the weighting between the mean squared error of the explanation maps and the network outputs is set to 10^{-5}.

method	iterations	learning rate
Gradient	1500	10^{-3}
Gradient×Input	1500	10^{-3}
Integrated Gradients	500	5×10^{-3}
Layerwise Relevance Propagation	1500	2×10^{-4}
Guided Back Propagation	1500	10^{-3}
Pattern Attribution	1500	2×10^{-3}

Tab. A.1.: Hyperparameters for VGG16 used in our analysis.

The patterns for the Pattern Attribution explanation method are trained on a subset of the ImageNet training set.

The baseline $\bar{x}$ for Integrated Gradients was set to zero. To approximate the integral, we use 30 steps for which we verified that the attributions approximately adds up to the score at the input.

Besides the VGG16 architecture, we test our method on the AlexNet, Densenet and ResNet architectures. Figure A.1 shows that the similarity measures are comparable for all network architectures for the Gradient method. The hyperparameter choices used in our experiments are summarized in Table A.2. The α parameter is set to 10^{-5}, like before for the VGG16 architecture. For experiments on CIFAR10 we perform 1500 iterations with a learning rate of 2×10^{-4} and $\alpha = 10^{-5}$.

network	iterations	lr
VGG16	1500	10^{-3}
AlexNet	4000	10^{-3}
Densenet121	2000	5×10^{-4}
ResNet18	2000	10^{-3}

Tab. A.2.: Hyperparameters used in our analysis for all networks for the Gradient explanation.

A.2.1. β–growth

In practise, we observe that we get slightly better results by increasing the value of β of the softplus activation $sp(x) = \frac{1}{\beta} \ln(1 + e^{\beta x})$ during the optimization of the adversarial inputs from a start value β_0 to a final value β_e using

$$\beta(t) = \beta_0 \left(\frac{\beta_e}{\beta_0} \right)^{t/T}, \tag{A.29}$$

where t is the current optimization step and T denotes the total number of steps.

We use this incremental adaptation of the β value for all methods except LRP for which we do not find any speed-up in the optimization as the LRP rules do not explicitly depend on the second derivative of the ReLU activations. Figure A.2 demonstrates that for large β values the softplus networks approximate the ReLU network well. Figure A.3 shows this for an example for the Gradient explanation method. We also note that for small β, the Gradient explanation maps become more similar to LRP/GPB/PA explanation maps.

A.3. Three channel attacks

For three channel attacks we aim to accurately reproduce the explanation map prior to taking its absolute values and summing over the color channels. Experiments

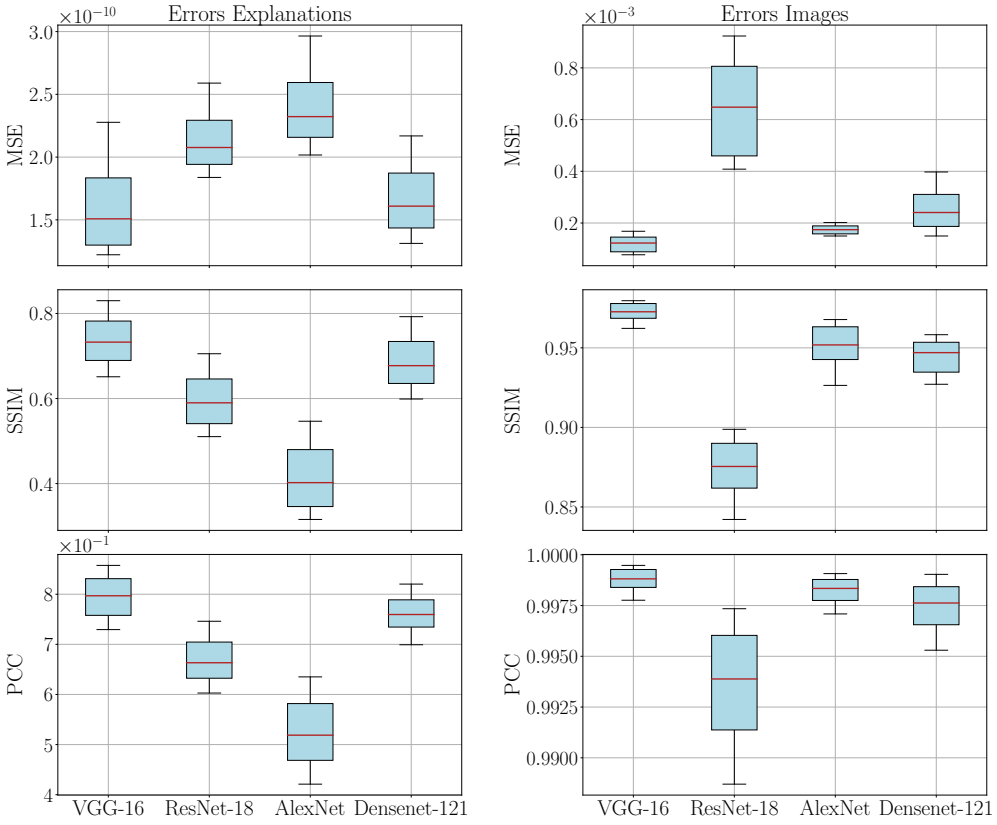


Fig. A.1.: Similarity measures between target map and manipulated map (left) and original image and manipulated image (right) for the Gradient explanation method applied to various architectures.

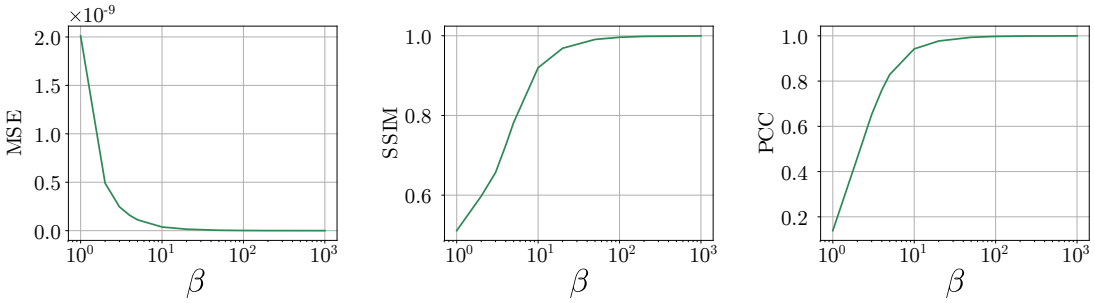


Fig. A.2.: Error measures between the gradient explanation map produced with the original network and explanation maps produced with a network with softplus activation functions using various values for β.

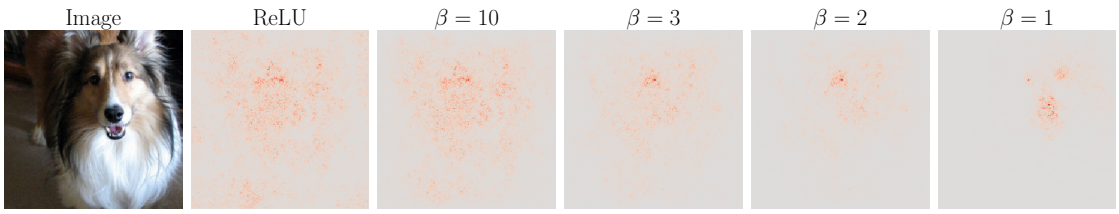


Fig. A.3.: Gradient explanation map produced with the original network and a network with softplus activation functions using various values for β.

show that this is a much harder optimization problem than producing the more commonly used explanation maps. We get mixed results for the different explanation methods. While LRP and Pattern Attribution seem to be vulnerable to three channel attacks, gradient-based explanations seem more robust. For GBP we observe large structured perturbations in the manipulated image. We show some examples in Figure A.4. A thorough hyperparamter search might still improve the effectiveness of three channel attacks. As explanation maps are usually not used as three channel images and, furthermore, structural reproduction of the target explanation seems far more relevant than exact color channel values, the limited effectiveness of our method for three channel explanations does not seem relevant in practice. LRP and Pattern Attribution were not originally defined as three channel explanations. Both methods yield mostly positive relevances in contrast to the other explanation methods. This might be the reason why they are more susceptible to manipulation even when attacking the three channel explanation. For the visualization in Figure A.4 we adapt the normalization for LRP and PA so that the mean value of the explanations roughly matches the mean value for the other explanation methods.

A.4. β-smoothing

One can achieve a smoothing effect when substituting the ReLU activations with softplus activations with small β value and then applying the usual rules for the different explanation methods.

Using softplus activations leads to a reduced curvature of the output manifold. In Figure A.5 we show this for a two layer neural net. When moving along the hypersurface of equal network output the gradient, i.e. the normal vector which is perpendicular to the hypersurface, consequently changes less abruptly than when we use ReLU activations.

A smoothing effect can also be achieved by applying SmoothGrad, see Figure A.6. We average over 10 perturbed images with different values for the standard deviation σ of the Gaussian noise. The noise level ν is related to σ by $\sigma = \nu \cdot (x_{\max} - x_{\min})$, where $x_{\max}$ and $x_{\min}$ are the maximum and minimum values the input image can take.

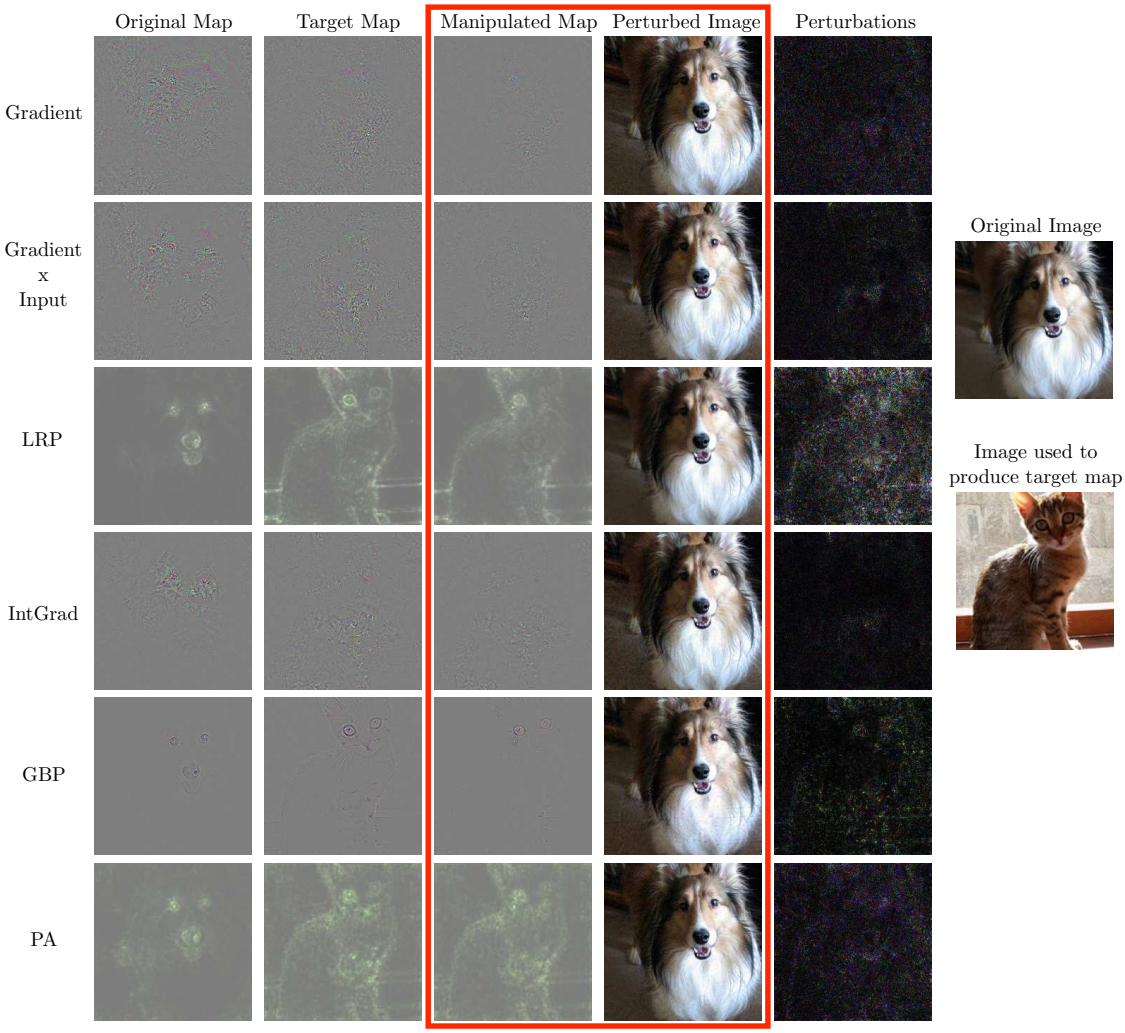


Fig. A.4.: Adversarial attacks on the 3-channel explanations for all considered explanation methods.

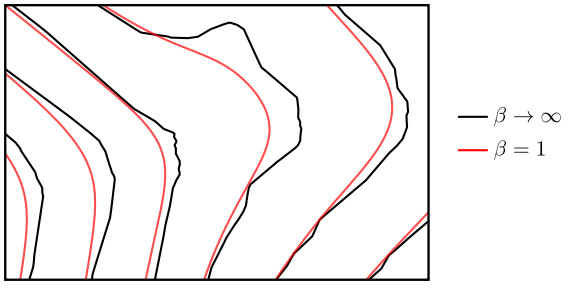


Fig. A.5.: Contour plot of a 2-layer neural network $f(x) = V^\top \text{softplus}(W^\top x)$ with $x \in [-1, 1]^2$, $W \in \mathbb{R}^{2\times 50}$, $V \in \mathbb{R}^{50}$ and $V_i, W_{ij} \sim \mathbb{U}(-1, 1)$. Using a softplus activation with $\beta = 1$ visibly reduces curvature compared to a activation with $\beta \to \infty$, i.e. ReLU activations.

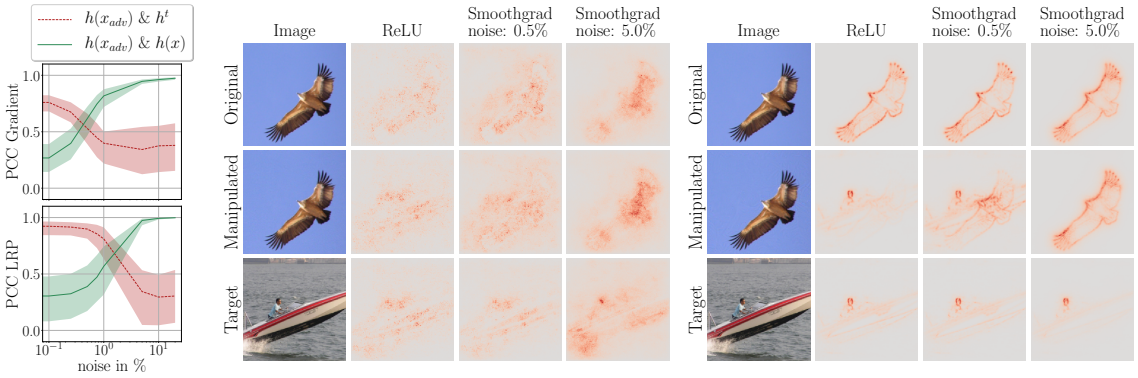


Fig. A.6.: Recovering the original explanation map with SmoothGrad. Left: Recovery is dependent on added noise. Line denotes median, 10^{th} and 90^{th} percentile are shown in semitransparent color. Center and Right: network input and the respective explanation maps for Gradient (center) and LRP (right).

The PCC between the manipulated explanation and the target explanation is lower for the smoothed explanation (see Figure A.7). This indicates that the target map is *not* closely reproduced. For fair comparison we also show the PCC between original and manipulated image, which is lower or equal for the manipulations produced using the smooth explanation methods. The smoothed explanations are therefore more robust than their unsmoothed counterparts when we add perturbations of the same size or even higher. We find equivalent results for SSIM and MSE.

For manipulation of SmoothGrad, we use β growth with $\beta_0 = 10$ and $\beta_e = 100$. For manipulation of β-smoothing, we set $\beta = 0.8$ for all runs. The hyperparameters for SmoothGrad and β-smoothing are summarized in Table A.3. The weighting of the loss terms was set to $\alpha = 10^{-5}$.

meta method	method	iterations	lr
β-smoothing	Gradient	500	2.5×10^{-4}
	Grad x Input	500	2.5×10^{-4}
	IntGrad	200	2.5×10^{-3}
	LRP	1500	2.0×10^{-4}
	GBP	500	5.0×10^{-4}
	PA	500	5.0×10^{-4}
SmoothGrad	Gradient	1500	3×10^{-3}
	LRP	1500	3×10^{-4}

Tab. A.3.: Hyperparameters used in our analysis for β-smoothing and SmoothGrad.

In Figure A.8, we directly compare the original explanation methods with the β-smoothed explanation methods. An increase in robustness can be seen for all methods: explanation maps for β-smoothed explanations have higher MSE and lower SSIM and PCC than explanation maps for the original methods. We always compare the

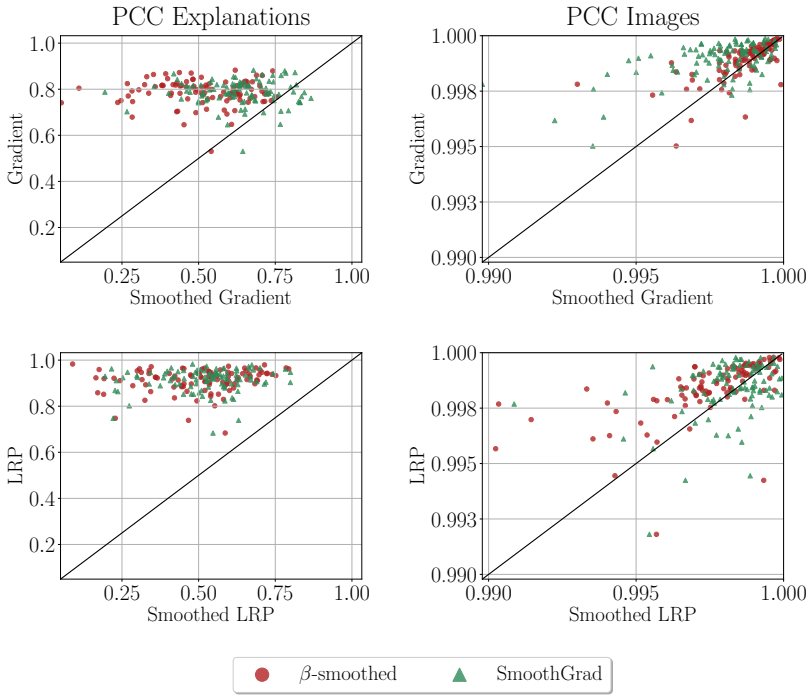


Fig. A.7.: Left: PCC between target map and manipulated map. Right: PCC between original and manipulated image.

target maps with the manipulated maps. We make sure that the similarity measures for the manipulated images are of comparable magnitude.

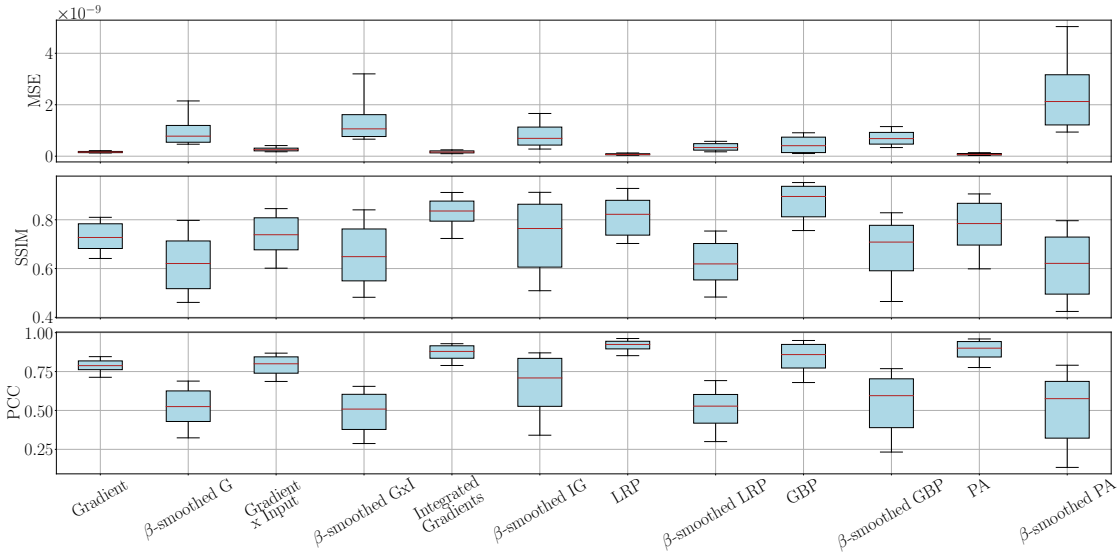


Fig. A.8.: Similarities to target map for the original and the β-smoothed explanation methods.

B. Appendices for Chapter 5

B.1. Proof of Theorem 3

Theorem 3. *Let $f : \mathbb{R}^N \to \mathbb{R}$ be a fully-connected neural network with L layers. The weights of the l-th layer are denoted by $W^{(l)}$ and its activation functions σ are twice-differentiable and bounded*

$$|\sigma'(x)| \leq \Sigma_1\,, \qquad\qquad |\sigma''(x)| \leq \Sigma_2\,. \tag{B.1}$$

The Hessian of the network is then bounded by

$$||H(f)||_F^2 \leq \sum_{m=1}^{L} \left(\prod_{l=1}^{m} ||W^{(l)}||_F^2 \prod_{l=m+1}^{L} ||W^{(l)}||_F \right) \Sigma_1^{L+m-2}\, \Sigma_2\,. \tag{B.2}$$

Proof: The activation at the final neural network layer L is given by

$$a^{(L)}(x) = (\sigma^{(L)} \circ \cdots \circ \sigma^{(1)})(x) \tag{B.3}$$

Its derivative $\partial_k a_i^{(l)}$ is equal to

$$\sum_{s_2 \ldots s_l} W_{is_l}^{(l)} \sigma' \left(\sum_j W_{ij}^{(l)} a_j^{(l-1)} \right) W_{s_l s_{l-1}}^{(l-1)} \sigma' \left(\sum_j W_{s_l j}^{(l-1)} a_j^{(l-2)} \right) \\ \ldots W_{s_2 k}^{(1)} \sigma' \left(\sum_j W_{s_2 j}^{(1)} x_j \right)\,.$$

We therefore obtain

$$\left\| \nabla a^{(l)} \right\|_F \leq (\Sigma_1)^l \prod_{i=1}^{l} \left\| W^{(i)} \right\|_F \tag{B.4}$$

From the expression for $\partial_k a_i^{(l)}$, we can straightforwardly derive that

$$\begin{aligned}\partial_l \partial_k a_i^{(L)} = &\sum_m \sum_{s_2 \ldots s_L} \{ \\ &W_{i s_L}^{(L)} \sigma' \left(\sum_j W_{ij}^{(L)} a_j^{(L-1)} \right) W_{s_L s_{L-1}}^{(L-1)} \sigma' \left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)} \right) \\ &\cdots \sum_p W_{s_{m+1} p}^{(m)} W_{s_{m+1} s_m}^{(m)} \sigma'' \left(\sum_j W_{s_{m+1} j}^{(m)} a_j^{(m-1)}(x) \right) \partial_l a_p^{(m-1)}(x) \\ &\ldots W_{s_2 k}^{(1)} \sigma' \left(\sum_j W_{s_2 j}^{(1)} x_j \right) \} \,.\end{aligned}$$

Restrict to the case for which the index i only takes a single value, i.e. $H_{ij}(f) = \partial_i \partial_j a^L(x)$, we get the bound (5.7) of Theorem 3.

B.2. Relu networks

As was discussed in the main text the bound (5.7) for softplus non-linearities diverges for ReLU non-linearities.

In the following, we will discuss how to generalize the analysis to networks with ReLU activations. We will establish that a distributional generalization of the Hessian can be derived for ReLU networks. A distributional form of the Hessian is sufficient for our purposes because in deriving a bound for the maximal change in explanation we only need to consider the Hessian under an integral, see (5.4). Since integrals over distributions are well-defined, the resulting expression is also well-defined. We will first illustrate this for a simple toy model before considering the general case.

B.2.1. Toy example

Consider the network (depicted in Figure B.1)

$$f(x) = \mathrm{ReLU}(w^{(1)T} x) + \mathrm{ReLU}(w^{(2)T} x) \tag{B.5}$$

with input vector $x \in \mathbb{R}^2$ and weight vectors $w^{(1)} = \frac{1}{\sqrt{2}}[1,1]^T$ and $w^{(2)} = \frac{1}{\sqrt{2}}[1,-1]^T$. The first derivative with respect to x_j is

$$\partial_j f(x) = w_j^{(1)} \theta(w^{(1)T} x) + w_j^{(2)} \theta(w^{(2)T} x) \,, \tag{B.6}$$

where we have defined the Heaviside step function

$$\theta(x) = \begin{cases} 1 & x \geq 0 \,, \\ 0 & x < 0 \,. \end{cases} \tag{B.7}$$

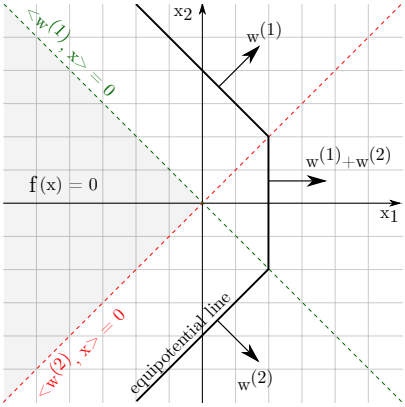


Fig. B.1.: Toy function. Lines of rootpoints are marked in green and red. The grey area shows where $f(x) = 0$

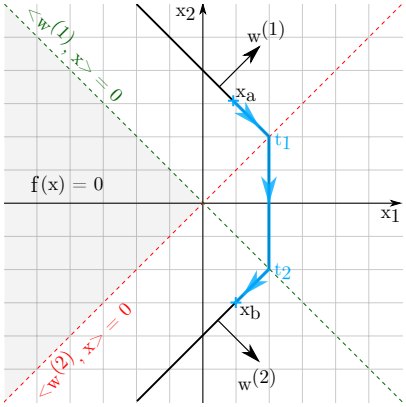


Fig. B.2.: Path (in blue) along an equipotential line (constant network output $f(x) = \text{const}$)

We note that the derivative of the step function is not well-defined at zero. However, a distributional generalization thereof can be defined, i.e.

$$\theta'(x) = \delta(x)\,, \tag{B.8}$$

where δ denotes the Dirac delta distribution.

With this definition, the ij-th entry of (the distributional generalization of) the Hessian matrix can formally be written as

$$\partial_i \partial_j f(x) = w_i^{(1)} w_j^{(1)} \delta(w^{(1)T} x) + w_i^{(2)} w_j^{(2)} \delta(w^{(2)T} x)\,. \tag{B.9}$$

By (5.4), the change in (Gradient) explanation when moving from point x to x_{adv} is then given by

$$(h(x) - h(x_{\text{adv}}))_j = \int_{-\infty}^{\infty} \sum_i \left(w_i^{(1)} w_j^{(1)} \delta(w^{(1)T} x) + w_i^{(2)} w_j^{(2)} \delta(w^{(2)T} x) \right) \dot{x}_i \, \mathrm{d}t\,, \tag{B.10}$$

where we have used the notation $x(t)$ for the curve connecting the unperturbed and perturbed data points.

For integrating over the delta distribution in composition with a (scalar-valued) function, we use

$$\int_{-\infty}^{\infty} f(t)\delta(y(t))\mathrm{d}t = \sum_{t_N} \frac{f(t_N)}{|y'(t_N)|} \tag{B.11}$$

with t_N being the roots of $y(t)$. Using this expression, we then obtain the following

change in saliency map

$$\begin{aligned}(h(x) - h(x_{\text{adv}}))_j &= \sum_{t_N} \frac{\sum_i w_i^{(1)} w_j^{(1)} \dot{x}_i}{\left|\sum_i w_i^{(1)} \dot{x}_i\right|} + \sum_{t_N} \frac{\sum_i w_i^{(2)} w_j^{(2)} \dot{x}_i}{\left|\sum_i w_i^{(2)} \dot{x}_i\right|} \\ &= \sum_{t_N} \operatorname{sgn}\left(\sum_i w_i^{(1)} \dot{x}_i\right) w_j^{(1)} + \sum_{t_N} \operatorname{sgn}\left(\sum_i w_i^{(2)} \dot{x}_i\right) w_j^{(2)}.\end{aligned}$$

Consider the blue path in Figure B.2 whose root points are denoted by t_1 and t_2. We note that these root points correspond to kinks in the curve $x(t)$ connecting the unperturbed and perturbed data point. Their corresponding normalized velocity vectors are given by $\dot{x}(t_1) = w^{(2)}$ and $\dot{x}(t_2) = (0, -1)^T$ respectively. We therefore obtain

$$\begin{aligned}h(x) - h(x_{\text{adv}}) &= \operatorname{sgn}\left(\langle w^{(1)}, \dot{x}(t_2)\rangle\right) w^{(1)} + \operatorname{sgn}\left(\langle w^{(2)}, \dot{x}(t_1)\rangle\right) w^{(2)} \\ &= \operatorname{sgn}\left(-\frac{1}{\sqrt{2}}\right) w^{(1)} + \operatorname{sgn}(1)\, w^{(2)} \\ &= w^{(2)} - w^{(1)}\end{aligned}$$

which is correct as $h(x) = w^{(2)}$ and $h(x_{\text{adv}}) = w^{(1)}$. It is important to stress that we have obtained this result despite the fact that the Hessian of the neural network g is only given in generalized distributional form.

B.2.2. General case

The argument of the previous section can be generalized to arbitrary fully-connected networks with weights W^l of layer $l \in \{1, \dots, L\}$. The general logic follows closely the toy model discussed in the previous section, i.e. a distributional generalization of the Hessian is derived and since on the right-hand-side of (5.4) the Hessian only appears under an integral, this distributional form is sufficient to obtain a bound on the maximal change in explanation due to a perturbation of the input. Using this technique, we derive the following theorem:

Theorem 4. *Let x and $x_{adv} = x + \delta x$ denote the unperturbed and perturbed data points respectively. We denote by $x(t)$ the curve connecting the unperturbed and perturbed points, i.e. $x(t = -\infty) = x$ and $x(t = +\infty) = x_{adv}$. Furthermore, we assume that all points on the curve have the same network output, i.e. $f(x(t_1)) = f(x(t_2))$ for all $t_1, t_2 \in \mathbb{R}$. The maximal change of explanation is then given by*

$$||h(x) - h(x_{adv})||^2 \leq \sum_{kinks(x(t))} \left(\prod_{l=1}^{L} \left\|W^{(l)}\right\|_F^2\right), \tag{B.12}$$

where the sum runs over all kinks of the curve $x(t)$.

We can give an intuition for the theorem by considering the blue curve of Figure B.2 for the toy model of the previous section. In this case, the sum over the kinks would run over $x(t_1)$ and $x(t_2)$, see Figure B.2. Only at these kinks, the gradient of the network will change. In the theorem, we then estimate this change by its maximal value, i.e. the change is equal to the product of all weights.

As a practical consequence of the theorem, we can make explanations more robust by weight decay also in the case of ReLU non-linearities.

Proof: Let $W^{(l)}$ be the weights of layer l. We denote the l-th layer by $\mathrm{ReLU}^{(l)}(x) = \mathrm{ReLU}(W^{(l)}x)$. It then follows that

$$\partial_k \,\mathrm{ReLU}\left(\sum_j W_{ij}x_j\right) = W_{ik}\,\theta\left(\sum_j W_{ij}x_j\right) \tag{B.13}$$

$$\partial_l\,\theta\left(\sum_j W_{ij}x_j\right) = W_{il}\,\delta\left(\sum_j W_{ij}x_j\right) \tag{B.14}$$

where θ and δ are the Heaviside step function and the delta distribution respectively. The activation at layer L is then given by

$$a^{(L)}(x) = (\mathrm{ReLU}^{(L)} \circ \cdots \circ \mathrm{ReLU}^{(1)})(x) \tag{B.15}$$

Its derivative $\partial_k a_i^{(L)}$ is equal to

$$\sum_{s_2 \ldots s_L} W^{(L)}_{is_L}\theta\left(\sum_j W^{(L)}_{ij}a_j^{(L-1)}\right) W^{(L-1)}_{s_L s_{L-1}}\theta\left(\sum_j W^{(L-1)}_{s_L j}a_j^{(L-2)}\right) \ldots W^{(1)}_{s_2 k}\theta\left(\sum_j W^{(1)}_{s_2 j}x_j\right)$$

Deriving this expression for $\partial_k a_i^{(L)}$ again, we obtain

$$\begin{aligned}\partial_l\partial_k a_i^{(L)} = &\sum_m \sum_{s_2 \ldots s_L} \{ \\ &W^{(L)}_{is_L}\theta\left(\sum_j W^{(L)}_{ij}a_j^{(L-1)}\right) W^{(L-1)}_{s_L s_{L-1}}\theta\left(\sum_j W^{(L-1)}_{s_L j}a_j^{(L-2)}\right) \\ &\cdots \sum_p W^{(m)}_{s_{m+1}p}W^{(m)}_{s_{m+1}s_m}\delta\left(\sum_j W^{(m)}_{s_{m+1}j}a_j^{(m-1)}(x)\right)\partial_l a_p^{(m-1)}(x) \\ &\ldots W^{(1)}_{s_2 k}\theta\left(\sum_j W^{(1)}_{s_2 j}x_j\right)\}\end{aligned}$$

We now restrict to the case that $a^{(L)}$ has only a single output value. As a result, the index i in the expression above only takes one value, i.e. $i = 1$. We define $f(x) = a_1^{(L)}(x)$ to ease notation. We then substitute this expression for $\partial_l \partial_k f = \partial_l \partial_k a_1^{(L)}$ in (5.4) and obtain

$$\begin{aligned}(h(x) - h(x_{\text{adv}}))_k = \sum_m \sum_{s_2 \dots s_L} \int_{-\infty}^{\infty} \mathrm{d}t \Big\{ \\ W_{1 s_L}^{(L)} \theta \left(\sum_j W_{ij}^{(L)} a_j^{(L-1)} \right) W_{s_L s_{L-1}}^{(L-1)} \theta \left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)} \right) \\ \dots \sum_{\hat{s}_m} W_{s_{m+1} \hat{s}_m}^{(m)} W_{s_{m+1} s_m}^{(m)} \delta \left(\sum_j W_{s_{m+1} j}^{(m)} a_j^{(m-1)}(x) \right) \dot{a}_{\hat{s}_m}^{(m-1)}(x) \\ \dots W_{s_2 k}^{(1)} \theta \left(\sum_j W_{s_2 j}^{(1)} x_j \right) \Big\},\end{aligned}$$

where we have used the notation $\partial_t a^{(m-1)} = \dot{a}^{(m-1)}$ for notational simplicity. Using the identity (B.11), we then obtain

$$\begin{aligned}(h(x) - h(x_{\text{adv}}))_k = \sum_m \sum_{x_N^m} \sum_{s_2 \dots s_L} \Big\{ \\ W_{1 s_L}^{(L)} \theta \left(\sum_j W_{ij}^{(L)} a_j^{(L-1)} \right) W_{s_L s_{L-1}}^{(L-1)} \theta \left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)} \right) \\ \dots W_{s_{m+1} s_m}^{(m)} \operatorname{sgn} \left(\sum_j W_{s_{m+1} j}^{(m)} \dot{a}_j^{(m-1)}(x_N^m) \right) \\ \dots W_{s_2 k}^{(1)} \theta \left(\sum_j W_{s_2 j}^{(1)} (x_N^m)_j \right) \Big\},\end{aligned}$$

where the sum over x_N^m runs over all zeropoints of $\sum_j W_{s_{m+1} j}^{(m)} a^{(m-1)}$ along the trajectory connecting x with x_{adv}. Using the fact that $|\theta(\bullet)| \leq 1$ and $|\operatorname{sgn}(\bullet)| \leq 1$, we obtain

$$||h(x) - h(x_{\text{adv}})|\,|^2 \leq \sum_m \sum_{x_N^m} \left\| W^{(L)} \right\|_F^2 \left\| W^{(L-1)} \right\|_F^2 \dots \left\| W^{(m)} \right\|_F^2 \dots \left\| W^{(1)} \right\|_F^2 . \tag{B.16}$$

As in the case of the toy model, the summands run over all kinks of the trajectory. This bound for ReLU networks depends purely on the network weights and the number of kinks passed when moving from x to x_{adv}. If we reduce the Frobenius norms of the weights, we also reduce the maximal possible change in explanation.

B.3. Interchangeability of softplus β

When training softplus networks with different β values it is interesting to consider how they differ as the beta values partially cancel out or can be absorbed into the weights and biases.

The softplus function is defined as:

$$\mathrm{sp}_\beta(x) = \frac{1}{\beta} \ln\left(1 + \mathrm{e}^{\beta x}\right) \tag{B.17}$$

Therefore, we can relate two softplus functions with different β values, β_1 and β_2, as follows:

$$\mathrm{sp}_{\beta_1}(x) = \frac{\beta_2}{\beta_1} \mathrm{sp}_{\beta_2}(\frac{\beta_1}{\beta_2} x) \tag{B.18}$$

A network consisting of linear layers and softplus activations with $\beta = \beta_1$ has weights $W^{(i)}$ and biases $b^{(i)}$. We can define a network with the same structure but a different softplus $\beta = \beta_2$ and weights $\tilde{W}^{(i)}$ and biases $\tilde{b}^{(i)}$. The networks give identical outputs for all inputs if we define the weights and biases of the second network in the following way:

$$\begin{aligned}
\tilde{W}^{(1)} &= \frac{\beta_1}{\beta_2} W^{(1)} \\
\tilde{W}^{(i)} &= W^{(i)}, \quad \forall i : 1 < i < n \\
\tilde{W}^{(n)} &= \frac{\beta_2}{\beta_1} W^{(n)} \\
\tilde{b}^{(i)} &= \frac{\beta_1}{\beta_2} b^{(i)}, \quad \forall i : i < n \\
\tilde{b}^{(n)} &= b^{(n)}
\end{aligned}$$

However, this mapping is not learned when training networks with different β values from scratch as the distribution over weight norms stays very similar while the distribution changes drastically when artificially changing the β value as demonstrated above. We show this effect for a few examples in Section B.3.1.

B.3.1. Examples

Artificially constructing networks, as explained above, leads to a larger variance in the weight norms. Even when no weight decay is used during training, the weight norms of the different network layers in one network tend to vary within one order of magnitude.

Figure B.3 shows the weights and biases for two networks from Table B.1 with $\beta = 1$ and $\beta = 10$ and the respective weights and biases for two networks that produce identical outputs but have changed β values. In both cases the average weight norm of the constructed network is higher than of the original.

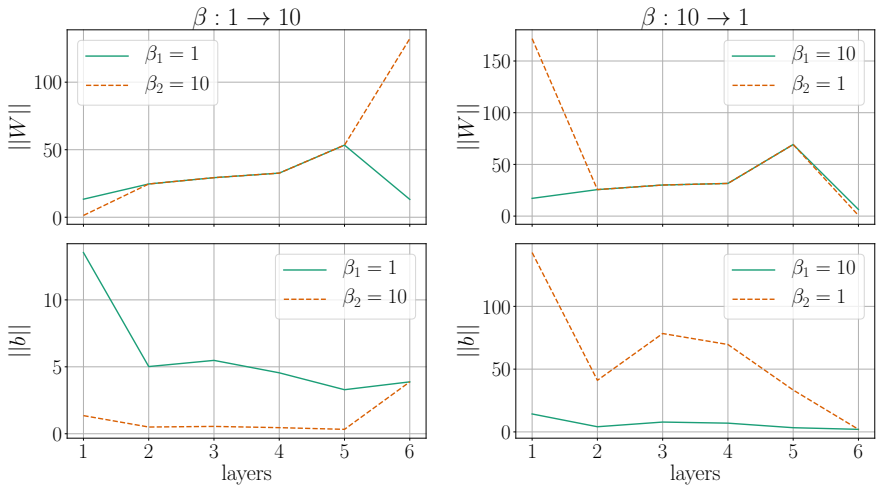


Fig. B.3.: Weights and biases for networks with identical outputs but different β value for the softplus activation. Left: β was changed from 1 to 10. Right: β was changed from 10 to 1.

Figure B.4 shows weights and biases for some of our networks from Table B.1. The weight and bias norms for each network are normalized with the respective maximum value over all layers. Without exception the highest weight norm is found in layer 5 in contrast to the maximum weight norm when we do the artificial β value switch. Thus training softplus networks from scratch does produce fundamentally different networks, that cannot be obtained by a mere rescaling of weights and biases.

B.4. Experimental analysis

B.4.1. Network structure

The structure of all simple CNNs, trained within the scope of Chapter 5, is depicted in Figure B.5. The activation function is either ReLU or softplus (for the networks trained with β-smoothing or Hessian minimization). In order to focus on the robustness we aimed to train the different networks to similar accuracy (albeit no longer than 200 epochs). We use Stochastic Gradient Descent with momentum and learning rate decay. We do not perform any further hyperparameter optimization. Statistics for all trained networks are summarized in Table B.1.

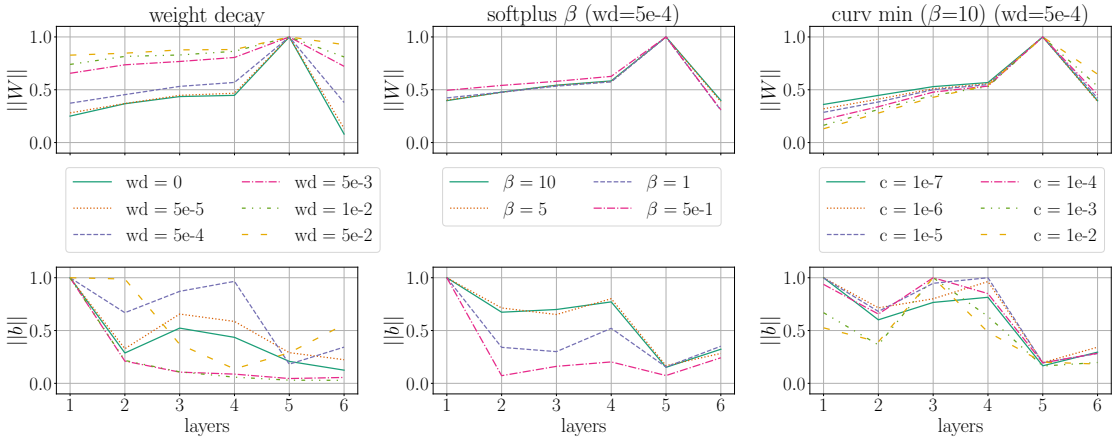


Fig. B.4.: Weights and biases plotted over layers for different networks (left column: networks with weight decay, middle column: networks with different β values, right column: networks trained with curvature minimization)

λ	β	ζ	accuracy	PCC for different ν			$\|\|W\|\|$	$\|\|H\|\|$
				0.005	0.01	0.025		
0	ReLU	0.0	85.75	0.73	0.57	0.32	30.79	-
5e-5	ReLU	0.0	86.38	0.77	0.62	0.36	23.28	-
5e-4	ReLU	0.0	88.63	0.84	0.70	0.45	11.37	-
5e-3	ReLU	0.0	86.10	0.89	0.80	0.62	4.80	-
1e-2	ReLU	0.0	81.41	0.89	0.80	0.64	3.61	-
0	10	0.0	85.61	0.81	0.63	0.34	30.01	503.61
0	5	0.0	85.60	0.88	0.73	0.39	28.70	280.71
0	1	0.0	85.60	0.93	0.85	0.61	27.76	59.21
0	5e-1	0.0	84.51	0.94	0.88	0.67	28.84	39.06
5e-5	10	0.0	86.36	0.86	0.70	0.39	22.91	298.79
5e-5	5	0.0	86.33	0.91	0.78	0.46	22.97	154.94
5e-5	1	0.0	86.03	0.94	0.86	0.62	22.73	51.63
5e-5	5e-1	0.0	85.34	0.94	0.88	0.67	23.76	36.43
5e-4	10	0.0	88.84	0.88	0.75	0.49	11.24	91.68
5e-4	5	0.0	88.76	0.91	0.79	0.53	11.36	59.57
5e-4	1	0.0	86.80	0.93	0.86	0.63	11.93	28.45
5e-4	5e-1	0.0	85.36	0.93	0.86	0.67	10.62	16.78
5e-3	10	0.0	86.13	0.91	0.82	0.64	4.81	9.54
5e-3	5	0.0	85.44	0.91	0.83	0.64	4.76	7.49
5e-3	1	0.0	83.35	0.92	0.86	0.71	4.86	3.79
5e-3	5e-1	0.0	77.60	0.96	0.93	0.85	4.66	2.02
1e-2	10	0.0	80.44	0.90	0.82	0.66	3.48	5.42
1e-2	5	0.0	77.57	0.89	0.82	0.65	3.33	5.27
1e-2	1	0.0	71.74	0.97	0.94	0.86	3.21	1.26
1e-2	5e-1	0.0	72.03	0.98	0.95	0.89	3.35	0.97

λ	β	ζ	accuracy	PCC for different ν			$\|\|W\|\|$	$\|\|H\|\|$
				0.005	0.01	0.025		
0	10	1e-7	85.65	0.95	0.87	0.60	24.72	56.77
0	10	1e-6	85.74	0.97	0.92	0.73	22.45	21.47
0	10	1e-5	85.56	0.98	0.95	0.84	20.59	8.49
0	10	1e-4	84.12	0.99	0.97	0.90	19.14	3.23
0	10	1e-3	82.40	0.99	0.98	0.94	17.91	1.19
0	10	1e-2	80.07	0.99	0.98	0.94	17.08	0.43
0	5	1e-7	86.26	0.95	0.88	0.64	25.44	49.58
0	5	1e-6	85.94	0.97	0.92	0.74	23.41	20.85
0	5	1e-5	85.87	0.98	0.95	0.83	21.88	7.91
0	5	1e-4	84.81	0.98	0.97	0.90	20.74	2.96
0	5	1e-3	83.12	0.99	0.98	0.93	19.72	1.29
0	5	1e-2	80.95	0.99	0.98	0.94	19.02	0.41
0	1	1e-7	85.24	0.94	0.88	0.69	27.69	31.61
0	1	1e-6	85.17	0.95	0.90	0.75	26.29	17.67
0	1	1e-5	84.85	0.97	0.94	0.83	25.11	7.62
0	1	1e-4	84.70	0.98	0.95	0.87	24.25	3.03
0	1	1e-3	82.68	0.98	0.96	0.90	23.23	1.16
0	1	1e-2	81.57	0.98	0.95	0.89	22.20	0.42
0	5e-1	1e-7	81.90	0.95	0.90	0.77	12.05	9.28
0	5e-1	1e-6	85.46	0.96	0.91	0.77	28.07	15.20
0	5e-1	1e-5	84.41	0.97	0.93	0.82	26.81	6.79
0	5e-1	1e-4	84.08	0.98	0.96	0.89	25.59	2.86
0	5e-1	1e-3	82.84	0.98	0.96	0.89	22.21	1.12
0	5e-1	1e-2	81.04	0.98	0.96	0.90	23.36	0.41
5e-5	10	1e-7	86.68	0.95	0.87	0.62	20.24	52.22
5e-5	10	1e-6	86.47	0.97	0.92	0.74	18.75	21.36
5e-5	10	1e-5	85.87	0.98	0.95	0.84	17.28	8.18
5e-5	10	1e-4	84.55	0.99	0.97	0.90	16.05	3.29
5e-5	10	1e-3	83.03	0.99	0.98	0.93	15.01	1.14
5e-5	10	1e-2	80.47	0.99	0.98	0.94	13.88	0.43
5e-5	5	1e-7	86.76	0.95	0.87	0.62	21.05	45.57
5e-5	5	1e-6	86.41	0.97	0.92	0.74	19.57	19.16
5e-5	5	1e-5	86.16	0.98	0.95	0.84	18.29	8.00
5e-5	5	1e-4	85.21	0.99	0.97	0.90	17.20	3.24
5e-5	5	1e-3	82.69	0.99	0.98	0.93	16.41	1.17
5e-5	5	1e-2	80.91	0.99	0.97	0.93	15.35	0.45
5e-5	1	1e-7	85.78	0.95	0.89	0.70	22.42	29.93
5e-5	1	1e-6	86.31	0.96	0.92	0.77	21.76	16.98
5e-5	1	1e-5	85.54	0.97	0.93	0.82	20.53	7.38
5e-5	1	1e-4	84.62	0.98	0.95	0.88	18.94	2.99
5e-5	1	1e-3	83.82	0.98	0.97	0.91	18.00	1.17
5e-5	1	1e-2	80.49	0.98	0.97	0.92	17.10	0.43

λ	β	ζ	accuracy	PCC for different ν			$\|\|W\|\|$	$\|\|H\|\|$
				0.005	0.01	0.025		
5e-5	5e-1	1e-7	84.84	0.95	0.89	0.70	23.02	26.39
5e-5	5e-1	1e-6	85.21	0.96	0.91	0.75	20.38	16.81
5e-5	5e-1	1e-5	82.98	0.96	0.92	0.81	12.51	6.07
5e-5	5e-1	1e-4	84.29	0.97	0.95	0.87	15.54	3.02
5e-5	5e-1	1e-3	82.67	0.98	0.96	0.89	14.95	1.12
5e-5	5e-1	1e-2	80.35	0.98	0.96	0.89	11.56	0.42
5e-4	10	1e-7	88.96	0.94	0.87	0.62	10.95	35.47
5e-4	10	1e-6	88.24	0.96	0.91	0.73	10.76	17.68
5e-4	10	1e-5	87.61	0.98	0.94	0.83	9.98	7.27
5e-4	10	1e-4	86.54	0.98	0.96	0.88	9.20	3.08
5e-4	10	1e-3	84.80	0.98	0.96	0.90	8.42	1.10
5e-4	10	1e-2	82.72	0.98	0.96	0.90	7.67	0.41
5e-4	5	1e-7	88.68	0.94	0.87	0.63	11.07	32.80
5e-4	5	1e-6	88.31	0.96	0.91	0.73	10.78	16.68
5e-4	5	1e-5	87.75	0.97	0.94	0.83	10.03	6.97
5e-4	5	1e-4	86.35	0.98	0.96	0.88	9.70	2.96
5e-4	5	1e-3	84.74	0.98	0.96	0.91	8.97	1.16
5e-4	5	1e-2	82.12	0.98	0.96	0.91	8.04	0.44
5e-4	1	1e-7	87.11	0.94	0.86	0.64	11.67	25.09
5e-4	1	1e-6	87.08	0.96	0.91	0.75	11.34	14.06
5e-4	1	1e-5	86.72	0.97	0.94	0.82	11.13	7.31
5e-4	1	1e-4	85.64	0.98	0.96	0.88	9.81	2.92
5e-4	1	1e-3	83.48	0.98	0.97	0.91	9.49	1.20
5e-4	1	1e-2	81.87	0.98	0.96	0.91	8.11	0.44
5e-4	5e-1	1e-7	78.10	0.93	0.87	0.73	7.07	4.57
5e-4	5e-1	1e-6	85.76	0.95	0.90	0.75	11.03	12.23
5e-4	5e-1	1e-5	85.56	0.97	0.93	0.82	10.82	6.20
5e-4	5e-1	1e-4	85.21	0.98	0.96	0.88	10.21	2.86
5e-4	5e-1	1e-3	81.05	0.96	0.93	0.84	6.99	0.94
5e-4	5e-1	1e-2	81.98	0.98	0.97	0.91	8.12	0.46
5e-3	10	1e-7	86.09	0.91	0.83	0.65	4.79	8.23
5e-3	10	1e-6	85.92	0.91	0.83	0.66	4.74	6.31
5e-3	10	1e-5	85.59	0.93	0.87	0.72	4.66	3.73
5e-3	10	1e-4	84.53	0.95	0.91	0.79	4.50	1.88
5e-3	10	1e-3	83.11	0.96	0.93	0.84	4.30	0.86
5e-3	10	1e-2	80.16	0.97	0.95	0.88	4.00	0.33
5e-3	5	1e-7	85.90	0.90	0.82	0.64	4.78	7.09
5e-3	5	1e-6	85.43	0.91	0.84	0.67	4.79	5.57
5e-3	5	1e-5	85.26	0.92	0.86	0.71	4.67	3.48
5e-3	5	1e-4	84.51	0.95	0.90	0.78	4.52	1.76
5e-3	5	1e-3	83.07	0.96	0.93	0.84	4.32	0.82
5e-3	5	1e-2	80.53	0.97	0.94	0.87	4.06	0.32

λ	β	ζ	accuracy	PCC for different ν			$\|\|W\|\|$	$\|\|H\|\|$
				0.005	0.01	0.025		
5e-3	1	1e-7	82.84	0.91	0.85	0.70	4.76	4.11
5e-3	1	1e-6	83.40	0.92	0.86	0.72	4.85	3.37
5e-3	1	1e-5	81.72	0.93	0.88	0.75	4.65	2.62
5e-3	1	1e-4	82.19	0.95	0.91	0.81	4.68	1.52
5e-3	1	1e-3	81.14	0.96	0.93	0.86	4.43	0.74
5e-3	1	1e-2	79.13	0.96	0.94	0.86	4.16	0.32
5e-3	5e-1	1e-7	77.41	0.96	0.93	0.85	4.59	2.03
5e-3	5e-1	1e-6	77.42	0.97	0.94	0.85	4.57	1.91
5e-3	5e-1	1e-5	76.88	0.97	0.94	0.86	4.47	1.57
5e-3	5e-1	1e-4	77.17	0.97	0.94	0.86	4.49	1.24
5e-3	5e-1	1e-3	76.68	0.97	0.95	0.89	4.29	0.64
5e-3	5e-1	1e-2	75.01	0.98	0.97	0.92	3.89	0.27
1e-2	10	1e-7	64.43	0.94	0.87	0.65	3.37	7.11
1e-2	10	1e-6	79.77	0.90	0.83	0.66	3.46	4.06
1e-2	10	1e-5	79.87	0.91	0.84	0.69	3.46	2.47
1e-2	10	1e-4	78.86	0.94	0.88	0.75	3.37	1.34
1e-2	10	1e-3	77.68	0.95	0.92	0.82	3.25	0.63
1e-2	10	1e-2	75.75	0.97	0.94	0.87	3.12	0.25
1e-2	5	1e-7	78.63	0.89	0.81	0.65	3.40	4.42
1e-2	5	1e-6	77.74	0.90	0.83	0.66	3.34	4.21
1e-2	5	1e-5	78.16	0.90	0.83	0.68	3.35	2.34
1e-2	5	1e-4	78.18	0.92	0.86	0.73	3.31	1.20
1e-2	5	1e-3	77.43	0.96	0.92	0.82	3.25	0.59
1e-2	5	1e-2	74.45	0.96	0.93	0.85	3.02	0.25
1e-2	1	1e-7	71.74	0.97	0.94	0.86	3.21	1.26
1e-2	1	1e-6	71.92	0.97	0.95	0.88	3.24	1.18
1e-2	1	1e-5	73.02	0.97	0.94	0.86	3.30	0.98
1e-2	1	1e-4	72.02	0.97	0.95	0.88	3.16	0.77
1e-2	1	1e-3	71.16	0.98	0.95	0.89	3.08	0.43
1e-2	1	1e-2	69.64	0.98	0.97	0.92	2.90	0.19
1e-2	5e-1	1e-7	70.53	0.98	0.96	0.90	3.20	0.87
1e-2	5e-1	1e-6	70.07	0.98	0.96	0.90	3.21	0.89
1e-2	5e-1	1e-5	70.46	0.98	0.96	0.89	3.20	0.79
1e-2	5e-1	1e-4	71.49	0.98	0.96	0.91	3.31	0.72
1e-2	5e-1	1e-3	70.06	0.99	0.97	0.93	3.11	0.41
1e-2	5e-1	1e-2	67.76	0.99	0.98	0.94	2.84	0.19

Tab. B.1.: Statistics of all network configurations. Columns show weight decay (λ), activation function (ReLU or β parameter for softplus), parameter for curvature minimization ζ, test accuracy (acc), mean Pearson correlation coefficient (pcc) for Gaussian noise with different noise levels ν, average weight norm ($\|W\|$) and average approximated Hessian norm ($\|H\|$).

```
CNN_CIFAR(
  (features): Sequential(
    (conv0): Conv2d(3, 32, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
    (acti0): ActivationFunction()
    (conv1): Conv2d(32, 32, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
    (acti1): ActivationFunction()
    (pool2): MaxPool2d(kernel_size=2, stride=2,
                    padding=0, dilation=1,
                    ceil_mode=False)
    (conv3): Conv2d(32, 64, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
    (acti3): ActivationFunction()
    (conv4): Conv2d(64, 64, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
    (acti4): ActivationFunction()
    (pool5): MaxPool2d(kernel_size=2, stride=2,
                    padding=0, dilation=1,
                    ceil_mode=False)
  )
  (classifier): Sequential(
    (view0): Reshape()
    (dens0): Linear(in_features=4096,
                    out_features=256,
                    bias=True)
    (acti0): ActivationFunction()
    (dens1): Linear(in_features=256,
                    out_features=10,
                    bias=True)
  )
)
```

Fig. B.5.: Setup of simple CNN for CIFAR10

B.4.2. Gradient explanation

In Figures B.6, B.7, and B.8, we show additional error measures for the Gradient explanation. PCC and SSIM increase with robustness while MSE decreases.

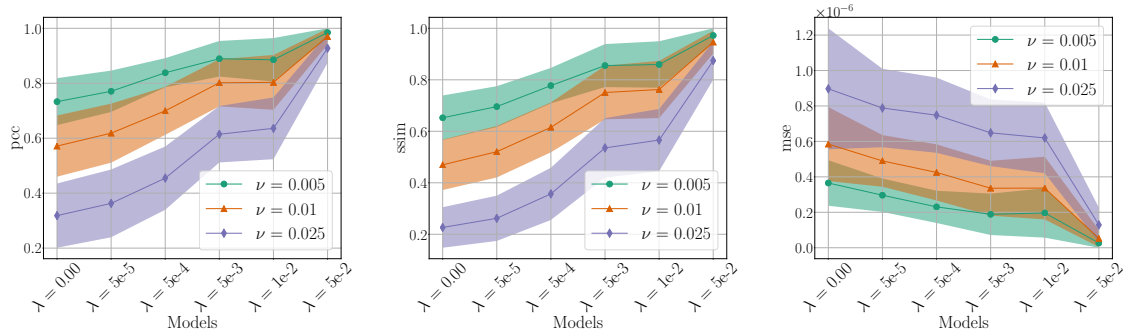


Fig. B.6.: PCC, SSIM and MSE between original Gradient explanation map and explanation after adding random noise to the image. PCC and SSIM are higher and MSE is lower for networks trained with weight decay λ. That means weight decay improves robustness of explanations. We show mean +/- std.

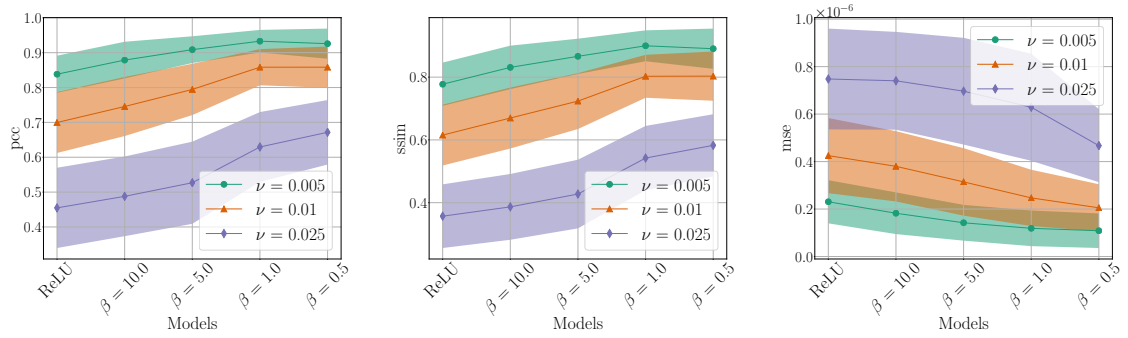


Fig. B.7.: PCC, SSIM and MSE between original Gradient explanation map and explanation after adding random noise to the image. PCC and SSIM are higher and MSE is lower for softplus networks trained with a small β value. That means softplus activations improve robustness of explanations. All nets were trained with weight decay (λ=5e-4). We show mean +/- std.

B.4.3. Other explanation methods

In Figures B.9, B.10, B.11, and B.12, we show how our proposed methods effect other explanation methods. The trend towards increased robustness is clearly visible for all considered explanation methods. We note that the explanations start from different levels of robustness but can still profit from our methods. The most resilient method against random input perturbations is Layerwise Relevance Propagation, followed by Guided Backpropagation, Integrated Gradients, Gradient×Input and Gradient in descending order.

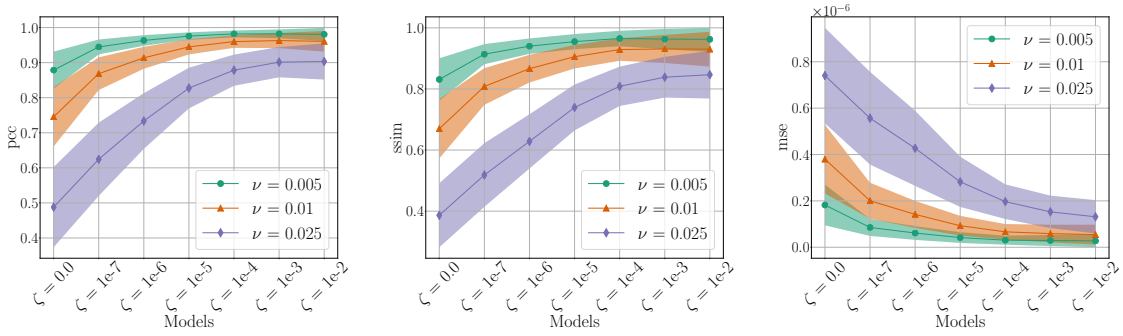


Fig. B.8.: PCC, SSIM and MSE between original Gradient explanation map and explanation after adding random noise to the image. PCC and SSIM are higher and MSE is lower for networks trained with strong curvature minimization. That means minimizing the curvature improves robustness of explanations. All nets were trained with softplus activations ($\beta = 10$) and weight decay (λ=5e-4). We show mean +/- std.

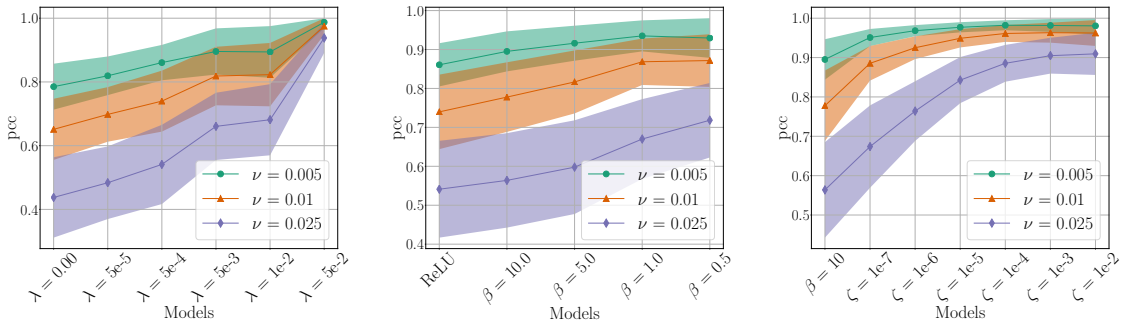


Fig. B.9.: PCCs (mean +/- std) between original Gradient×Input explanation map and explanation after adding random noise to the image. left: effect of weight decay λ, middle: effect of softplus β (for λ = 5e-4), right: effect of curvature minimization (for λ = 5e-4, β=10).

B.4.4. Other types of noise

In the main text we only consider Gaussian noise. We repeat our experiments from 5.2 for the Gradient explanation when we perturb the input images with Laplacian noise and salt-pepper noise.

B.4.4.1. Laplace noise

We sample random noise from the Laplace distribution

$$p(x|\mu, b) = \frac{1}{2b} \exp -\frac{|x - \mu|}{b} \tag{B.19}$$

where μ is the data mean and $b = (x_{\max} - x_{\min})\nu$ is a scale parameter which depends

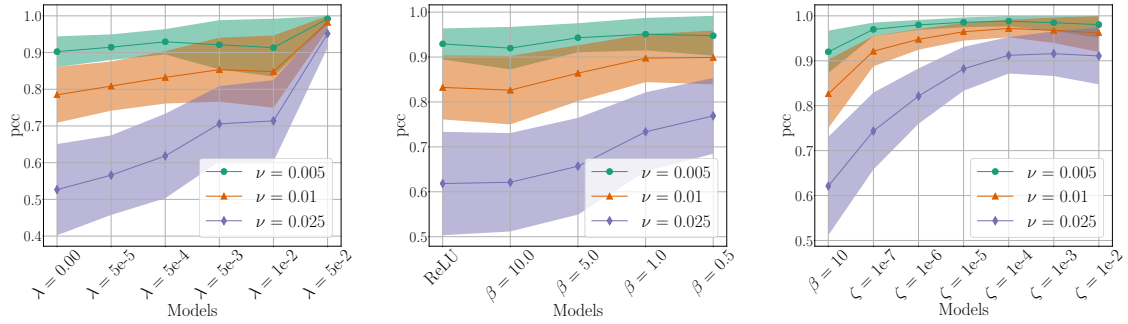


Fig. B.10.: PCCs (mean +/- std) between original Integrated Gradients explanation map and explanation after adding random noise to the image. left: effect of weight decay, middle: effect of softplus β (for λ = 5e-4), right: effect of curvature minimization (for λ = 5e-4, β=10).

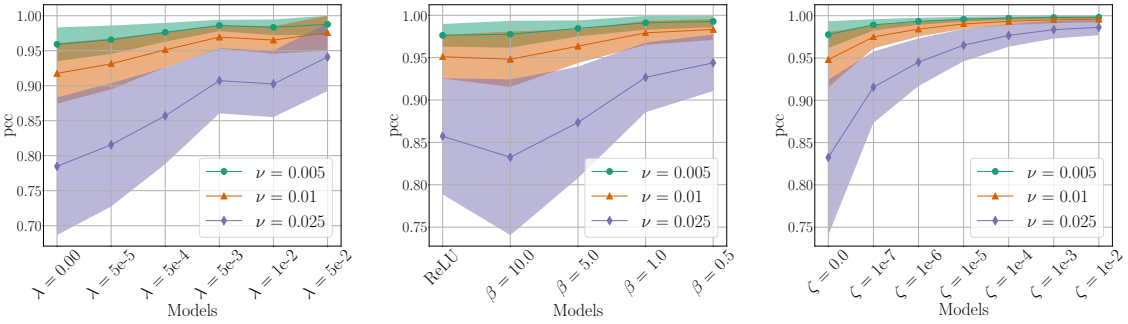


Fig. B.11.: PCCs (mean +/- std) between original Guided Backpropagation explanation map and explanation after adding random noise to the image. left: effect of weight decay, middle: effect of softplus β (for λ = 5e-4), right: effect of curvature minimization (for λ = 5e-4, β=10).

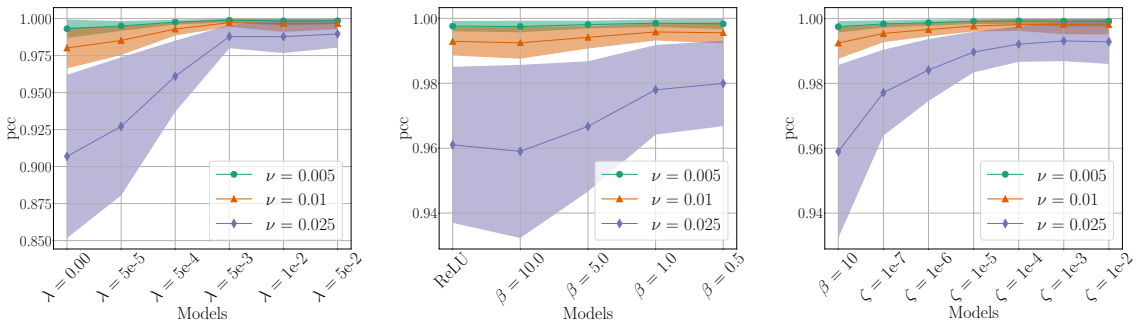


Fig. B.12.: PCCs (mean +/- std) between original Layerwise Relevance Propagation explanation map and explanation after adding random noise to the image. left: effect of weight decay, middle: effect of softplus β (for λ = 5e-4), right: effect of curvature minimization (for λ = 5e-4, β=10).

on the noise level ν. Figure B.13 shows effects on the Gradient explanation when adding Laplace noise to the input images. We see that the results look statistically very similar to the results for Gaussian noise.

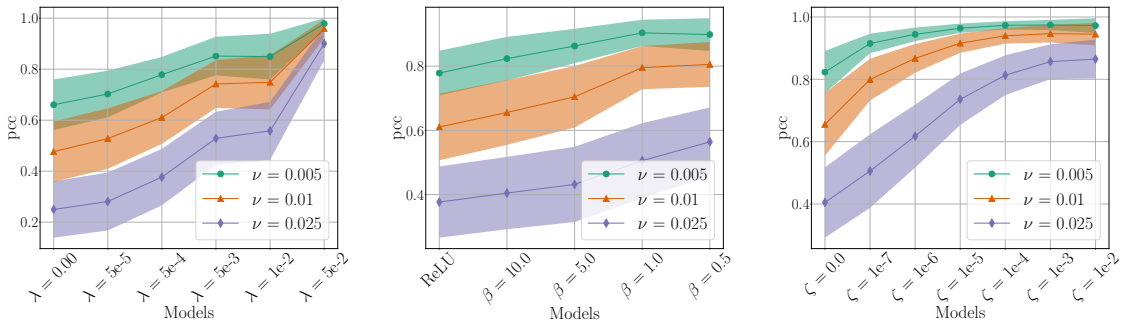


Fig. B.13.: PCCs (mean +/- std) between original Gradient explanation map and explanation after adding Laplace noise to the image. left: effect of weight decay, middle: effect of softplus β (for λ = 5e-4), right: effect of curvature minimization (for λ = 5e-4, β=10).

B.4.4.2. Salt-pepper noise

To perturb an image with salt-pepper noise we randomly select $50 \cdot \nu$ % of the pixels in the image and switch them to $x_{\max}$ (white) or $x_{\min}$ (black) at random. We select a very small amount of pixels (for noise level $\nu = 0.005$ only 3 pixels) to be perturbed as salt-pepper noise has a very strong effect on the classification accuracy which we aim to keep approximately constant.

Figure B.14 shows effects on the Gradient explanation when adding salt-pepper noise to the input images. We can still see a significant improvement in robustness when training networks with our proposed methods. However the effect for softplus activations and Hessian minimization is less pronounced than for Laplace or Gaussian noise.

B.5. Hessian norm approximation

In Section 5.1.3, we showed that for the number of samples $N \to \infty$ the sampling approximation approaches the true Hessian norm. To include the Hessian approximation in our training procedure, we need to fix a certain number of samples and then perform a Monte-Carlo estimate. Figure B.15 shows the relative error between the sampled and the true Hessian norm. Increasing the sample size noticeably reduces the error, but a sample size of one already has a relative error of only 6%. If we average over a batch of images, the error reduces further. This means that for training and validation sampling once per image is in practice sufficient.

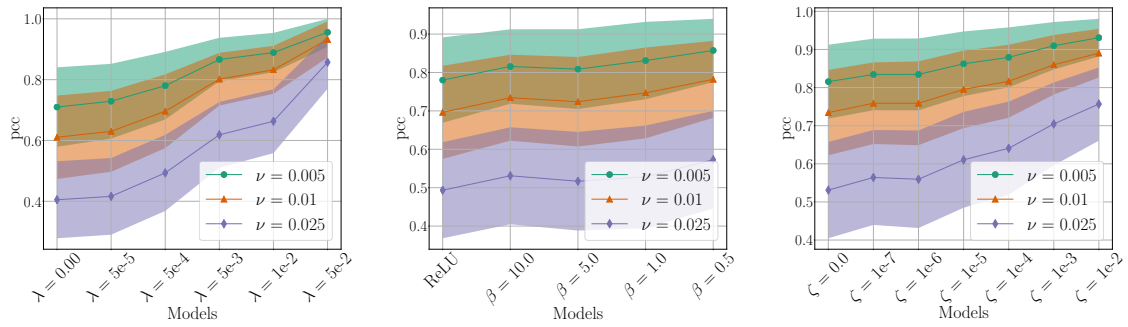


Fig. B.14.: PCCs (mean +/- std) between original Gradient explanation map and explanation after adding salt-pepper noise to the image. left: effect of weight decay, middle: effect of softplus β (for $\lambda = 5e\text{-}4$), right: effect of curvature minimization (for $\lambda = 5e\text{-}4$, β=10).

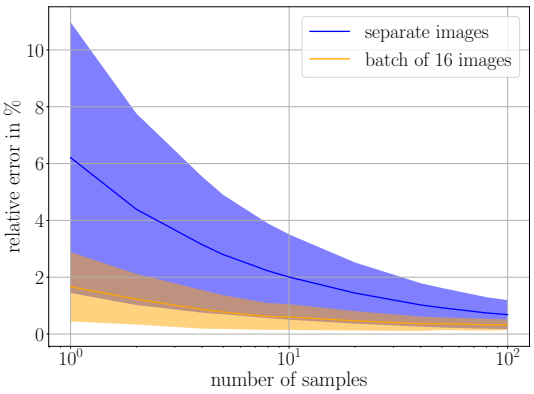


Fig. B.15.: Relative error (mean and standard deviation) between True Hessian norm and approximation via sampling

B.6. Additional network structures and data sets

Network architectures: In addition to our simple CNN we use two different architectures to test our methods: VGG16 [159] without batch normalization and ResNet18 [9].

Configuration: We select the most promising parameters for each of our proposed methods and train with four different parameter configurations for each network architecture. We train a vanilla network without weight decay ($\lambda = 0$), a network with moderate weight decay ($\lambda = 5e\text{-}4$), a network with softplus activations ($\lambda = 5e\text{-}4$, $\beta = 10$) and a network with curvature minimization ($\lambda = 5e\text{-}4$, $\beta = 10$, $\zeta = 1e\text{-}6$). All networks were trained for 60 to 70 epochs to make them more comparable and selected based on their validation set accuracy. We list test accuracies for the different networks in Table B.2.

architecture	λ	β	ζ	accuracy
VGG16	0.0	ReLU	0.0	84.0
VGG16	5×10^{-4}	ReLU	0.0	84.2
VGG16	5×10^{-4}	10	0.0	79.6
VGG16	5×10^{-4}	10	10^{-6}	80.8
ResNet18	0.0	ReLU	0.0	89.0
ResNet18	5×10^{-4}	ReLU	0.0	89.8
ResNet18	5×10^{-4}	10	0.0	88.6
ResNet18	5×10^{-4}	10	10^{-6}	89.2

Tab. B.2.: Accuracies on ImageNette test set for VGG16 and ResNet18 with different parameter configurations.

In Figure B.16 we compare weight norms and hessian norms between the three network architectures we analyse. Compared to the CNN trained on CIFAR, approximated Hessian norms for VGG16 are an order of magnitude smaller even without curvature minimization. For ResNet18 they are two orders of magnitude smaller.

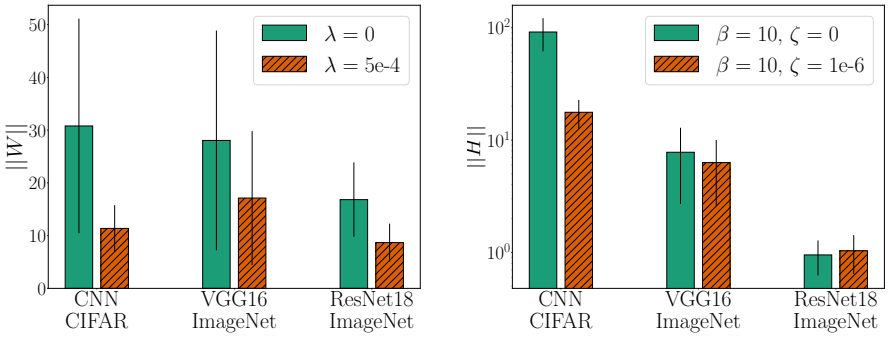


Fig. B.16.: Left: effect of weight decay on different architectures. Right: effect of curvature minimization on different architectures (a weight decay of $\lambda =$ 5e-4 was used)

C. Appendices for Chapter 6

C.1. Proofs

C.1.1. Proof of Theorem 4

In this section, we provide the proof for Theorem 4, which we repeat here for completeness.

Theorem 4. *For $\epsilon \in (0,1)$ and g a normalizing flow with Kullback–Leibler divergence* $\mathrm{KL}(p,q) < \epsilon$*,*

$$\gamma_{\perp_i}^{-1} \to 0 \qquad as \qquad \delta \to 0$$

for all $i \in \{1, \ldots, N_{\mathcal{X}} - N_{\mathcal{D}}\}$.

Since normalizing flows are diffeomorphisms, g^{-1} exists and is differentiable, $\mathcal{Z} = \mathcal{X}$ and γ is a non-singular metric on all of $\mathcal{X}$. Furthermore, the base distribution $q : \mathcal{X} \to \mathbb{R}$ transforms like a density,

$$q_x(x) = q_z(g^{-1}(x)) \left| \frac{\partial z^a}{\partial x^\alpha} \right| , \tag{C.1}$$

where $q_{x,z}$ denote q in z^a and x^α coordinates, respectively, $q_{x,z} : \mathbb{R}^{N_{\mathcal{X}}} \to \mathbb{R}$. We will assume that q_z is the univariate Gaussian distribution.

We assume that the Kullback–Leibler divergence between p and q is small, i.e. that

$$\mathrm{KL}(p,q) < \epsilon \tag{C.2}$$

for some small $\epsilon \in (0,1)$. Then, since $\ln(1/a) \geq 1 - a$,

$$\begin{aligned} \epsilon &> \int_{S_x} p_x(x) \ln \left(\frac{p_x(x)}{q_x(x)} \right) \mathrm{d}x \\ &\geq \int_{S_x} p_x(x) \left(1 - \frac{q_x(x)}{p_x(x)} \right) \mathrm{d}x \\ &= 1 - \int_{S_x} q_x(x) \mathrm{d}x \end{aligned} \tag{C.3}$$

and therefore

$$\int_{S_x} q_x(x) \mathrm{d}x > 1 - \epsilon \,. \tag{C.4}$$

Intuitively, this means that most of the induced probability mass lies in the support of p.

We now write q in z^a coordinates using (C.1) and then evaluate the integral in y^μ coordinates,

$$\begin{aligned} 1-\epsilon &< \int_{S_x} q_z(g^{-1}(x)) \left|\frac{\partial z^a}{\partial x^\alpha}\right| \mathrm{d}x \\ &= \int_{S_y} q_z(g^{-1}(x(y))) \left|\frac{\partial z^a}{\partial x^\alpha}\right| \left|\frac{\partial x^\alpha}{\partial y^\mu}\right| \mathrm{d}y\,. \end{aligned} \tag{C.5}$$

Using the block-diagonal form (6.21) of the metric in y^μ coordinates, the integration measure simplifies to

$$\left|\frac{\partial z^a}{\partial x^\alpha}\right| \left|\frac{\partial x^\alpha}{\partial y^\mu}\right| = \sqrt{|\gamma_{\mu\nu}|} = \sqrt{|\gamma_{\mathcal{D}}|} \prod_{i=1}^{N_{\mathcal{X}}-N_{\mathcal{D}}} \sqrt{|\gamma_{\perp_i}|}\,. \tag{C.6}$$

Therefore, we have

$$1-\epsilon < \int_{\mathcal{D}_y} \sqrt{|\gamma_{\mathcal{D}}|} \prod_{i=1}^{N_{\mathcal{X}}-N_{\mathcal{D}}} \int_{-\delta/2}^{\delta/2} \sqrt{|\gamma_{\perp_i}|} q_z(z(y)) \mathrm{d}y^i_\perp \mathrm{d}y_\parallel\,. \tag{C.7}$$

Since q_z is bounded, as $\delta \to 0$, we need $|\gamma_{\perp_i}| \to \infty$ in order to keep the integral above the bound. Therefore, $\gamma^{-1}_{\perp_i} \to 0$ for $\delta \to 0$.

C.1.2. Proof of Theorem 5

In this section, we provide the proof for Theorem 5, which we repeat here for convenience.

Theorem 5. *If $g : \mathcal{Z} \to \mathcal{X}$ is a generative model with $\mathcal{D} \subset g(\mathcal{Z})$ and image $g(\mathcal{Z})$ which extends in any non-singular orthogonal direction $y^i_\perp$ outside of $\mathcal{D}$,*

$$\gamma^{-1}_{\perp_i} \to 0$$

for $\delta \to 0$ for all non-singular orthogonal directions $y^i_\perp$.

For any $x_{\mathcal{D}} \in \mathcal{D}$, let $x_0 \in S$ be on the negative $y^i_\perp(x_{\mathcal{D}})$ coordinate line such that $p(x_0) < \epsilon$ for some small ϵ and let $x_1 \in S$ be on the positive $y^i_\perp(x_{\mathcal{D}})$ coordinate line such that $p(x_1) < \epsilon$, as illustrated in Figure 6.6. Then, the assumption that $g(\mathcal{Z})$ extends beyond $\mathcal{D}$ in non-singular directions implies that the segment of the $y^i_\perp(x_{\mathcal{D}})$ coordinate line between x_0 and x_1 lies entirely in $g(\mathcal{Z})$.

Let $\tau : [0,1] \to S$ be the coordinate-line segment between x_0 and x_1. In summary, we have

1. $\tau(0) = x_0$ and $\tau(1) = x_1$ with $p(x_0) < \epsilon$ and $p(x_1) < \epsilon$

2. $\exists\, t_{\mathcal{D}} \in [0,1]$ such that $\tau(t_{\mathcal{D}}) = x_{\mathcal{D}} \in \mathcal{D}$
3. $\tau_{\perp}^{j} = 0$ for $j \neq i$, $\tau_{\parallel} = \text{const.}$

In particular, 3) implies that the tangent vector of τ points along the $y_{\perp}^{i}$ coordinate vector: $\tau'(t) \propto \partial_{y_{\perp}^{i}}$.

Let $L(\tau)$ denote the length of τ, i.e.

$$L(\tau) = \int_0^1 \sqrt{\gamma(\tau'(t), \tau'(t))}\, \mathrm{d}t \tag{C.8}$$

$$= \int_0^1 \sqrt{\gamma_{\mu\nu}(\tau(t)) \frac{\mathrm{d}\tau^{\mu}}{\mathrm{d}t} \frac{\mathrm{d}\tau^{\nu}}{\mathrm{d}t}}\, \mathrm{d}t\,. \tag{C.9}$$

Following point 3) above, we can perform the implicit sums over μ and ν and get

$$L(\tau) = \int_0^1 \sqrt{\gamma_{\perp_i}(\tau(t))} \left| \frac{\mathrm{d}\tau^{i}}{\mathrm{d}t} \right| \mathrm{d}t \tag{C.10}$$

$$= \int_{x_{0,\perp}{}^{i}}^{x_{1,\perp}{}^{i}} \sqrt{\gamma_{\perp_i}(y_{\perp}^{i})}\, \mathrm{d}y_{\perp}^{i}\,. \tag{C.11}$$

Since S has, by construction, (Euclidean) extension δ orthogonal to $\mathcal{D}$ in y^{μ} coordinates, with $\delta \ll 1$, we perform a Taylor expansion of the metric around $x_{\mathcal{D}}$

$$\gamma_{\mu\nu}(\tau(t)) = \gamma_{\mu\nu}(x_{\mathcal{D}}) + \mathcal{O}(\tau^{\rho}(t) - x_{\mathcal{D}}^{\rho}) \tag{C.12}$$

and obtain to first order

$$L(\tau) \approx \sqrt{\gamma_{\perp_i}(x_{\mathcal{D}})}\, (x_{1,\perp}{}^{i} - x_{0,\perp}{}^{i})\,. \tag{C.13}$$

Again, since S has range δ in $y_{\perp}^{i}$-direction, we have $(x_{1,\perp}{}^{i} - x_{0,\perp}{}^{i}) < \delta$ and therefore

$$\gamma_{\perp_i} > \frac{L^2(\tau)}{\delta^2}\,. \tag{C.14}$$

We now change the x^{α}- and y^{μ} coordinates such that $\delta \to 0$, corresponding to a data distribution which is more and more concentrated on $\mathcal{D}$. As we change coordinates, $L(\tau)$ is constant as a geometric invariant[1] and we obtain from (C.14)

$$l\gamma_{\perp_i}^{-1} \to 0\,, \tag{C.15}$$

as desired.

[1] Since τ lies entirely in $g(\mathcal{Z})$, there is a curve σ in $\mathcal{Z}$ whose image under g is τ. Together with the properties 1) and 2) of τ this implies in particular that $L(\tau) = L(\sigma)$ is not infinitesimal.

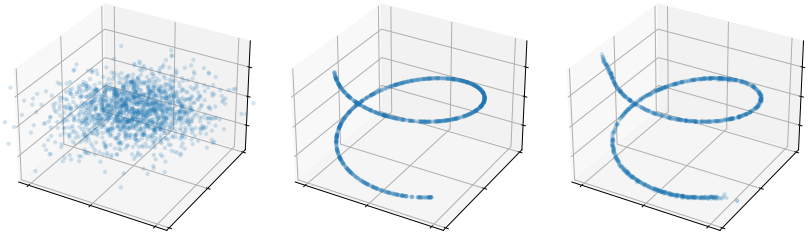


Fig. C.1.: From left to right: distribution in the base space of the flow, target distribution, learned distribution

C.2. Details on Experiments

C.2.1. Toy Example

The flow used for the toy example is composed of twelve RealNVP-type coupling layer blocks. Each of these blocks includes a three-layer fully-connected neural network with leaky ReLU activations for the scale and translation functions.

For training, we sample from the target distribution defined by

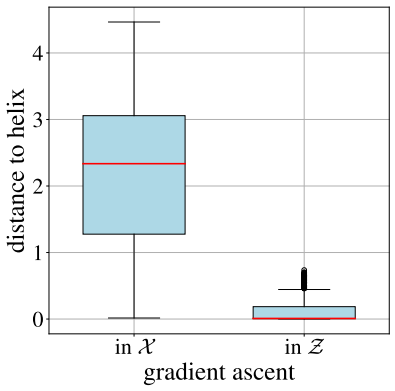
$$\begin{aligned} x_3 &\sim U(-4, 4)\,, \\ x_2 &= \cos(x_3)\,, \\ x_1 &= \sin(x_3)\,. \end{aligned}$$

We train for 5000 epochs using a batch of 500 samples per epoch. We use the Adam optimizer with standard parameters and learning rate $\lambda = 1 \times 10^{-4}$. This takes around 10 minutes on a standard CPU. After successful training, we can map samples from a multivariate standard normal distribution to the data distribution, see Figure C.1.

In order to train a classifier we first define the ground truth: points with z-coordinate smaller than zero belong to the one class and points with z-coordinate bigger than zero belong to the other class. We train a neural network with 256 hidden neurons with ReLU activations and one output neuron with sigmoid activation to near perfect accuracy on this classification task.

We then run the gradient ascent optimization in image space $\mathcal{X}$ and in the base space of the flow $\mathcal{Z}$. We start from samples from the true data distribution and set the target to 0.1 if the network predicted a value larger than 0.5 for the original data point, otherwise we set the target to 0.9.

As we have an analytical description of the helix which serves as the data manifold in this toy example, we can calculate the shortest distance between any point and the helix. The figure on the right shows that the counterfactuals found in $\mathcal{Z}$ lie significantly closer to the data manifold than adversarials found in $\mathcal{X}$. We test this for 1000 optimizations in $\mathcal{Z}$ and $\mathcal{X}$ respectively. Boxes in the plot extend from lower to upper quartile, red lines mark the medians, whiskers mark the 1.5×IQR (interquartile range) and circles mark outliers.



For more details we refer to our github implementation[2].

C.2.2. Generative models

Flows

We show generated samples for all Flows in Figure C.2. We use the RealNVP[3] architecture for MNIST and the Glow[4] architecture for CelebA and CheXpert. For the training, we use the Adam optimizer with a learning rate of 1×10^{-4} and weight decay of 5×10^{-4} for all flows.

MNIST: we train for 30 epochs on all available training images. Bits per dimension on the test set average to 1.21.

CelebA: we train for 8 epochs on all available training images. We use 5 bit images. Bits per dimension on the test set average to 1.32.

CheXpert: we train for 4 epochs on all available training images. Bits per dimension on the test set average to 3.59.

Mall: we train for 47 epochs on all available training images. Bits per dimension on the test set average to 0.96.

GANs

MNIST: We use a deep convolutional GAN[5] that we train for 170 epochs using the Adam optimizer with weight decay of 5×10^{-4} and learning rate of 2×10^{-4}.

[2] https://github.com/annahdo/counterfactuals/blob/main/toy_example.ipynb
[3] https://github.com/fmu2/realNVP
[4] https://github.com/rosinality/glow-pytorch
[5] https://github.com/AKASHKADEL/dcgan-mnist

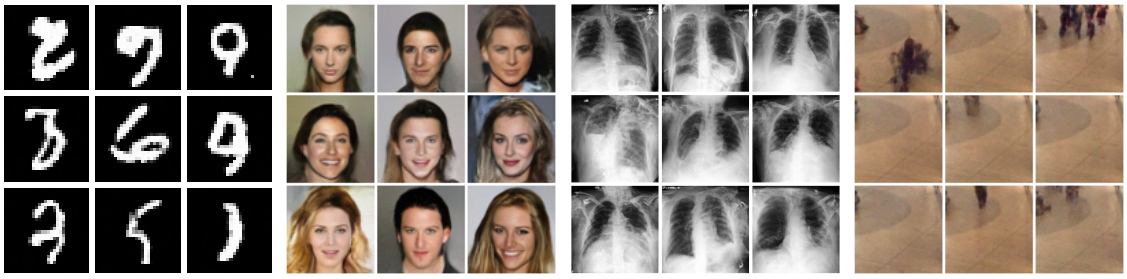


Fig. C.2.: Generated samples for all normalizing flows used in the paper. From left to right: RealNVP on MNIST, Glow on CelebA, Glow on CheXpert and Glow on Mall.

CelebA: We use a progressively grown GAN[6] and train on 600000 randomly selected training images.

CelebA-HQ: We use the pre-trained HyperStyle[7] GAN.

VAEs

We use Adam without weight decay and with a learning rate of $\lambda = 5 \times 10^{-3}$ for all VAEs.

MNIST: We train a simple convolutional VAE for 80 epochs. To evaluate the IM1 measure we train two additional VAEs with the same structure on only the training images with label four and nine respectively for 100 epochs.

CelebA: The VAEs for CelebA are only used to evaluate the IM1 measure. We train 2 simple convolutional VAEs for 100 epochs on all training images with blond attribute equal to one and all training images with blond attribute equal to 0 respectively.

C.2.3. Classifiers

All classifiers have a similar structure consisting of convolutional, pooling and fully connected layers. We use ReLU activations and batch normalization. For MNIST we use four convolutional layers and three fully connected layers. For CelebA and CheXpert we use six convolutional layers and four fully connected layers. For the training, we use the Adam optimizer with a weight decay of 5×10^{-4} for all classifiers.

MNIST: We train for 4 epochs using a learning rate of 1×10^{-3}. We get a test accuracy of 0.99.

CelebA: We train a binary classifier on the blond attribute. We partition the data sets into all images for which the blond attribute is positive and the rest of the

[6] `https://github.com/rosinality/progressive-gan-pytorch`
[7] `https://github.com/yuval-alaluf/hyperstyle`

images. We treat the imbalance by undersampling the class with more samples. We train for 10 epochs using a learning rate of 5×10^{-3}. We get a balanced test accuracy of 93.63% by averaging over true positive rate (93.95%) and true negative rate (93.31%).

CheXpert: We train a binary classifier on the cardiomegaly attribute. For the training data the cardiomegaly attibute can have four different values: blanks, 0, 1, and -1. We label images with the blank attribute as 0 if the no finding attribute is 1, otherwise we ignore images with blank attributes. We also ignore images where the cardiomegaly attribute is labeled as uncertain. Using this technique, we obtain 25717 training images labeled as healthy and 20603 training images labeled as cardiomegaly. We do not treat the imbalance but train on the data as is. We train for 9 epochs using a learning rate of 1×10^{-4}. We test on the test set, which was produced in the same way as the training set. We get a balanced test accuracy of 86.07% by averaging over the true positive rate (84.83%) and true negative rate (87.27%).

C.2.4. U-Net:

The U-Net [210] follows an hourglass structure. The first part consists of multiple convolutional, batch normalization, ReLU, and pooling layers that gradually reduce the spatial dimensions while increasing the channel dimensions. The second block consists of upsampling, concatenation of feature maps from the first part, convolutional, batch normalization and ReLU activation layers. The last layer has the same spacial dimension as the input but only one channel corresponding to a probability map. For using the U-Net in order to count pedestrians, Ribera et al. [211] add an additional fully connected layer with ReLU activations that combines the information from the last layer and the central encoding layer to estimate the number of objects of interest present[8].

C.2.5. Optimization of counterfactuals and adversarial examples

Counterfactuals and adversarial examples are found using the Adam optimizer with standard parameters. We vary only the learning rate λ. For our main experiments, we use the base space of normalizing flows to find the counterfactuals. We set the threshold for the confidence of the target class high when searching for counterfactuals and adversarial examples. We therefore get more visually expressive results. Of course in practice, one might wish to find counterfactuals with lower target confidence. We show an example optimization with different confidence thresholds in Figure C.3.

MNIST: We use $\lambda = 5 \times 10^{-4}$ for conventional adversarial examples and $\lambda = 5 \times 10^{-2}$ for counterfactuals found via the flow. We do a maximum of 2000 steps stopping

[8] `https://github.com/javiribera/locating-objects-without-bboxes`

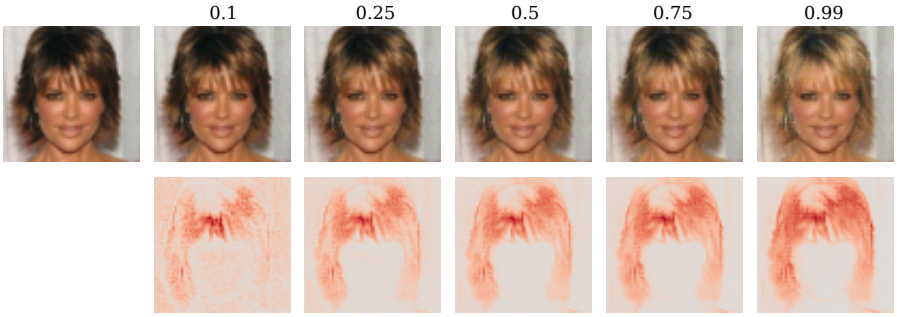


Fig. C.3.: Top row: original image and evolution throughout optimization. Numbers indicate confidence with which the image is classified as 'blond'. Second row: absolute differences to original image summed over color channels.

early when we reach the target confidence of 0.99. We perform attacks on 500 images of the true class 'four'. All conventional attacks and 498 of the attacks via the flow reached the target confidence of 0.99 for the target class 'nine'.

CelebA: We use $\lambda = 7 \times 10^{-4}$ for conventional adversarial examples and $\lambda = 5 \times 10^{-3}$ for counterfactuals found via the flow. We do a maximum of 1000 steps stopping early when we reach the target confidence of 0.99. We perform attacks on 500 images of the true class 'not-blond'. 492 conventional attacks and 496 of the attacks via the flow reached the target confidence of 0.99 for the target class 'blond'.

CheXpert: We use $\lambda = 5 \times 10^{-4}$ for conventional adversarial examples and $\lambda = 5 \times 10^{-3}$ for counterfactuals found via the flow. We do a maximum of 1000 steps stopping early when we reach the target confidence of 0.99. We perform attacks on 1000 images of the true class 'healthy'. All conventional attacks and 990 of the attacks via the flow reached the target confidence of 0.99 for the target class 'cardiomegaly'.

Mall: We use $\lambda = 1 \times 10^{-4}$ for conventional adversarial examples and $\lambda = 5 \times 10^{-3}$ for counterfactuals found via the flow. We do a maximum of 5000 steps stopping early when we reach the target regression value of 10 when we are maximizing pedestrians and 0.01 when we are minimizing pedestrians. We perform attacks on 100 images with few people (average regression value of 0.7) and 100 images with many people (average regression value of 3.6). All attacks reached the target values.

GANs: For counterfactuals found in the latent space of GANs we do a maximum of 1000 steps with $\lambda = 5 \times 10^{-3}$ for MNIST, CelebA and CelebA-HQ.

VAEs: For counterfactuals found in the latent space of the VAE trained on MNIST we do a maximum of 1000 steps with $\lambda = 5 \times 10^{-3}$.

Pedestrians' head positions for counterfactuals and adversarial examples with original and oracle U-Net

In Figure C.4, we show the localization of heads for the counterfactuals and the adversarial examples for the Mall data set from Figure 6.9 using the original and the oracle U-Net. In order to find the head locations, the regression value is rounded to the closest integer representing the number of pedestrians in the image. A Gaussian mixture model with the number of pedestrians as components is then fitted to the probability map. Finally the head positions are defined as the means of the fitted Gaussians.

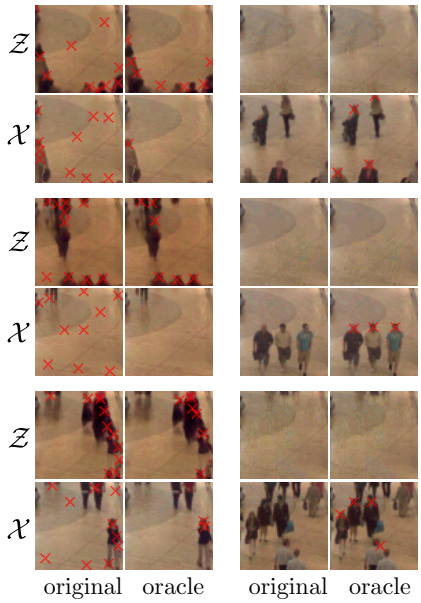


Fig. C.4.: Head locations detected with original and oracle U-Net.

Figure C.4 shows that the original U-Net is deceived by the adversarial examples (rows marked by $\mathcal{X}$): When maximizing pedestrians (first column) the original U-Net produces false positives, leading to markers at locations where there are no pedestrians. When minimizing pedestrians, the adversarial examples (third column) fool the original U-Net into making false negative errors, that is failing to detect pedestrians, although they are clearly present. The oracle U-Net on the other hand produces regression values and probability maps that enable correct identification of pedestrian's head positions (or lack thereof) for the adversarial examples when maximizing (second column) and minimizing (forth column) pedestrians. For the diffeomorphic counterfactuals (rows marked by $\mathcal{Z}$), the predictions of the two U-Nets are similar, showing that these counterfactuals generalize to the independently trained oracle U-Net.

C.2.6. Similarity to source images

To evaluate the proximity between counterfactuals/adversarials and their corresponding source images we calculate the Euclidean differences in $\mathcal{X}$ space as well as in $\mathcal{Z}$ space and compare them to the respective Euclidean differences for all images of the source class. For MNIST, CelebA and CheXpert we calculate the Euclidean differences for a maximum of 2000 test images for each counterfactual/adversarial. For the Mall data set we calculate the distances to 400 training images with $r < 1$ when considering counterfactuals/adversarials for which we maximized r and we calculate the distances to 400 training images with $r > 3$ when considering counterfactuals/adversarials for which we minimized r. In addition we calculate all Euclidean differences between counterfactuals/adversarials and their respective source images. We show the distribution of distances to the respective source images and to all images of the source class in Figure C.5.

As expected the Euclidean differences between adversarial examples and their respective source images are very small when measured in $\mathcal{X}$ space but a lot larger when measured in $\mathcal{Z}$ space. The Euclidean differences for counterfactuals measured in $\mathcal{X}$ are larger than those for adversarials but we can still observe that counterfactuals are significantly closer to their respective source image than to other images of the same class. The Euclidean distances measured in $\mathcal{Z}$ space are significantly smaller for counterfactuals than for adversarials, indicating that adversarial examples lie off manifold.

The effect is less pronounced for images from the Mall data set, as those have little variance in the background. Comparing similarities between source image and images of the source class might therefore not be as meaningful as for the other datasets.

C.2.7. Similarity between all images

We calculate the Euclidean distances in $\mathcal{X}$ and $\mathcal{Z}$ of randomly selected test images (all classes), adversarial examples and counterfactuals to randomly selected images from the training data set (all classes). Figure C.6 shows the distribution of distances for the data sets MNIST, CelebA and CheXpert. We note that for distances in $\mathcal{X}$ the distributions for original images, adversarial examples and counterfactuals are very similar while for distances in $\mathcal{Z}$ the distribution of distances for adversarial examples is notably shifted to the right, meaning that adversarial examples are further away from random data samples when the distances are measured in $\mathcal{Z}$, that is on the manifold. The effect is most notable for CelebA and CheXpert, for which the distances in $\mathcal{Z}$ of counterfactuals closely match the distribution of distances between images from the data set.

The original distribution of images from the Mall data set is strongly skewed towards few pedestrians. We can therefore not expect to achieve insights from comparing distributions of manipulated images.

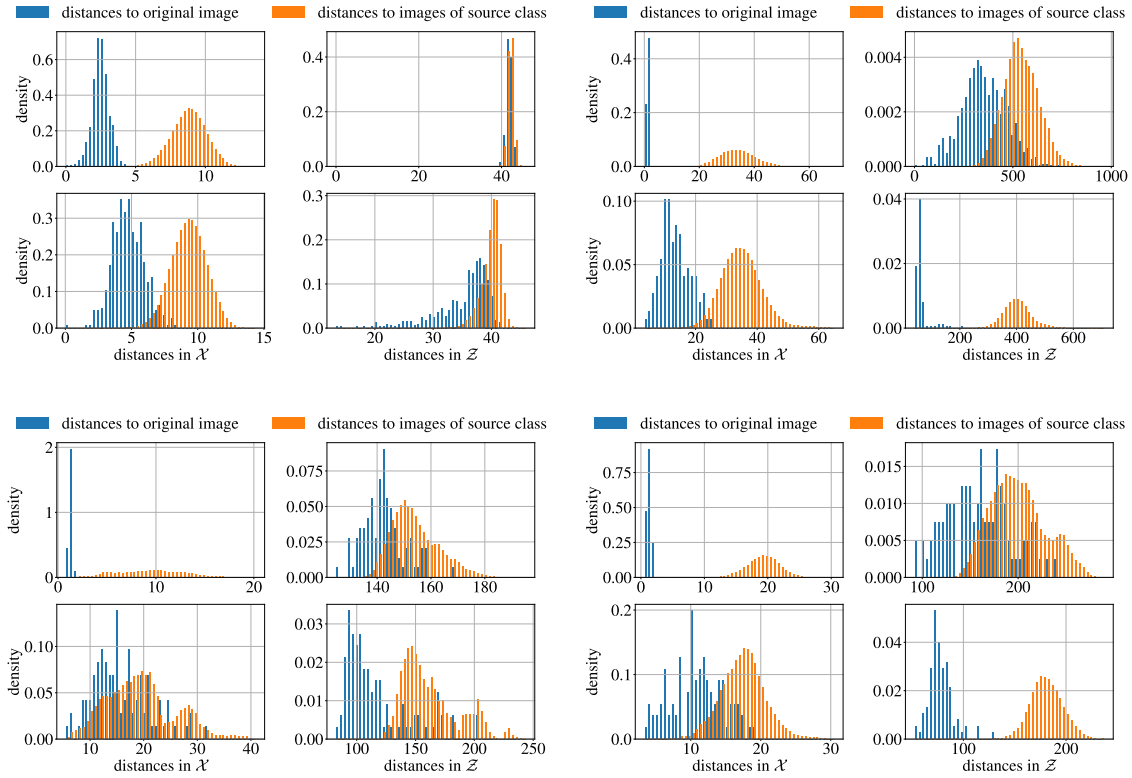


Fig. C.5.: Euclidean distances in $\mathcal{X}$ and $\mathcal{Z}$ for adversarial examples (uneven rows) and counterfactuals (even row) for the MNIST (top left), CheXpert (top right), Mall from few to many (bottom left) and Mall from many to few (bottom right).

C.3. Additional experiments

C.3.1. Unsatisfactory results

For some images it seems like our method does not obtain satisfactory counterfactuals. We show some examples for the CelebA data set in Figure C.7. Possible explanations from left to right could be: It is unclear what should happen to a bald person when going blond. People with dark skin and blond hair are rarely seen in the data set so the generative model has difficulty generating such images. People with hats are not very common in the data set, so the hat is mistaken for hair and made blond.

C.3.2. Revealed class correlations

Counterfactuals can reveal biased classifiers or data sets. For the CelebA data set the blond attribute seems correlated with light skin and make up, see Figure C.8. This can be a feature rather than a bug if one aims to discover said biases. If the behavior is unwanted one could include additional regularizers that keep the prediction for attributes other than the target attribute close to their initial values.

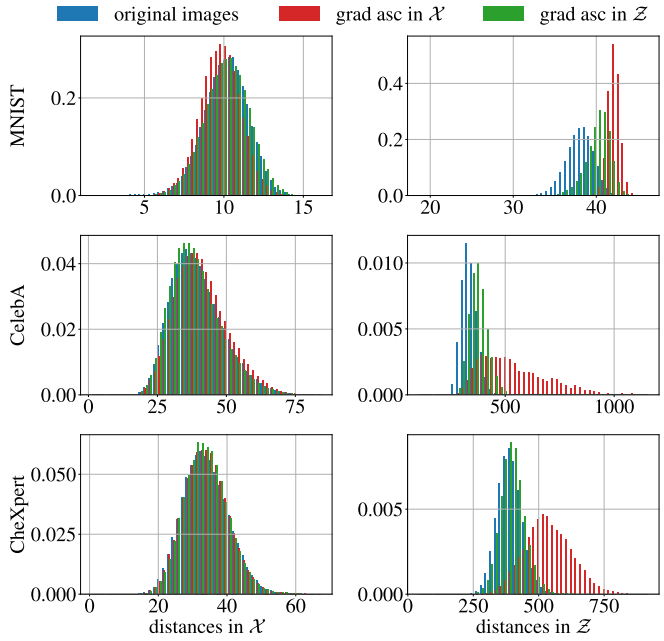


Fig. C.6.: Distributions of Euclidean distances in $\mathcal{X}$ and $\mathcal{Z}$ for test images, adversarial examples and counterfactuals for three data set.

Fig. C.7.: For these examples our method did not achieve satisfactory results.

C.3.3. Different target queries

Different classes for source and target images lead to qualitatively different results. For MNIST (see Figure C.9) transitions from '6' to '2' or from '5' to '1' yield counterfactuals that are not easily identified as the target class by a human.

For Glow used with CelebA we observe that our approach fails to generate satisfactory images for target classes 'hat' and 'glasses' (see Figure C.10). These failures may be due to the quality of the generative model as images of people with hats ($\approx 5\%$) and glasses ($\approx 6\%$) are comparatively rare in the CelebA training set. Generating counterfactuals in the opposite direction (from 'hat' to 'no hat' or 'glasses' to 'no glasses') gives more satisfactory results.

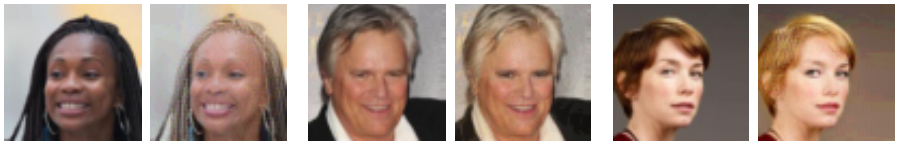


Fig. C.8.: Counterfactuals reveal correlations. Left: blond hair is associated with light skin. Middle and right: blond hair is associated with makeup.

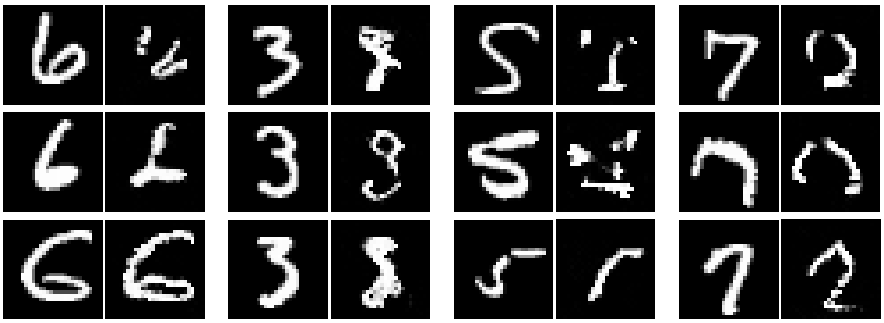


Fig. C.9.: Randomly selected counterfactuals for different source and target classes for the MNIST data set. Optimization tasks for blocks from left to right: $6 \to 2, 3 \to 8, 5 \to 1, 7 \to 0$.

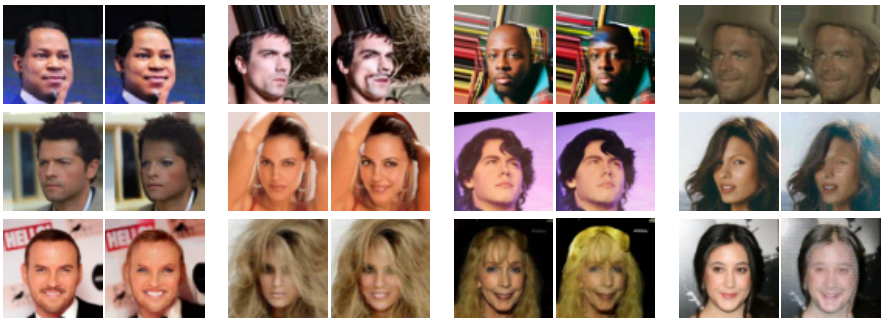


Fig. C.10.: Randomly selected counterfactuals for different source and target classes for the CelebA data set. Optimization tasks for blocks from left to right: 'male' to 'not male', 'not smiling' to 'smiling', 'no hat' to 'hat', 'no glasses' to 'glasses'.