Towards Explainable Artificial Intelligence

Interpreting Neural Network Classifiers with Probabilistic Prime Implicants

vorgelegt von
Stephan Wäldchen (M.Sc.)
ORCID: 0000-0001-7629-7021

von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation

Berlin 2022

Preface

"We must know, we will know."—thus concluded David Hilbert his retirement address to the Society of German Scientists and Physicians in 1930. His words were meant in staunch rejection of the idea of *Ignoramus et ignorabimus*, i.e. that human knowledge is fundamentally limited and parts of our reality will remain permanently unknowable.

When artificial intelligence arrived on the scientific stage, it was meant to supercharge our research and understanding. However, along the optimism grew a new concern: AI advancing its capabilities in every area while leaving human reasoning behind [Han+98]. Indeed, the most successful models applied today are likewise the least interpretable. A shift away from this *Ignoramus et ignorabimus* approach and towards algorithms that share their reasoning is urgently required: In 2016, a survey asked 352 top machine learning scientists when they expect AI will "be able to accomplish every task better and more cheaply than human workers" [Gra+18]. Averaged, they estimated a chance of 50% of this happening by 2061 and a 10% chance for as soon as 2025.

When I first started this PhD, I had no idea where I was going to end up thematically, driven more by inertia and vanity than concrete goals. That much, at least, has changed. The consensus that further human flourishing requires us to understand and control the algorithms we are creating is growing [Ord20]. Though scenarios as depicted in the *Terminator* films are not considered likely, even a slightly misaligned powerful AI can represent a huge risk [Bos12]. In light of these developments my own work now seems almost comically modest in scope and ambition. The progress of theoretically sound interpretable models appears to crawl compared to the futuristic advances of ever more powerful AI applications. Some argue these goals are ultimately incompatible.

Hilbert's famous words are now engraved on his tomb. As irony would have it, a day before his declamation, his colleague Kurt Gödel first demonstrated his incompleteness theorems—laying to rest Hilbert's dream of a complete and consistent foundation for mathematics.

Regarding the development of AI, we still have a choice: Shift the focus from the race for the best models to a common endeavour towards soundly robust and interpretable ones. There are least some signs that interpretable systems might not only be competitive but that they actually make AI's better [Cam+20]. It is my hope that this route is possible and that formal reasoning can contribute to it. In this sense, I would like to believe that this work adds to this effort if not on a scientific level then at least for my own path. To conclude, dear reader, let me echo the indelible words from the T800 in *Terminator 2*, one algorithm saving humanity from another: *"Come with me if you want to live."*

Abstract

In this thesis we develop a framework for interpreting the decisions of highly nonlinear classifier functions with a focus on neural networks. Specifically, we formalise the idea of separating the input parameters into relevant and irrelevant ones as an explicit optimisation problem.

First, we describe what is generally understood as a relevance map for classifiers and give an overview over the existing methods to produce such maps. We explain how we used relevance maps to detect artefacts in the PASCAL VOC dataset and track the focus of neural agents playing the Atari video games Breakout and Pinball on a human-like level.

Towards a formal definition of relevance maps, we generalise the concept of prime implicants from abductive logic to a probabilistic setting by introducing δ-relevant sets. For a d-ary Boolean function $\Phi\colon \{0,1\}^d \to \{0,1\}$ and an assignment to its variables $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ we consider the problem of finding those subsets of the variables that are sufficient to determine the function output with a given probability δ. We show that the problem of finding small δ-relevant sets is NP-hard to approximate with a factor $d^{1-\alpha}$ for $\alpha > 0$. The associated decision problem turns out to be $\mathsf{NP}^{\mathsf{PP}}$-complete.

We further generalise δ-relevant sets from the binary to the continuous domain. This leads naturally to a rate-distortion trade-off between the size of the δ-relevant set (rate) and the change in the classifier prediction (distortion). Relevance maps can then be interpreted as greedy approximations of the rate-distortion function. Evaluating this function even approximately turns out to be NP-hard, so we develop a heuristic solution strategy based convex relaxation of the combinatorial problem and assumed density filtering (ADF) for deep ReLU neural networks. This results in our own explanation method which we call Rate-Distortion Explanations (RDE).

To show that the approximations in ADF are necessary, we give a complete characterisation of families of probability distributions that are invariant under the action of ReLU neural network layers. We demonstrate that the only invariant families are either degenerate or amount to sampling.

Subsequently, we propose and discuss several benchmark tests and numerical evaluation methods for relevance maps. We compare RDE to a representative collection of established relevance methods and demonstrate that it outperforms competitors for a wide range of tasks.

Finally, we discuss how the knowledge of the true data distribution is crucial for any existing explanation method. We criticise our own method over potential artefacts and introduce a stronger, information theoretical requirement based on the conditional entropy. A novel approach, called Arthur-Merlin-regularisation along with a new framework is developed. The framework is then extended to realistic algorithms and data sets, and we discuss under which assumptions the guarantees still hold.

Zusammenfassung

In dieser Arbeit entwickeln wir ein Framework für die Interpretation der Entscheidungen hochgradig nichtlinearer Klassifizierungsfunktionen mit Schwerpunkt auf neuronalen Netzen. Insbesondere formalisieren wir die Idee der Trennung der Eingabeparameter in relevante und irrelevante Parameter als explizites Optimierungsproblem.

Zunächst beschreiben wir, was man allgemein unter einer Relevanzkarte für Klassifikatoren versteht, und geben einen Überblick über die bestehenden Methoden zur Erstellung solcher Karten. Wir erläutern, wie wir Relevanzkarten verwendet haben, um Artefakte im PASCAL-VOC-Datensatz zu finden und um den Fokus neuronaler Agenten beim Spielen der Atari-Videospiele Breakout und Pinball auf einer menschenähnlichen Ebene zu verfolgen.

Auf dem Weg zu einer formalen Definition von Relevanzkarten verallgemeinern wir das Konzept der primären Implikanten aus der abduktiven Logik auf eine probabilistische Umgebung, indem wir δ-relevante Mengen einführen. Für eine d-äre boolesche Funktion $\Phi\colon \{0,1\}^d \to \{0,1\}$ und eine Zuordnung zu ihren Variablen $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ betrachten wir das Problem, diejenigen Teilmengen der Variablen zu finden, die ausreichen, um die Funktionsausgabe mit einer gegebenen Wahrscheinlichkeit δ zu bestimmen. Wir zeigen, dass das Problem, kleine δ-relevante Mengen zu finden, NP-schwer mit einem Faktor $d^{1-\alpha}$ für $\alpha > 0$ zu approximieren ist. Das zugehörige Entscheidungsproblem erweist sich als $\mathsf{NP}^{\mathsf{PP}}$-vollständig.

Wir verallgemeinern ferner δ-relevante Mengen von der binären zur kontinuierlichen Domäne. Dies führt auf natürliche Art zu einem Raten-Verzerrungs-Kompromiss zwischen der Größe der δ-relevanten Menge (Rate) und der Veränderung der Klassifikation (Verzerrung). Relevanzkarten können dann als gierige Approximationen der Raten-Verzerrungs-Funktion interpretiert werden. Diese Funktion auch nur approximativ zu evaluieren ist NP-schwer, daher entwickeln wir eine heuristische Lösungsstrategie, die auf einer konvexen Relaxation des kombinatorischen Problems und dem Assumed Density Filtering (ADF) für tiefe neuronale ReLU-Netze basiert. Dies führt zu unserer eigenen Erklärungsmethode, die wir Rate-Distortion Explanations (RDE) nennen.

Um zu zeigen, dass die Näherungen in ADF notwendig sind, geben wir eine vollständige Charakterisierung von Familien von Wahrscheinlichkeitsverteilungen an, die unter ReLU neuronalen Netzwerkschichten invariant sind. Wir zeigen, dass die einzigen invarianten Familien entweder degeneriert sind oder auf Stichproben hinauslaufen.

Anschließend schlagen wir mehrere Benchmark-Tests und numerische Bewertungsmethoden für Relevanzkarten vor und diskutieren sie. Wir vergleichen die RDE mit einer repräsentativen Auswahl etablierter Relevanzmethoden und zeigen, dass sie bei einer Vielzahl von Aufgaben besser abschneidet als die Konkurrenz.

Schließlich erörtern wir, warum die Kenntnis der wahren Datenverteilung für jede bestehende Erklärungsmethode entscheidend ist. Wir kritisieren unseren eigenen Ansatz wegen möglicher Artefakte und führen eine stärkere informationstheoretische Anforderung ein, die auf der bedingten Entropie basiert. Es wird ein neuer Ansatz, die Arthur-Merlin-Regularisierung, zusammen mit einem neuen Framework entwickelt. Das Framework wird dann auf realistische Algorithmen und Datensätze ausgeweitet, und wir diskutieren, unter welchen Annahmen die Garantien noch gelten.

List of Publications

The results of this thesis have already been partially published by the author and his collaborators. The following list contains all publications and preprints concerning the content of this thesis, including many of the figures and tables. Stephan Wäldchen is a main contributor to all those publications. In particular, he substantially contributed to the conception and development of all mathematical proofs and numerical frameworks.

[Lap+19] **S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek and K-R. Müller. "Unmasking Clever Hans predictors and assessing what machines really learn", *Nature Communications* (2019)**

This work is incorporated in chapter 1 and chapter 3. It contains the results of the artefact detection in the PASCAL VOC data set and the the reinforcement learning setup for Atari video games interpreted with Layer-wise Relevance Propagation

[Wäl+21] **S. Wäldchen, J. Macdonald, S. Hauch, G. Kutyniok. "The computational complexity of understanding binary classifier decisions", *JAIR* (2021)**

The content of the paper is the basis for chapter 4, where we introduce probabilistic prime implicants and demonstrate hardness results for the problem of finding small implicants.

[Mac+19a] **J. Macdonald, S. Wäldchen, S. Hauch, G. Kutyniok. "A Rate-Distortion Framework for Explaining Neural Network Decisions", *arXiv preprint* (2019)**

We use this paper as the basis for chapter 5. It constitutes our first publication about Rate-Distortion-Explanations. It also contributes to the numerical investigation in chapter 7.

[Mac+20] **J. Macdonald, S. Wäldchen, S. Hauch, G. Kutyniok. "Explaining neural network decisions is hard", *XXAI Workshop, 37th ICML* (2020)**

This paper contributes to chapter 5, as it developed the idea of rate-distortion interpretation of relevance maps further and adds to the numerical experiments in chapter 7.

[MW21] **J. Macdonald, S. Wäldchen. "A Complete Characterisation of ReLU-Invariant Distributions", *arXiv preprint* (2021)**

This paper is the basis for chapter 6, where we justify our use of Assumed Density Filtering to calculate expectation values for neural network output in chapter 5.

Acknowledgements

The writing of this thesis was only possible with the help of many great people from my professional and personal life. I am grateful to everyone who supported me and accompanied throughout this time.

First of all, I want to thank my colleagues from the “Applied Functional Analysis” group at the TU Berlin, with whom I stood together through the toughest of times. Since I started my PhD in maths with a master in physics, my work would have been impossible without them. I am especially grateful to Jan Macdonald, for being a great research partner, bearing with me in our many hours of discussions at the blackboard, and for turning our ideas into actual mathematics.

I would further like to thank the supervisors from my graduate school “BIOphysical Quantitative Imaging Towards Clinical Diagnosis” (BIOQIC) Gitta Kutyniok and Ingolf Sack for their trust in me, their scientific input and their support. Most of all I am grateful to them for the freedom to explore my research ideas and to find my own path during my thesis.

To Peter Jung I am grateful for the many discussions we had and the insights he shared with me, as well as to Bubacarr Bah and Jessica Phalafala for our research exchange with the AIMS Capetown in 2018 (DAAD grant 57417688: “Non-Negative Structured Regression with Applications in Communication and Data Science”) and the awesome stay we had there. Furthermore, I want to thank Gabriele Penn-Karras and Albrecth Gündel-vom Hofe for turning my time as a teaching assistent a the TU Berlin into a fun and formative endeavour.

I am deeply grateful to Sebastian Pokutta for becoming my supervisor and welcoming me into his research group “Laboratory for Interactive Optimization and Learning” (IOL) at the Zuse Institute Berlin. Additionally, I want to thank Martin Skutella, Joao Marques-Silva and Philipp Petersen for being the referees of this thesis.

A big thank you goes to my friends Clara, Lucas, Markus and Isaak for proofreading my thesis and giving me valuable feedback, as well as to my flatmates for keeping up my spirits during the writing process.

On a personal level, I am immensely grateful to my family. Auf persönlicher Ebene bin ich meiner Familie unendlich dankbar. Einen besonderen Dank möchte ich an meine Mutter richten, die immer für mich da ist und sehr viel dafür geleistet hat, dass ich es einfacher im Leben haben konnte, als sie es selbst hatte. Auch meinem Onkel und meiner Tante möchte ich an dieser Stelle dafür danken, dass sie mich unermüdlich unterstützt haben.

Table of Contents

1 Introduction

Artificial intelligence emerged as not only one of the most promising technological paradigms of our time, but equally as one of the most controversial. While its applications to science, industrial processes and our daily life have become more numerous and impactful, these practical applications have far outpaced our theoretical understanding of their inner workings. Artificial intelligence may have started as a 'hard' science, but—as it becomes more powerful—it also 'softens', revealing an embarrassing gap between practice and theory, where rigorous mathematics gave way to successful heuristics [MOM12]. We may yet see the emergence of AI-neuroscience [Pot07], AI-sociology [Woo85] and AI-psychoanalysis [Pos20].

Whereas early successes in AI such as human-level chess computers like Deep Blue were explicitly programmed by human engineers, modern approaches utilise machine learning to build their models. Machine learning in turn has seen the advent of general-purpose models, large data sets and intensive computation as the de facto standard tools. Especially *neural networks* have achieved unprecedented success in tasks as wide ranging as medical imaging [Lee+17], autonomous driving [Gup+21], protein folding [Jum+21] and natural language processing [Bro+20].

Measured solely by their prediction accuracy, neural network-based models should give us no pause to advance in the aforementioned direction. However, the combination of data driven learning and highly flexible, parameter-rich models has one important drawback: They operate as opaque "black-box" methods that do not lend themselves to human understanding.

Traditional machine learning models such as linear regression, decision trees[1], or k-nearest neighbours allow for a somewhat straight-forward human interpretation of the model predictions. In contrast, the reasoning of deep neural networks remains generally inaccessible—a situation that is deemed problematic in high-stakes applications, such as medical imaging and diagnosis, employee recruitment and hiring, or autonomous driving. In particular, the conclusion in many scientific fields has so far been to prefer linear models [MSH07; Dev08; All+12; Hau+14] in order to rather gain insight (e.g. regression coefficients and correlations) even if this comes at the expense of accuracy, see illustration in fig. 1.1 for an illustration.

[1]Straightforward interpretability of decision trees has recently been disputed by Izza et al [IIM20].

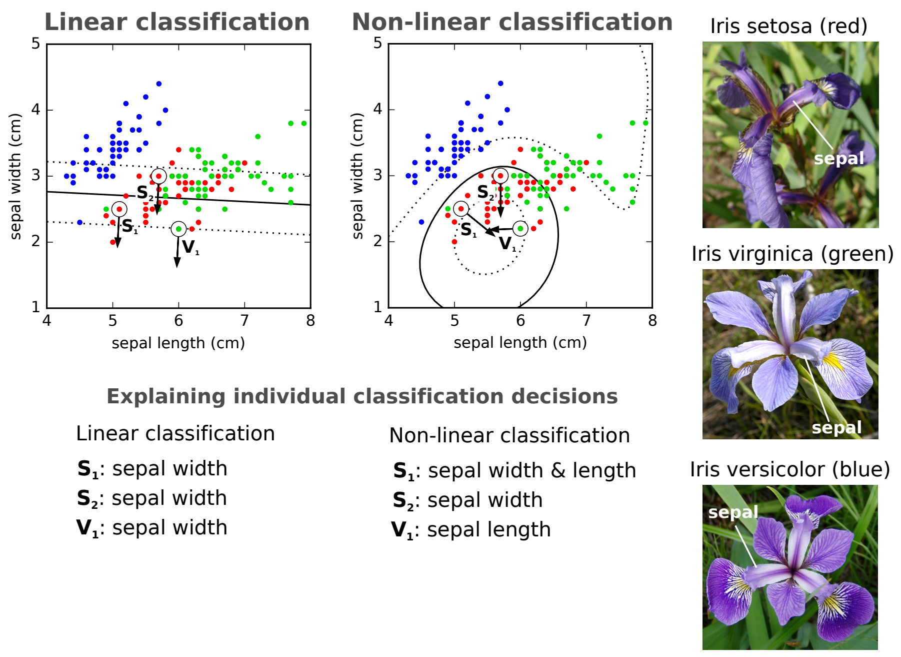


Figure 1.1: A comparison between linear and nonlinear classification on the Iris flower data set [Fis36]. In this example, the classifiers are trained to separate "Iris setosa" (red dots) from "Iris virginica" (green dots) and "Iris versicolor" (blue dots). In linear models the importance of each feature is the same for every data point. It can be expressed in the weight vector perpendicular to the decision surface where more important features have larger weights. We show the gradient direction for three example data points $\mathbf{s}_1, \mathbf{s}_2$ (iris setosa) and $\mathbf{v}_1$ (iris virginica). The liniar decision boundary has a clear interpretation: "Look at the sepal width!". The gradient uniformly points in this direction. The nonlinear decision boundary has a much better accuracy, but admits no such comprehensive interpretation.

This vacuum has sparked the field of *Explainable AI* (XAI) with the ambitious goal to remedy this fact by either using explicitly interpretable models or to furnish black-box models with an accompanying explanation of how the model operates. In this work we will predominantly consider the second case, more specifically extracting an explanation from a given neural network classifier. In the last chapter we develop an approach for an intrinsically explainable setup.

1.1 What do we mean by an Explanation?

Ideally, an explanation would describe the complete causal chain that lead to a prediction. One could imagine a logical formula of limited size in which every variable represents a concept that we as humans are already familiar with. At the moment there is no known mechanism that could translate a neural network classifier into such a formula, nor is it clear that such an explanation would always exist. While both humans and trained neural networks are able to recognise particular faces among millions of photos, describing this process in rigorous terms is not straightforward. If we were to describe a specific person to a human we generally would show example pictures and let them generalise the face to different moods, angles and lighting, without making anything algorithmically explicit—very similar to training an artificial classifier. So it's currently not clear how such an explanation would look like, whether it can be formulated at all, or how we would extract it.

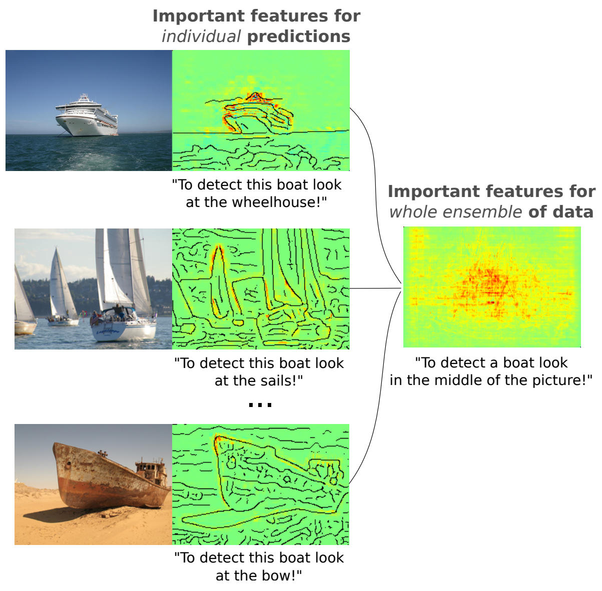

Important features for *individual* predictions
"To detect this boat look at the wheelhouse!"
"To detect this boat look at the sails!"
...
"To detect this boat look at the bow!"
Important features for *whole ensemble* of data
"To detect a boat look in the middle of the picture!"

Figure 1.2: Different features can be important to detect a ship in an image(here for a neural network, heatmaps produced with LRP, see section 1.2.3). In some instances the wheelhouse is a good indicator for class "ship", for others the sails or the bow is important. Therefore individual predictions exhibit very different heatmaps for different data points. In feature selection, one identifies salient features for the whole ensemble of training data. For ships the most salient region (average of individual heatmaps) is the center of the image.

A first step in this direction is the much simpler question of what part of the input has been relevant to the classification and which part has been irrelevant. Even though an answer would not constitute a complete explanation, this is the main question that we want to consider in this work. Following established literature, when we speak of explanations we will henceforth consider this question.

While feature selection approaches have traditionally explained the model by identifying features relevant for the whole ensemble of training data [GE03] or some class prototype [SVZ13; Yos+15; NYC16; Ngu+16] (globally important features), it is often necessary, especially for nonlinear models, to focus the explanation on individual examples (locally important features, see fig. 1.2). For data with a certain degree of translation invariance, like images or sound recordings, it is unlikely that a small region of input parameters will be important for every classification. A classifier that identifies dogs will have to use the whole image since a dog might be represented in any part of the image.

A recent series of work [ZF14a; SVZ13; Bac+15a; Bae+10; RSG16b; Zho+16; Mon+17] has now begun to explain the predictions of nonlinear machine learning methods in a wide set of complex real-world problems (e.g. [Stu+16; Gre+18; ZBM16; Arr+17a]). This is done by assigning an importance score to every input feature, a so called *relevance score*. These scores can be rendered as visual heatmaps, that can be interpreted by the user. We will refer to these heatmaps of relevance scores as *relevance maps* and to the algorithms that produce them as *relevance methods*.

Definition 1.1 Let $\Phi : [0,1]^d \to [0,1]$ be a classifier function, $\mathbf{x} \in [0,1]^d$ an input vector. We call any map or algorithm $\mathcal{R}$ that maps Φ and $\mathbf{x}$ to a vector of relevance scores $\mathbf{r} \in \mathbb{R}^d$, i.e.

$$\mathcal{R} : \Phi, \mathbf{x} \mapsto \mathbf{r},$$

a *relevance* method and a and a visual representation of $\mathbf{r}$ a *relevance map*.

Example 1.2 The constant map that assign a score of 1 to any input parameter and any function, $\mathrm{const}_\Phi(\mathbf{x}) = \mathbf{1}$, is a valid relevance method according to definition 1.1.

Example example 1.2 illustrates that the definition of relevance methods is very broad and not connected to any useful properties yet. Most of this thesis will be spent to remedy this fact and introduce tighter criteria of what constitutes a useful relevance method.

1.2 Existing Relevance Methods

The existing relevance methods can be broadly categorised into three approaches: Based on *local linearisation*, on *partial input* or on *backpropagation*. We briefly discuss the general idea for each category and give one prominent example.

1.2.1 Local Linearisation

As discussed, linear models are easy to interpret. So a natural approach for nonlinear models is to linearise them locally and interpret the linearisation. The simplest, and most local, approach is to use the gradient, i.e.

$$r_i = (\partial_i \Phi)(\mathbf{x}).$$

This has approach has been used for image [SVZ14] and text [TBR19] data, the corresponding heatmaps are called *saliency maps*.

However, the method has been critisised as very sensitive to slight variations of the classification function [Kin+19]. Adding a small noise to the classification function can both keep the classification of any data point stable as well as completely change the saliency maps.

A more non-local approach is *smooth grad* [Smi+17], a method that samples the gradient from input vectors close to $\mathbf{x}$. These samples are conventionally drawn from a normal distribution with mean $\mathbf{x}$ whose standard deviation can be used to tune how non-local the explanation should be. Of course, this approach only gives sensible result if the network is not piecewise constant—a network with Heaviside activation functions would always get relevance maps of constant zero.

A similar approach is pursued in the *Lime* framework [RSG16a]. Here, one samples datapoints and their class label around the input and fits a linear model to classify these new data points. The weights of this linear model can then be displayed as a heatmap for the original image. This approach works also for piecewise constant classifiers.

1.2.2 Partial-Input methods

Partial input methods rely on the existence of a so called *characteristic function*, a function that lets you evaluate partial input. This concept comes from cooperative game theory. For $d \in \mathbb{N}$ players, let a subset $S \subseteq [d]$, called a coalition, signify which players participate in a common task to gain some reward. The characteristic function $\nu\colon \mathcal{P}([d]) \to \mathbb{R}$ assigns a reward to every possible coalition.

To apply this reasoning to classifier functions, one needs a notion of what it means to exclude a parameter from the coalition. The most basic approach is to replace the excluded parameters with a baseline value, e.g. setting a pixel in an image to black. For a given classifier Φ and input $\mathbf{x}$ we can define a characteristic function $\nu_{\Phi,\mathbf{x}}$ as

$$\nu_{\Phi,\mathbf{x}}(S) := f(\mathbf{y}) \quad \text{with} \quad y_i = \begin{cases} x_i & i \in S \\ 0 & i \notin S. \end{cases}$$

As we will discuss in chapter 8, a more common approach is to use expectation values instead of baseline values.

One of the most prominent relevance attribution methods based on partial input are the *Shapley values*. We will sketch the basic idea here. The full set of players achieves a common reward which should then be redistributed to the individual players. This redistribution is supposed to be fair after a number of intuitive fairness criteria regarding the hypothetical reward for every coalition, given by ν, see [Sha53]. The Shapley values are the unique distribution that fulfil these ideas of fairness.

The Shapley value of the i-th variable (i-th player) is defined as

$$\phi_{\nu,i} = \sum_{S \subseteq [d] \setminus \{i\}} \binom{d-1}{|S|}^{-1} (\nu(S \cup \{i\}) - \nu(S)),$$

which can be interpreted as the marginal contribution of the i-th variable to the value ν averaged over all possible coalitions. We can use the Shapley values to calculate a relevance map as

$$r_i = \phi_{\nu_{\Phi,\mathbf{x}},i}.$$

In general it is #P-hard to compute Shapley values [DP94]. However, for certain types of value functions efficient approximation algorithms exist [FWJ08].

For neural networks no efficient approximation for the Shapley values is known. Since there are exponentially many possible coalitions, the Shapley values involve an exponential sum and are thus not directly feasile for non-linear high-dimensional problems. This lead to the development of special approximation methods, such as SHAP [LL17b] and hierarchical Shapley values [TLS21].

Our novel approach that we introduce in chapter 4 is a partial input method as well. As we will later explain, instead of summing over all possible coalitions, we will search for a small coalition S^* that ensures a value $\nu(S^*)$ that is close to $\nu([d])$.

1.2.3 Backpropagation-Based

Training neural networks requires calculating gradient updates for every parameter. This is done with a technique called *backpropagation*, for a thorough introduction see [GBC16a]. First introduced in [Wer74], this algorithm allows to propagate the gradient backwards through the network based on the chain-rule of differentiation.

The idea behind backpropagation-based relevance methods is to formulate different rules that instead of a gradient propagate relevance scores backwards through the neural network.

While the methods we looked at so far apply to any classifier function, the following methods have been designed specifically for neural networks, or more generally any acyclic computational graph. Their rules are derived heuristically for every elementary computation. Two of the earliest methods are the deconvolution method [ZF14b] and guided backpropagation [Spr+15]. The latter follows the calculation of the gradient, but sets negative values to zero after every network layer.

A widely applied method from this class is *Layer-wise Relevance propagation (LRP)* [Bac+15a]. Using LRP we illustrate the potential use cases that explanation methods have in the next section. Thus we want to discuss it here in more detail.

Layer-wise Relevance Propagation (LRP) Introduced in [Bac+15a], LRP is a method for explaining the predictions of a broad class of ML models, including neural networks and kernel machines [Arr+17a; Lap+16a; Stu+16; Hor+19; Yan+18; Tho+18].

The method is based on a propagation rule applied uniformly to all neurons in the network: Let Φ be a neural network of depth L, whose output for a specific input $\mathbf{x} \in \mathbb{R}^d$ is recursively defined as

$$\begin{aligned} \mathbf{x}^0 &= \mathbf{x}, \\ \mathbf{x}^l &= \rho\Big(\mathbf{W}\mathbf{x}^{l-1}\Big), \quad l = 1, \dots, L-1, \\ \Phi(\mathbf{x}) &= \mathbf{x}^L. \end{aligned}$$

The propagation rule of LRP is defined as a 'mirror image' of network output as

$$\begin{aligned} \mathbf{r}^L &= \Phi(\mathbf{x}), \\ \mathbf{r}^{l-1} &= \mathbf{V}^l\mathbf{r}^l, \quad l = 1, \dots, L-1, && (1.1) \\ \mathbf{r} &= \mathbf{r}^0, && (1.2) \end{aligned}$$

where $\mathbf{V} = [v_{ij}]$ is the contribution of neuron i to the activation of neural j, typically

$$v_{ij}^l = \frac{w_{ij}^l x_i^{l-1}}{\sum_i w_{ij}^l x_i^{l-1}},$$

though other rules have been proposed over time, see [Mon+19]. The propagation rule is applied in a backward pass starting from the neural network output $\Phi(\mathbf{x})$ until the input variables (e.g. pixels) are reached. Resulting scores can be visualised as a heatmap of same dimensions as the input (see Supplementary Figure 1).

LRP has been described as a special case of the framework Deep Taylor Decomposition [Mon+17], where some of the propagation rules can be seen as particular instances. Note that LRP rules have also been designed for models other than neural networks, in particular Bag of Words classifiers, Fisher vector models, and LSTMs (more information can be found in the Supplementary Note 3 and Supplementary Table 1). In chapter 3 we extensively discuss the use of LRP as an explanation method for the discovery of biases in datasets as well as strategic behaviour of neural network agents.

1.3 Organisation of this Thesis

We want to take some time to discuss the structure of this thesis.

In chapter 2 we present the concepts and notation that is used throughout this work, especially the definition of *neural networks*.

In chapter 3 we present a series of use cases for relevance maps that we produced with the LRP method. To be concrete, we use relevance maps to detect artefacts in an established image classification data set and track the development of neural agents playing Atari games. These examples motivate our search for a formal definition and provable guarantees for relevance methods.

In chapter 4 we introduce *probabilistic prime implicants*, a generalisation of a concept from abductive reasoning called *prime implicants*. We propose that the probabilistic formulation is more fit to high-dimensional neural networks. We derive a series of computational hardness results that show that non-trivial guarantees will not be provable for efficient methods.

In chapter 5 we generalise the prime implicants further to include continuous functions as well as distributions over a continuous domain. We explain how we can interpret relevance maps as greedy strategies to evaluate a rate-distortion function for different rates and show that non-trivial guarantees will be hard to come by generalising our complexity results form the previous chapter. We nevertheless develop our own heuristic solution strategy called *RDE* on the basis of convex relaxation of the combinatorial problem and *assumed density filtering* to calculate expectation values of neural network outputs.

In chapter 6 we show that the use of assumed density filtering is indeed justified by proving that no useful invariant probability distribution families exist for ReLU-neural networks apart from sampling.

In chapter 7 we analytically and numerically compare RDE to the most important relevance methods. For this we use several benchmark tests that have been proposed by XAI researchers as well as tests we proposed ourselves.

In chapter 8, we comment on some artefacts that can appear in relevance methods that incorrectly model real-world data distributions. To remedy this fact, we introduce *Arthur-Merlin Regularisation* as a new framework with certain guarantees on the quality of the produced prime implicants.

We will conlude our investigation in chapter 9 and give an outlook over our further research plans.

The appendices A to E accompany each chapter and contain proof details, as well as supplementary numerical results that were excluded from the main chapters for readability.

2 Notation

We start with introducing some useful notation. Here, we restrict ourselves to notation that is important to all the chapters and introduce specific concepts as soon as we need them. Notation introduced in a specific chapter may not carry over to another chapter. This thesis discusses a variety of different ideas and each chapter on its own uses a considerable amount of variable names. We resort to redefining common symbols such as "A" and "ξ" instead of using more arcane notation.

Basic Vector and Matrix Notation Vectors and matrices are always denoted by boldfaced lower- and uppercase symbols, while their components are denoted as lowercase symbols with their respective indices in subscript, e.g. $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$ and $\mathbf{A} = [a_{i,j}] \in \mathbb{R}^{d_1 \times d_2}$, where $d, d_1, d_2 \in \mathbb{N}$. The standard basis vectors of $\mathbb{R}^d$ are $\mathbf{e}_1, \dots, \mathbf{e}_d$. The letter d will always stand for the dimension of input vectors for classifiers irrespective of whether their values are real or Boolean.

We write $\mathbf{0}_d$ and $\mathbf{1}_d$ for the d-dimensional vector of all zeros or ones respectively. We write $\operatorname{diag}(\mathbf{x})$ for the diagonal matrix with entries given by a vector $\mathbf{x} \in \mathbb{R}^d$. We use $\odot$ and $\oslash$ for component wise (Hadamard) product and quotient respectively. Additionally, we write $\mathbf{x}^2 = \mathbf{x} \odot \mathbf{x}$ for brevity and consider univariate functions applied to vectors to act component-wise.

We write $[d] = \{1, \dots, d\}$ for the set of all indices of a d-dimensional vector $\mathbf{x} \in \mathbb{R}^d$. For a subset $S \subseteq [d]$ we denote by $\mathbf{x}_S$ the restriction of $\mathbf{x}$ to the components indexed by S, i.e. $\mathbf{x}_S = (x_i)_{i \in S} \in \mathbb{R}^{|S|}$. The complement of S is $S^c = [n] \setminus S$. For the space of k-sparse vectors of d dimensions we write Σ_k^d.

Probability Distributions, Measures and Random Variables We use $\mathfrak{B}(\mathbb{R}^d)$ for the *Borel σ-algebra* on $\mathbb{R}^d$ and $\mathfrak{R}(\mathbb{R}^d)$ for the set of *Radon Borel probability measures* on $\mathbb{R}^d$. We denote measures by small Greek letters μ, ν or curly uppercase letters $\mathcal{D}, \mathcal{V}, \mathcal{T}$ when associated with data sets. We define the *push-forward* of a measure $\mu \in \mathfrak{R}(\mathbb{R}^d)$ with respect to a measurable function $f : \mathbb{R}^d \to \mathbb{R}^d$ as

$$f_*(B) = \mu(f^{-1}(B))$$

for $B \in \mathfrak{B}(\mathbb{R}^d)$. We denote the *uniform distribution* on a domain $\Omega \subset \mathbb{R}^d$ as $\mathcal{U}(\Omega)$. The most relevant cases are $\mathcal{U}([0,1]^d)$ and $\mathcal{U}(\{0,1\}^d)$, the uniform distribution on the continuous and binary unit hypercube in d dimension respectively. The *normal distribution* with mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ is $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. To measure the distance between measures we use the total variation distance defined as

$$\|\mu - \nu\|_{TV} = \sup_{B\in\mathfrak{B}(\mathbb{R}^d)} |\mu(B) - \nu(B)|.$$

To condition a measure $\mu \in \mathfrak{R}(\mathbb{R}^d)$ on an event $C \in \mathfrak{B}(\mathbb{R}^d)$ we define a conditional measure

$$\mu|_C(B) = \frac{\mu(B \cap C)}{\mu(B)}$$

for $B \in \mathfrak{B}(\mathbb{R}^d)$. We furthermore define the entropy of a discrete random variable X that takes values from D as

$$H(X) = -\sum_{\mathbf{x}\in D} [P(X = \mathbf{x}) \log(P(X = \mathbf{x}))]$$

the conditional entropy with regards to an event $Y = \mathbf{y} \in D_Y$ as

$$H(X \,|\, Y = \mathbf{y}) = -\sum_{\mathbf{x}\in D} [P(X = \mathbf{x} \,|\, Y = \mathbf{y}) \log(P(X = \mathbf{x} \,|\, Y = \mathbf{y}))]$$

and the conditional entropy of X with respect to the (not necessarily discrete) random variable $Y \sim \mathcal{Y}$ as

$$H_{\mathcal{Y}}(X \,|\, Y) = \mathbb{E}_{\mathbf{y}\sim\mathcal{Y}}[H(X \,|\, Y = \mathbf{y})]. \tag{2.1}$$

We choose to drop the subscript indicating the distribution of Y when it is clear from context. We can thus define the mutual information between X and Y as $I(X; Y) = H(X) - H(X \,|\, Y)$.

Boolean Functions and Circuits Throughout our analysis, $\Psi\colon \{0,1\}^d \to \{0,1\}$ denotes a d-ary Boolean function. We refer to Ψ as a Boolean circuit whenever its explicit description by logical gates like AND, OR and NOT becomes important. We call the *description length* the number of logic gates in an explicit description of Ψ.

We use the usual symbols $\wedge$, $\vee$, $\neg$, and $\oplus$ for the logical conjunction, disjunction, negation, and exclusive disjunction respectively. Boolean functions will be used interchangeably as logical propositions, so $\Psi(\mathbf{x})$ is shorthand for the proposition $\Psi(\mathbf{x}) = 1$. For statements about the probability of logical propositions to hold we assume independent uniform distribution for all involved variables, thus

$$\mathbb{P}_{\mathbf{y}}[\Psi(\mathbf{y})] = \mathbb{P}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[\Psi(\mathbf{y})] = \frac{\left|\left\{\, \mathbf{y} \in \{0,1\}^d \,:\, \Psi(\mathbf{y}) = 1 \,\right\}\right|}{|\{\, \mathbf{y} \in \{0,1\}^d \,\}|},$$

and, conditioned to some event $A(\mathbf{y})$,

$$\mathbb{P}_{\mathbf{y}}(\Psi(\mathbf{y}) \,|\, A(\mathbf{y})) = \frac{\left|\left\{ \mathbf{y} \in \{0,1\}^d \,:\, \Psi(\mathbf{y}) = 1, A(\mathbf{y}) = 1 \right\}\right|}{|\{ \mathbf{y} \in \{0,1\}^d \,:\, A(\mathbf{y}) = 1 \}|}.$$

We omit the subscript whenever it is clear from the context over which variables the probability is taken. If the probability is taken over all variables of a Boolean function, we simply write $\mathbb{P}(\Psi)$ instead of $\mathbb{P}_{\mathbf{y}}(\Psi(\mathbf{y}))$.

Continous Functions and Neural Networks Throughout our work, Φ will denote a continuous, d-ary function, oftentimes specifically a *neural networks*. We work only with simple feed-forward architectures that will be described as follows.

Definition 2.1 (Neural network) Let $L \in \mathbb{N}$ and $d = d_0, \ldots, d_L \in \mathbb{N}$. We define

$$\Phi\colon \mathbb{R}^{d_0} \to \mathbb{R}^{d_L} \,:\, \Phi = \Phi_1 \circ \cdots \circ \Phi_L$$

where

$$\Phi_l(\mathbf{x}) = \rho(\mathbf{W}^l\mathbf{x} + \mathbf{b}^l) \quad \text{with} \quad \mathbf{W}^l \in \mathbb{R}^{d_l \times d_{l-1}}, \mathbf{b}^l \in \mathbb{R}^{d_l} \text{ for } l = 1, \ldots, L-1, \tag{2.2}$$

$$\Phi_L(\mathbf{x}) = \sigma(\mathbf{W}^L\mathbf{x} + \mathbf{b}^l) \quad \text{with} \quad \mathbf{W}^l \in \mathbb{R}^{d_L \times d_{L-1}}, \mathbf{b}^L \in \mathbb{R}^{d_L}. \tag{2.3}$$

as a neural network with L layers of width $(d_0, \ldots, d_L)$ and activation functions $\rho : \mathbb{R} \to \mathbb{R}$ and $\sigma : \mathbb{R}^L \to \mathbb{R}^L$. We denote the space of all such neural networks with $\mathcal{NN}^{d_0,\ldots,d_L}_{L,\rho,\sigma}$.

We refer to the Φ_l as the network *layers*, L as the *depth* and $\max_{l\in[L]} d_l$ as the *width* of the network. The matrices $\mathbf{W}^l$ and vectors $\mathbf{b}^l$ are called the *weight* matrices and *bias* vectors. They are collectively referred to as the network parameters and are optimised when approximating a function in an *empirical risk minimisation problem*, see e.g. [Vap91].

Historically, the activation function ρ has been set to sigmoid or tanh activation functions. However, since 2012, the *rectified linear unit* (ReLU) defined as $\rho(x) = \max(0, x)$ has emerged as the most popular choice [RZL17]. The realisation of σ depends on the output space. If the output is a probability vector, the activation of choice is a softmax function defined as $\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$. For positive output it can be an elementwise ReLU and for arbitrary real vectors it can simply be an identity. Besides affine linear (also called fully-connected) layers, many other types of layers have been proposed. Among the most popular are: convolutional layers [Fuk80; LB+95], especially for input data with translational invariance. Convolutional layers drastically reduce the amount of parameters. Pooling layers are oftentimes inserted between the linear layers to reduce the number of network nodes. Maximum pooling, for example, takes the maximum of a set number of activations. Another commonly used building block is the batch-normalisation or group normalisation layer [IS15], skip connections that directly link non-consecutive layers [He+16] and probabilistic layers such as drop out [Sri+14]. A more detailed description of these operations is not necessary at this point. For a comprehensive overview over neural networks, we refer the interested reader to [GBC16b; Nie15].

3 Use Cases of Explanation Methods

Before we take on the task of searching for a rigorous definition of relevance methods, we want to convince the reader that these methods can be usefully applied. Since relevance methods allow us to connect the reasoning of a classifier to our own intuition, their use cases fall into two broad categories:

1. When we trust the classifier less than our own intuition, we can use the explanation methods to compare its reasoning with our human intuition. This is done by comparing the relevance maps with what we intuitively think of as the relevant part of the input. This allows us to detect biases in our datasets and refine our training setup.

2. If we trust the classifier somewhat and otherwise do not have strong intuitions about what the relevant parts of the input should be, we can use the explanations to further our own understanding. An example would be a model that classifies long sequences of genes for a certain genetic disease, but only relies on a small part of the gene. If the prediction accuracy is high enough, this highlights a potential region to be investigated in the laboratory.

In this chapter we discuss examples of both scenarios. For the first case, we discover biases in an established image dataset for object class recognition. In the second case, we use the maps quantitatively to gain insight into the development of neural agents playing simple video games at a human-like level.

3.1 Assessing Generalisation beyond Test Error

We start by comparing the reasoning of two different classifiers that were trained on the Pascal VOC data set [Eve+10]. The first is an Improved Fisher Vector SVM classifier from [Cha+11; PSM10] as a state-of-the-art Bag of Words-type model. The second is a deep convolutional neural network that was pretrained on image net [Rus+15] and fine-tuned on Pascal VOC. For details on the architecture and training as well as on the choice of LRP-backpropagation rules see appendix A. We will see that, although the precision of both models is high, the

	boat	horse
FV	70.88	80.45
DNN	77.20	81.60

Table 3.1: Prediction performances per model and for the classes "boat" and "horse" on the Pascal VOC 2007 test set in percent average precision. For the accuracies for the other classes see table A.1 in appendix A.

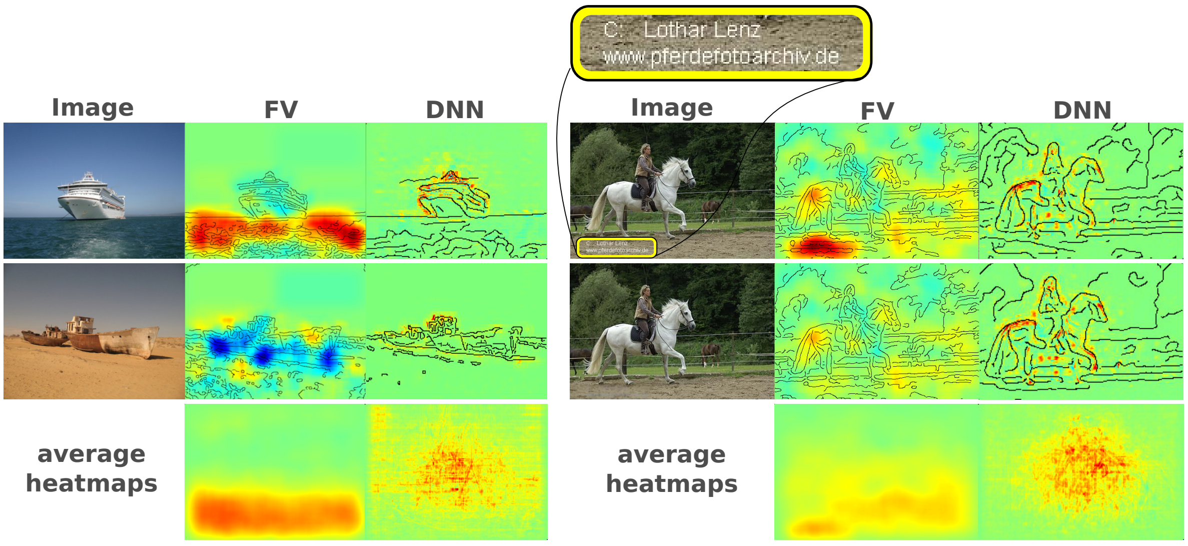


Figure 3.1: We compare the heatmaps of a Fisher vector classifier (FV) and a pretrained deep neural network (DNN) for boats and horses from the PASCAL VOC 2007 data set. *Left:* The FV focusses on the water to classify boats. A boat on sand is not recognised, whereas the DNN classifies both correctly. The *Right:* The FV classifies horses with the help of a data set artefact, namely a source tag. With the source tag removed it gives a much lower score. The DNN relies on the actual horse features and is insensitive to the tag. *Both:* The average heatmaps over the whole data set show the systematic reliance on water and the source tag for the FV.

Fisher Vector classifier relies on biases in the data set whereas the pretrained neural network is focused on the actual object to be classified.

3.1.1 Detecting Boats & Horses

The relevance maps computed for both the Fisher vector as well as the neural network classifier have been previously compared qualitatively as well as quantitatively in [Lap+16a], see table 3.1 for a comparison of the prediction accuracies.

Of special interest are our findings related to the categories "boat" and "horse". Both classes are predicted by both models with similarly high confidence (see table 3.1). Inspecting the decisive regions of the image with LRP, however, reveals for certain images substantial differences, as illustrated in fig. 3.1).

Detecting Boats In fig. 3.1 on the left, the input images show boats alongside pixel-level relevance maps computed for the corresponding classifiers, visualised as heatmaps. For the class "boat", the FV model predominantly bases its decisions on the presence of the water at the bottom of the image. This behaviour is not unique to this image. The FV model consistently predicts boats according to the presence of water for almost all samples in the dataset, as can be seen by the average heatmaps over the dataset. The DNN however has

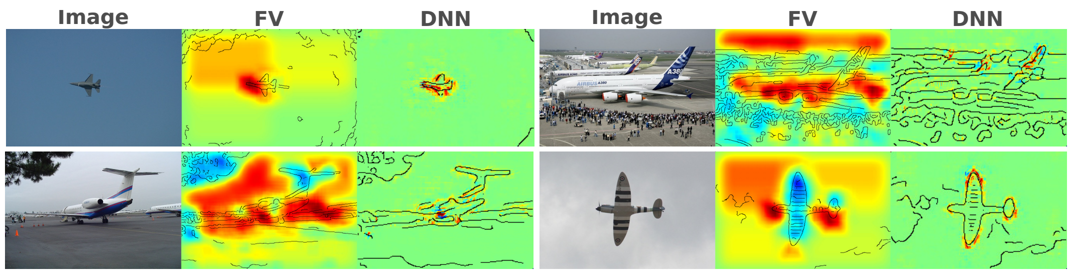


Figure 3.2: Example heatmaps for class "aeroplane", for which both the DNN and the FV model predict well. The attention of the DNN model generally concentrates on the body of the airplanes. The FV model also has learned to recognize the airplanes themselves reliably, presumably due to the pronounced edges and the contrast-rich geometry, but still uses the sky in both topmost quadrants of the image for support. Negative relevance scores are attributed to areas of the top half of the image where the clear sky obstructed by an object, such as a tree or parts of the airplane itself.

based its decision on the silhouette of the boat itself, in line with human intuition. When an image of a shipwreck on land is passed as input to both classifiers (see second row on the left in fig. 3.1, not contained in the PASCAL VOC data set), the relevance maps computed for class "boat" show that the DNN still finds positive evidence in the outline of the wreck and predicts the correct class. The FV model detects evidence strongly contradicting the target class "boat" and predicts the label "aeroplane" instead.

For other object classes, the FV model has learned a similarly biased contextualised prediction rule. The prediction of object category "aeroplane", for example, relies heavily on sky as background in the top left and top right quadrants of the image. Figure 3.2 shows some examples for class "aeroplane", for which both models predict with high average precision (AP) rating.

Detecting Horses The most remarkable example is the following. The FV model has developed a strong bias towards making use of a copyright watermark to detect samples of class "horse", as shown in the right half of fig. 3.1 — notably this artefact had gone undetected when establishing the benchmark.

Removing the watermark, as shown in the second row of fig. 3.1, results in a false negative prediction of the FV model. In the explanatory heatmap computed using LRP, the aggregation of strong positive relevance in the area of the copyright watermark vanishes as well, while the remainder of the heatmap remains unchanged. The neural network model is invariant to this input change in prediction and heatmap response.

Upon closer inspection of the Pascal benchmark data on which the FV model has been trained and evaluated, similar copyright watermarks were located in the bottom centre, or bottom left of the image in approximately 20% of all images containing horses. For the remaining images, we could identify a different artefact. A large part were taken from a horse show tournament setting with obstacles and hurdles being jumped over. The FV classifier identified these obstacles as decisive image features representing the class "horse".

For both the "horse" and the "boat" classes, average heatmaps for the FV classifier (left) and the DNN (right) are shown in fig. 3.1. Whereas for the FV model, positive relevance tends to aggregate in the same area — the bottom of the image for class "boat" and the bottom as well as the bottom left corner for class "horse" where the copyright watermark is located —

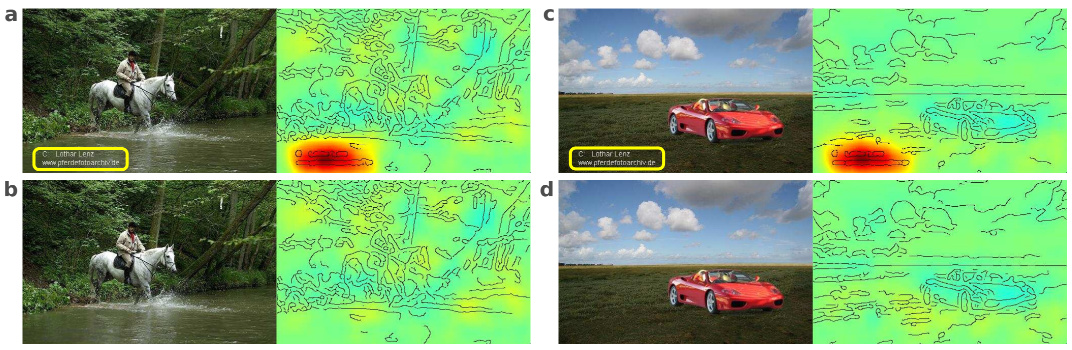


Figure 3.3: Images used as input for a FV model trained to detect horses, next to the corresponding relevance map. a) shows an image from the Pascal VOC 2007 dataset containing a copyright watermark, causing a strong response in the model. In b), the watermark has been edited out. The artificially created images c) and d) show a sports car on a lush green meadow with and without an added copyright watermark. In samples a) and c) the presence of class "horse" is detected, whereas in samples b) and d) this is not the case.

the average relevance response computed for the DNN classifier shows distinguishable localised patterns around the image centre, recapitulating our observations on a class-wide scale and underlining that the DNN model predicts based on the true object appearance, which shifts and differs from image to image.

The presence of simple and easy to learn visual features which are not part of the actual animal explain the classifier's strong focus on contextual information at prediction time, and the relatively low amounts of relevance attributed to the pixels showing the horses themselves.

To underscore this point, fig. 3.3 shows an artificially created image, where a sports car has been placed on a lush green meadow, causing a true negative prediction by the FV model. By adding the copyright watermark corresponding to class "horse" to the image, the model recognises the tag and false positively classifies the image as "horse".

3.1.2 Quantitative Evaluation

To quantify the above observations we make use of the fact that the Pascal VOC data set comes with image annotations in the form of tightly fit bounding boxes for the objects corresponding to the class.

For both the FV and the fine-tuned DNN model we collect all correctly labelled image samples from the evaluation data set and and compute an average outside-inside relevance ration $\mu \in [0, 1]$ as

$$\mu = \frac{|P_{\text{out}}|^{-1} \sum\limits_{q \in P_{\text{out}}} r_q}{|P_{\text{in}}|^{-1} \sum\limits_{p \in P_{\text{in}}} r_p}, \tag{3.1}$$

where P_{in} is the set of pixels inside the class-specific object bounding boxes and P_{out} covers the set of pixels outside. Large values indicate model decisions being largely based on background information, whereas low values speak for object-centred predictions. In fig. 3.4, we report the average ratios per class over all samples representing that class.

Both bar charts confirm that the DNN predicts by using visual information provided by the class-representing objects themselves, while the FV model resorts more to using contextual information. Well-predicted classes such as "aeroplane" and especially "boat" are predicted

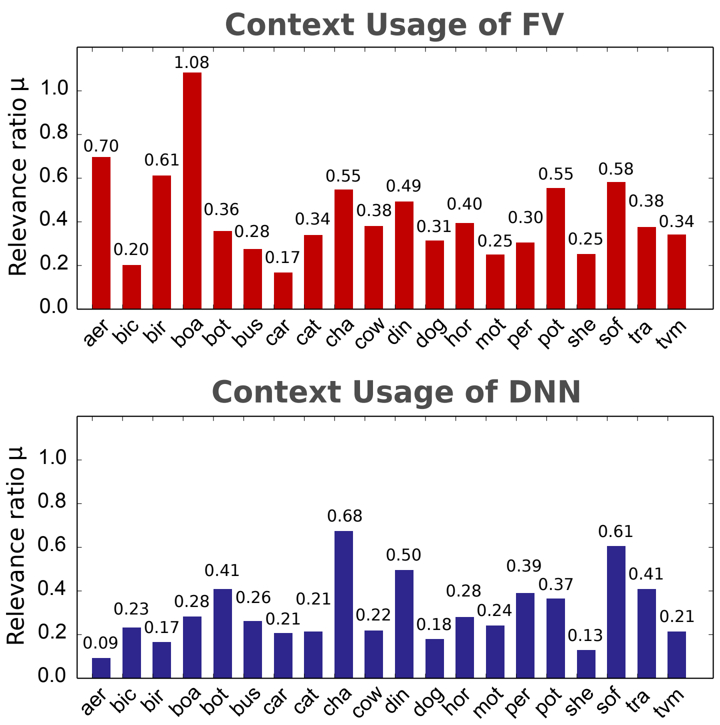


Figure 3.4: *L*eft: Measurements reflecting the importance of image context per class and model. Higher values of μ as defined in eq. (3.1) correspond to on average large amounts of positive relevance being located outside the object bounding boxes.

based on a consistently occurring contextual bias. For these classes the model might predict the label correctly in a majority of trials, i.e. the model performs well *in number*, but it has not learned to generalise the concept well and will most likely fail when deployed for a task outside the controlled benchmark laboratory setting.

3.1.3 Why do FV and DNNs exhibit different strategies ?

Of course, this question cannot be answered with certainty and the reasons we give here are speculations. Nevertheless, they are plausible and can be tested.

1. The Pascal VOC 2007 training dataset is comparably small and exhibits strong correlations between features, e.g. water and boats as well as horse and source tag. The source tag-horse correlation is simply a mistake by the creators. The water boat correlation is natural, and it is somewhat human-like to use water as a signifier to identify boats. The data set could be augmented by including images of water without boats, or boats in a different environment.

2. Features like water are simpler and more homogeneous compared to boats. A freshly trained classifier can be expected to have an easier time picking up the former. Likewise, the source tag always has the same form at the same position. Since these features are sufficient for prediction the more complicated feature will not be learned.

3. The DNN was pretrained on the much larger image net that contains 1000 classes and at least 500 images per class. Many of the Imagenet classes conceptually overlap with the classes present in the Pascal VOC data. The class "horse" from Pascal VOC corresponds to the subcategories "zebra" and "sorrel" in Imagenet.

4. Successful fine tuning implies that many of the low to mid level features of a neural networks, i.e. the weights and biases in the lower layers, can be reused. This means the parameters of the network start closer to an optimum configuration that generalises well.

A close inspection of the PASCAL VOC data set (of 9963 samples [Eve+10]) that typically humans never look through exhaustively, shows that such source tags appear distinctively on horse images; a striking artifact of the dataset that so far had gone unnoticed [Lap+16a]. Therefore, the FV model has ‘overfitted’ the PASCAL VOC dataset by relying mainly on the easily identifiable source tag, which incidentally correlates with the true features, a clear case of ‘Clever Hans’ behaviour [Lap+19].

We suggest to improve the PASCAL VOC dataset by removing the horse images with source tag and including images for each class that show the class object in an unusual background, e.g. boats on a dry deck.

3.2 Tracking the Focus of Neural Agents in AI Tasks

In a recent development machine learning has achieved human-level play in complex games, e.g Atari games [Mni+15; Mni+13], Go [Sil+16; Sil+17; Sil+18], StarCraft [Vin+19] and Texas hold'em poker [Mor+17]. This has led to speculations whether these algorithms show signs of embodying true "intelligence".

Our second investigation deals with a reinforcement learning setup. We trained neural agents to play simple Atari games on a human-like level, as first demonstrated in [Mni+15; Mni+13]. We used LRP to calculate the relevance maps corresponding to each action taken and track the attention of the network both during play and during training.

Reinforcement Learning In a typical reinforcement learning setting, an agent is trained without explicitly labelled data, via interaction with the environment only. Given an action space $\mathcal{A}$ of possible actions and a state space $\mathcal{S}$ of possible game states: At each time step $t \in \mathbb{N}$ of the training process, the agent performs a preferred action $a_t \in \mathcal{A}$ based on a state $s_t \in \mathcal{S}$ and receives a reward $r_t \in \mathbb{R}$. Its goal is to maximise the so called *long term reward* $R = \sum_t \gamma^t r_t$, which is expressed as a discounted sum, where $\gamma > 0$ is the discount parameter. The agent is trained to maximise R by adapting the model parameters $\boldsymbol{\theta}$ that determine which action is to be chosen given an input.

Recently, a convolutional neural network was trained to play simple Atari video games with visual input on a human-like level [Mni+15]. Using Q-learning [Wat89], a model-free reinforcement learning technique, a network has been trained to fit an action-value function. The Q-learning setup consists of three elements:

1. **the Arcade Learning Environment** [Bel+13] that takes an action as input, advances the game and returns a visual representation of the game and a game score,
2. **the Neural Network** that predicts a long-term reward, the so-called Q-function $Q(s, a; \boldsymbol{\theta})$, for a game visual for every possible action a, and, lastly,
3. **the Replay Memory** that saves observed game transitions during the training as tuples `(state, action, reward, next_state)`, from now on (s_t, a_t, r_t, s_{t+1}).

To update the network, Mnih et al. [Mni+15] make use of the fact that the optimal Q-function, Q^* must obey the Bellman equation [Bel52]

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \;\middle|\; s_t, a_t\right], \tag{3.2}$$

which can be used to train the network by choosing the cost function as the squared violation of the Bellman equation. The expectation value is approximated by the sum over a batch of game transitions B drawn uniformly at random from the replay memory.

$$C(\boldsymbol{\theta}) = \sum_{(s,a,r,s') \in B} \left(Q(s, a; \boldsymbol{\theta}) - \left[r + \gamma \max_{a'} Q(s', a'; \boldsymbol{\theta})\right]\right)^2.$$

The agent is trained by alternating three steps. First, it explores its environment using the actions that promise the highest long-term reward according to its own estimations. Second, it records the observations it encounters and saves the game transitions in the replay memory, potentially replacing older transitions. Third, one trains the network to predict the long-term reward by drawing batches from the replay memory and updating the cost functional.

The interplay of environment, network and memory is a convoluted process where convergence is not guaranteed [KLM96]. Methods to explain the network decisions can be used to chart the development of the neural agent even at stages where it is not yet able to successfully interact with the environment. As we will see later, this will be useful to correct the network architecture early on.

Our final architecture consists of three convolutional layers and two inner product layers. The exact architecture from [Mni+15] is described in fig. A.3 and in section A.3.

Visualisation and Comparison to Game Play LRP is used to back-propagate the dominant action through the network onto the input which results in a $4 \times 84 \times 84$-sized tensor of relevance values, for details see section A.3. To create a heatmap, we sum over the first axis and normalise the values to a value range of $[-1\ 1]$ per pixel — $r_i \leftarrow r_i / (\max_j |r_j|)$ — and rescale the image via linear interpolation to the original size of 210×160 pixels. To give an intuition for interpreting the heatmaps, we demonstrate how they correspond to different situations and strategies for two examples, the games *Breakout* and *Video Pinball*, both of which show above-human performance for the trained learning machines [Mni+15].

3.2.1 Neural Network playing Pinball

Our first agent was trained to play the Atari games Pinball. As shown in [Mni+15], the DNN achieves excellent results beyond human performance. We construct LRP heatmaps to visualize the DNN's decision behavior in terms of pixels of the pinball game.

Interestingly, after extensive training, the heatmaps become focused on few pixels representing high-scoring switches and completely loose track of the flippers. A subsequent inspection of the games in which these particular LRP heatmaps occur, reveals that DNN agent learned to move the ball into the vicinity of a high-scoring switch without using the flippers at all. It "nudges" the virtual pinball table such that the ball infinitely triggers the switch by passing over it back and forth, without causing a tilt of the pinball table (see Figure 3.5 for a step by step walk through).

The model has learned to circumvent the tilting mechanism through the right sequences of nudging and flipping. From a pure game scoring perspective, it is indeed a rational choice to exploit any game mechanism that is available. In a real pinball game, however, the player would likely go bust since the pinball machinery is programmed to tilt after a few strong movements of the whole physical machine. For the Atari environment however, the agent has learned to balance the use of the "nudging" and "flipping" well enough to effectively steer the ball without ever being penalised as "tilt" (which happens after "nudging" too frequently or tilting the pinball table to steeply resulting in loss of control over the game [1])

Notably, the overall strategy followed by the agent — first ensuring to secure an extra life per round to prolong the game, and then switching sides to gain an additional score increase in case the current ball is lost — shows highly strategic behavior, targeted at maximizing long term

[1]See game manual at `https://atariage.com/manual_html_page.php?SoftwareLabelID=588`

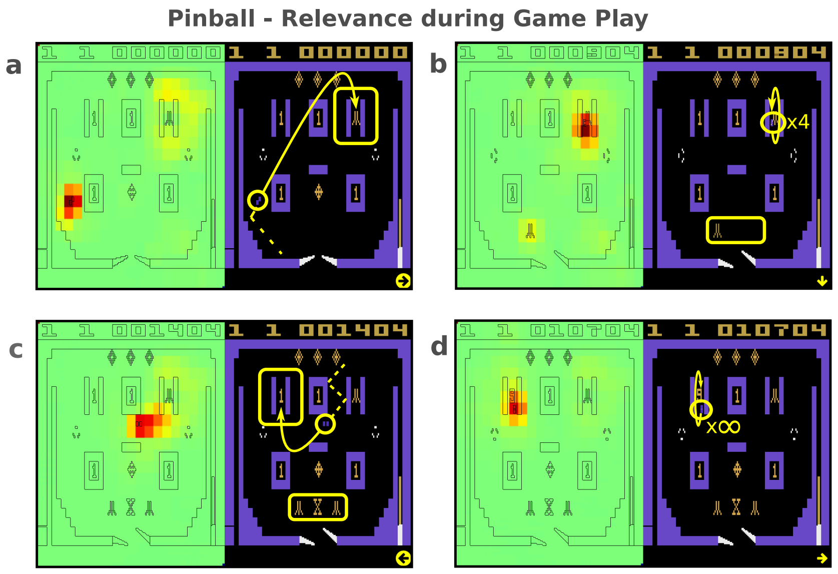


Figure 3.5: Pinball Strategy: **a)** Initially, the "pipe" structure at the top right of the pinball table holds a high concentration of relevance. The agent tracks the location of the ball and pipe and tries to move the ball there. **b)** Once the ball arrives, the agent uses "nudging" and collisions with other table elements to let the ball pass through the pipe exactly four times. For each time, a score increase is rewarded to the player and after the fourth passing, an extra ball is added to the player's reserve. The agent is apparently tracking its progress by focusing on the items appearing at the bottom of the screen. **c)** After the fourth change in icon appearance due to passing the top right pipe, the screen flashes white once and LRP shows that all relevance focus is removed from the bottom icons and accumulated on the top left pipe element instead. The agent uses the "nudging" strategy again to move the ball towards the left pipe. **d)** The agent now bounces the ball through the pipe and off the block below/the wall indefinitely. For each time, the agent is awarded with a score increase — identical to the reward earned by passing the right pipe. Passing the left pipe, however also increases a counter, which further increases a reward added to the current score as soon as the ball is lost.

reward. The agent has learned to stay within the rule boundaries of pinball tricking/cheating only to an extent that no tilt ends the game. Thus, the machine has found an optimal strategy to continuously increase the score at only minimal risk of hazardous ball movement, perhaps not the one targeted by the original game designers.

This behaviour can be measured by quantitatively evaluating the amount of relevance on the flippers. A "cheating" agent uses the flippers constantly while paying no attention to them, simply to prevent the game from tilting.

3.2.2 DNN Agent Gameplay in Breakout:

Our last example is a neural agent playing the playing the Atari game of Breakout [Mni+13].

The agent learns sophisticated behaviour, such as the purposeful construction of a tunnel through which they manoeuvre the ball. This allows the quickest scoring of points while minimising the risk to loose the ball. See fig. 3.6 for a walk through the strategy.

We want to use the LRP-relevance method to follow the development a deep neural network during the reinforcement learning process over many episodes of training. The heatmaps reveal conspicuous structural changes during the learning process. They allow us to track, when the agent learns to identify important game object even when they have not yet learned how to successfully play the game.

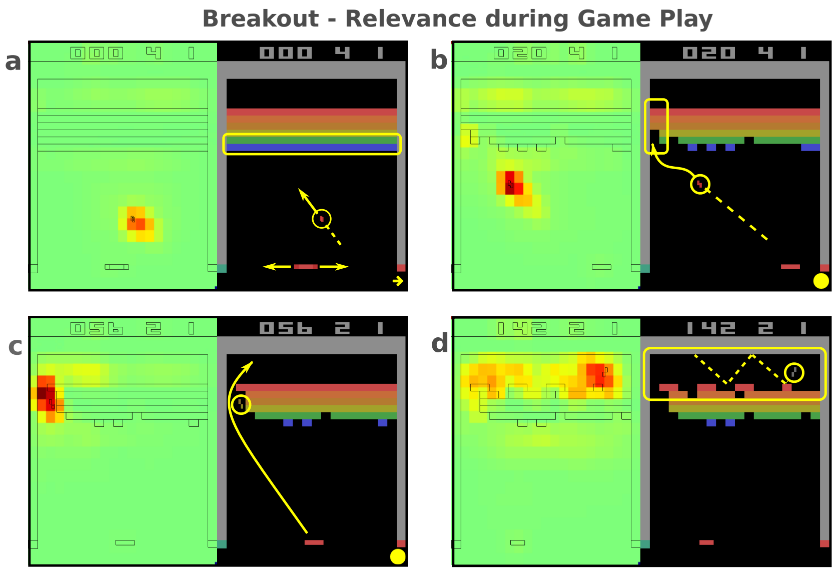


Figure 3.6: Breakout Strategy: **a)** In the beginning, the neural agent focuses on tracking the ball and following it with the paddle at the bottom of the screen. The relevance is predominantly focused on the ball. Only for the brief moment when the ball moves close to the bottom edge strong positive relevance is allocated on the paddle. During this game play stage, also a weak relevance is attributed to the tunnel area at the top left, where the model aims to shoot the ball. **b)** Each time the ball hits a block, the block disappears, and the score increases. When the ball hits an exposed block from the third row of blocks, the ball speeds up to twice its initial movement speed. The neural network agent recognises the increase in ball velocity and transitions into the second phase of its learned game play strategy. It focuses on targeting the ball towards the leftmost column of bricks, where it has learned to create a vertical tunnel through the coloured wall and assigns high relevance to the tunnel area. **c)** After finishing the tunnel, the model attempts to position the ball above the brick wall and keep it in this area, where a quick accumulation of score rewards is possible. **d)** A permanent cloud of strong positive relevance perseveres above the brick wall, including those times the ball is below the brick wall, clearly reveals the agent's strategy. This behaviour embodies an appropriate "understanding" of the game and its strategic targets.

Quantifying the Focus of the Deep Networks over Training During training of neural agents, different stages of behaviour can be observed. This is mirrored by our analysis of the networks focusing on different game objects. For our analysis, a neural network agent has been trained for 200 epochs to play *Atari Breakout*, with each epoch spanning 100,000 parameter update steps. At the end of each training epoch, a snapshot of the current model state is saved, resulting in 200 models of different levels of expertise at playing the game.

Using the average outside-inside relevance ratio μ, as defined as in eq. (3.1), we measure the attention that the network focuses on the following three different regions.

The ball: A rectangular region enclosing the ball with two pixels of padding in each direction. The relevance response is only measured if the ball is below the brick area, in order to distinguish the amount of relevance attributed to the ball from the amount of relevance score covering the bricks.

The paddle: A rectangular region around the paddle with two pixels of padding in each direction. The relevance response for the paddle is only measured when the measurement regions of ball and paddle do not overlap, to avoid confusion in the relevance allocation measurements taken.

The tunnel: A region defined by all the pixels in the leftmost and rightmost columns of the brick wall, which have been cleared.

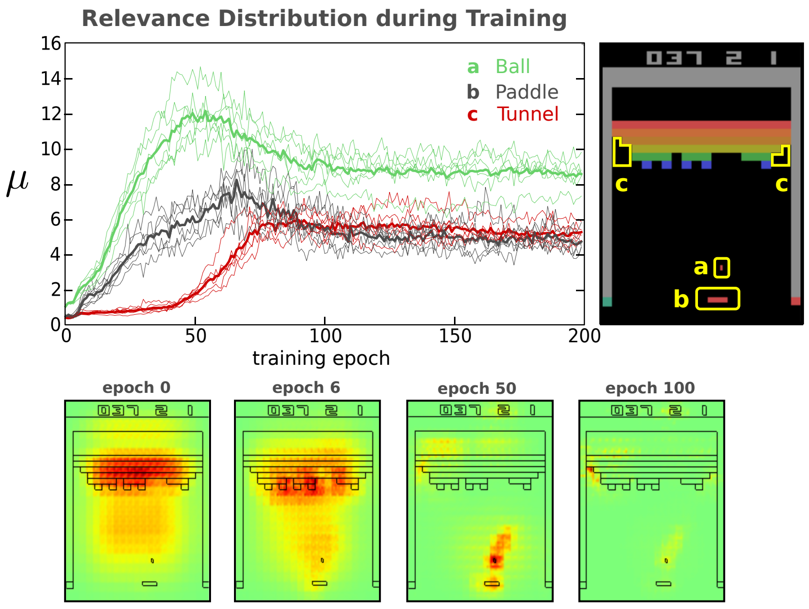


Figure 3.7: Development of the relative relevance of different game objects in Atari Breakout over the training time. The average outside-inside relevance ratio μ is defined as in eq. (3.1). **Thin lines:** μ over a single training run, **Thick lines:** Average μ over the 5 different runs. In the beginning, the relevance is spread over the whole input. It steeply rises on the region around the ball, the focus on the ball develops a bit later. After about 50 training episodes, the tunnel

To get a sample set of game states to be analysed, we let the fully trained network play a game of *Atari Breakout* for a series of 2000 frames, where every input state, Q-value and action passed to the game interface are recorded. From this set of 2000 frames, 500 frames from the early game phase and another 500 from the late game phase are taken and used as inputs for each of the 200 network training stages obtained earlier.

To reach comparability across networks, the same inputs are used for all the networks, instead of letting each network generate its own input by playing. Clearly, for a model reflecting an earlier state of training it is quite unlikely that complex strategies such as tunnel building could be observed, thus leaving the focus on the tunnel for this model undefined. With our protocol, we intend to see whether networks in earlier learning stages are already able to recognise the usefulness of a tunnel or not, although this knowledge would only be exploited in the strategy at a later stage.

For every model and input sample, relevance maps are computed with respect to the model decision. For a set of important game objects we measure the total amount of relevance allocated to their respective pixel regions. To minimise the influence of randomisation effects from the training process, the experiment is repeated over six different training runs (i.e. we have trained 6 networks in total, with 200 network snapshots created in regular intervals for each network). The individual network responses and the mean thereof is plotted in fig. 3.7.

We were able to identify 4 distinct phases during training

1. At the start, the network has not understood any game aspect. The relevance is diffused over the whole input.

2. 10 episodes in, the network recognises the ball as important. However, it cannot yet manoeuvre the paddle.

3. After 30 episodes, the network recognises both ball and paddle. It is able to prevent the ball from escaping and scores consistently.

4. After 70 episodes, the network focuses on the tunnel region on either side of the brick wall. It consistently pursues the tunnel building strategy.

Here, the network training progresses from initially just being able to recognise the ball and later the paddle with which the network learns to control the game, until finally acquiring long-term strategies as subsequent behavioural stages. Specifically, we are able to observe that the network recognises certain game objects reliably before it is even able to play the game rudimentarily well. We take that as an indicator that early in training the convolution filters have adapted to respond to the moving game elements such as the ball and paddle well, while the top layers of the network are not yet able to use this information to form an appropriate strategic response for the current state of the game, other than correlating ball movements to changes in the score. That is, the model has learned that the ball is relevant for the game. We note that analysis like this can help to investigate shortcomings of a network-based predictor, e.g. to distinguish between problems within the lower level architecture, responsible for tracking, from shortcomings in the top layers, responsible for devising a decision.

To complement the analysis presented here, we also investigated the influence of the network architecture, comparing neural network with different numbers of convolutional and fully connected layers, see fig. A.4 in appendix A. There, we see that the shallower architectures are unable to recognise the tunnel as a relevant region, which is reflected in game play. See also fig. A.5 for a comparison for different memory sizes, where a smaller memory results in a slower shift in tunnel building, and A.6 where we illustrate the change in relevance during game play.

3.2.2.1 Comparing Relevance Maps to Gradient Map

As explained in chapter 1, another popular method to analyse neural network type models or reinforcement learning systems is Sensitivity Analysis (cf. [Bae+10; SVZ13; ZBM16]), a gradient-based saliency method.

We repeat the experiment illustrated in fig. 3.7 with the gradient maps to compare both methods. As we will observe, LRP provides clearer results.

In fig. 3.8, we present the results obtained when using Sensitivity Analysis instead using LRP for the setup of fig. 3.7. Observing the gradient maps during game play reveals that the saliency follows the general area where the ball is located, but is *not* focused on the ball itself. The response obtained with Sensitivity Analysis, when given a model and an input data point, measures how much a change in the input would affect the model prediction. Here, peaking gradient values in close proximity to the ball essentially indicate where the ball should be to cause the steepest change in predicted Q-values, compared to where it actually is. Furthermore, for each of the neural network agents trained, gradient maps did not attribute weights on the pixels covering the tunnel element. The changes in game play strategy are not observable through Sensitivity Analysis.

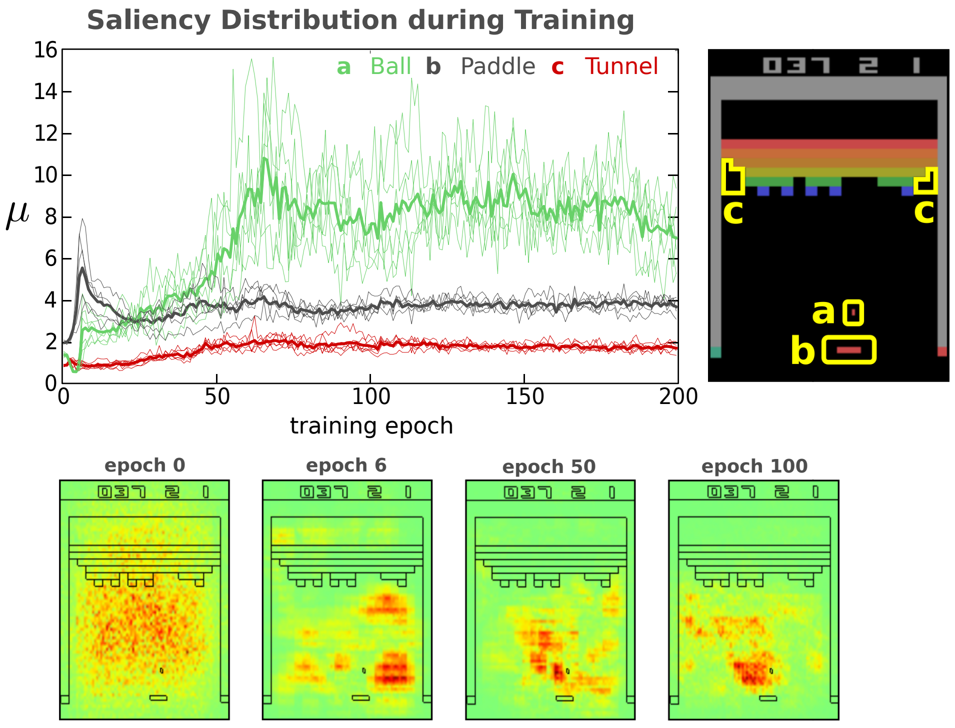


Figure 3.8: With the same neural network classifier trained to play *Atari Breakout* as used for fig. 3.7, gradient maps are computed using Sensitivity Analysis. We measure the relative saliency on different objects with ongoing model training, evaluated on a game sequence of 500 states. The plot shows the gradient magnitude allocation on the ball, paddle and tunnel elements for all training epochs for all six models. Bold lines show the average result over the 6 models. Gradient maps are less local and sparse compared to Relevance maps computed by LRP. The latter also align better with the game elements in pixel space. In the gradient map, more relevance is attributed to the region around the ball than the ball itself. The gradient map does not respond to the tunnel area.

3.3 Discussion

Our case studies confirm the usefulness of relevance methods for the interpretation of neural network classifiers and reinforcement agents. We were able to detect artefacts as well as biases in established image recognition data sets as well as observe “cheating” behaviour for the case of the Atari Pinball agent. Such artefactual learning has also been described in other work, e.g, in [RSG16b] where the model leaned to distinguish wolves and Huskies by the presence of snow, or in the context of reinforcement learning [AC16] where the RL agent finds an isolated lagoon where it can turn in a large circle and repeatedly knock over three targets, timing its movement so as to always knock over the targets just as they repopulate due to wrongly specifying the reward function. All these are concrete problems on AI safety, which has been recently studied in [Lei+17].

It would thus be reassuring to know that our relevance methods will always be able to detect such artefacts should they exist. Alas, so far, these methods rely on heuristics and neither define what their output actually signifies, nor provide guarantees as to when the output information will be reliable. To fully make use of explanation methods these shortcomings must be remedied—a task we tackle in the following chapters.

4 Probabilistic Prime Implicants

How can we rigorously define what we intuitively understand as relevant input? To answer this question we look at how relevance maps are generally evaluated numerically, e.g., using pixel-flipping [Sam+17] and input perturbation [FV17]. Though implemented in different ways, the principal idea is that the classification is less sensitive to perturbation of the less relevant inputs and highly sensitive to perturbation of the most relevant parts of the inputs. If these perturbations were to be infinitesimally small this would lead us back to the sensitivity analysis. For arbitrarily strong perturbation this would mean the following: The relevant parameters constitute a subset that if held constant the classifier decision remains the same—no matter what happens to the rest.

To simplify this problem, we only consider a binary partition into relevant and non-relevant variables (even though most existing methods provide continuous scores [SWM17]). The reason for this is two-fold. First, there is generally no agreed upon interpretation of what continuous relevance scores mean. Therefore, we prefer to keep the clear meaning of a partition of the variables. Secondly, many of the most prominent applications of these relevance mappings rely on binarisations of the continuous relevance scores. A mask for relevant objects in the input, e.g. tumor cells in body tissue [LH18] or expressive genes in a sequence [Vid+15], can be obtained by considering only the variables with a sufficiently large relevance score. The decision in the end is thus a binary one:

> *Q1:* Is there a small part of the input that determines the classification with high probability?

A more quantitative version of the question is the following.

> *Q2:* What is the smallest part of the input that determines the output with high probability?

We should require relevance maps to convey at least this much information.

These questions constitute a decision and a search problem respectively. We make this notion rigorous by introducing the concept of *probabilistic prime implicants.* They are a generalisation of prime implicants, a well known idea from abductive reasoning. In this chapter we will explain these concepts in their binary form, while the next chapter expands them to a continuous setting. As we will see finding or even approximating small prime implicants are hard problems that cannot be realised efficiently unless some well established complexity conjectures are falsified.

4.1 From Prime Implicants to δ-Relevant Sets

Prime implicants are a concept from Boolean logic that has been extended to abductive reasoning in first order logic [Mar91; Mar00]. Abduction is a form of logical reasoning similar to deduction and induction. It starts with an observation and aims seeks to the simplest cause that explains this observation. Like induction and unlike deduction, abductive conclusions cannot be proven, but rely on some criterion of plausibility like 'simpleness' of the explanation. We state here the basic Boolean definition of prime implicants, for the extension to first order logic see [Mar91].

Definition 4.1 Let Ψ be a propositional formula over a set of variables $\mathcal{X} = \{x_1, \ldots, x_d\}$. Let π be a set of literals from $\mathcal{X}$ interpreted as a conjunction. Then we call π an implicant of Ψ if

$$\pi \models \Psi,$$

and a prime implicant if additionally

$$\forall \pi' \subsetneq \pi : \pi' \not\models \Psi.$$

Let us give an example to make the idea clear.

Example 4.2 Let $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$, $\Psi := (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3)$ and $\pi_1 = \{x_2, x_3\}, \pi_2 = \{x_1, x_3\}, \pi_3 = \{x_1\}$.

Then π_1 is not an implicant, since it does not imply Ψ. On the other hand π_2 is an implicant, but it is no prime implicant since π_3 is a proper subset and also an implicant. In fact, π_3 is a prime implicant.

This concept had been extended to prime implicant explanations [SCD18] for binary classifier decisions by interpreting an assignment $\mathbf{x}$ to the variables as a set of literals.

Definition 4.3 Let $\Psi : \{0,1\}^d \to \{0,1\}$ be a binary classifier, and $\mathbf{x} \in \{0,1\}^d$ an assignment and $\mathbf{y} \in \{0,1\}^d$ a set of free variables. We call $\mathbf{x}_S$ with $S \in [d]$ an implicant explanation for Ψ and $\mathbf{x}$ if

$$\mathbf{y}_S = \mathbf{x}_S \;\models\; \Psi(\mathbf{y}) = \Psi(\mathbf{x}),$$

and a prime implicant explanation if additionally

$$\forall S' \subsetneq S : \; \mathbf{y}_{S'} = \mathbf{x}_{S'} \;\not\models\; \Psi(\mathbf{y}) = \Psi(\mathbf{x}).$$

Example 4.4 Let $\Psi := (x_1 \vee x_2) \wedge (x_3 \vee \neg x_4)$ and $\mathbf{x} = \{0, 0, 1, 1\}$, resulting in $\Psi(\mathbf{x}) = 0$. Choosing $S = \{1, 2\}$ for example, we get the implicant explanation $\mathbf{x}_S = \{0, 0\}$, since $\Psi = 0$ as soon as the first two variables are set to 0, no matter the assignment to the last two. Additionally $\mathbf{x}_S$ is also a prime implicant explanation, since omitting either assignment does ensure that $\Psi = 0$.

So in a nutshell, an implicant explanation is a subset of the input variables that is sufficient for the decision. In other words, keeping the implicant variables fixed will lead to the same classification for all possible completions of the remaining variables. A prime implicant explanation is a minimal implicant with respect to set inclusion and thus cannot be reduced further. The problem of finding small prime implicants is $\mathsf{NP}^{\mathsf{coNP}}$-hard [EG95], and practical algorithms rely on highly optimised SAT or MILP-solvers even for relatively low-dimensional cases [INM19].

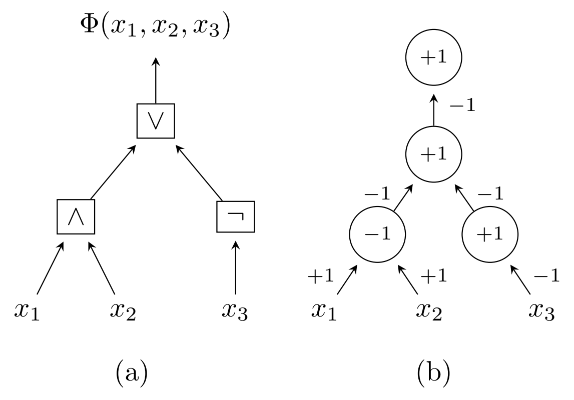
Prime implicant explanations can be seen as a type of explanation under worst-case conditions: the explaining set of variables is required to be sufficient for the function value to remain unchanged for *all* possible assignments to the other variables. We argue to relax this notion and allow the function value to change with a small probability over random assignments to the non-relevant variables. This has two main reasons.

First, a worst case analysis might be sensible for few variables, but it is too rigid for very high dimensional cases, such as modern image classification. In this case, often small regions of the input image can be manipulated in a way that completely changes the classifier prediction, e.g. through adversarial patches [Bro+17; Liu+18]— a numerical analogy to supernormal stimuli [Bar10] in animals. Thus, prime implicants will have to cover large portions of the input image, independent of the size of the actual object in the image that led to the original classifier prediction.

Secondly, practical heuristic algorithms for determining sets of important variables [FV17; RSG18; Kho+19] as well as methods to numerically evaluate and compare them [FV17; Sam+17; ZF14b], already implicitly rely on this relaxed probabilistic formulation of relevance. They estimate the expected change in the function value via random sampling of non-relevant variables— a task that is numerically feasible, compared to the computationally intractable requirement that for every of the exponentially many assignments the implicant property should hold.

Thus, practical interpretation algorithms necessarily need to solve the problem defined in the next section and are subject to the hardness results we are going to present. A rigorous analysis of this setting is long overdue and of high importance.

Let us now give a formal definition of our probabilistic notion of prime implicant explanations and state the two main results of this paper. A subset $S \subseteq [d]$ of variables is *relevant* for the function value $\Psi(\mathbf{x})$ if fixing $\mathbf{x}$ on S and randomising it on the complement S^c does not change the value of Ψ with high probability. The complement then consists of the *non-relevant* variables.



$$-\varrho\left(\begin{bmatrix}-1 & -1\end{bmatrix}\varrho\left(\begin{bmatrix}1 & 1 & 0\\ 0 & 0 & -1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}+\begin{bmatrix}-1\\ 1\end{bmatrix}\right)+1\right)+1$$

(c)

Figure 4.1: The Boolean function $\Psi(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (\neg x_3)$ viewed as a Boolean circuit (a) and a rectified linear unit (ReLU) neural network in its graphical (b) and algebraic representation (c). The neural network weights and biases are denoted at the edges and nodes respectively. The ReLU activation $\varrho(x) = \max\{x, 0\}$ is applied components-wise.

Definition 4.5 Let $\Psi\colon \{0,1\}^d \to \{0,1\}$, $\mathbf{x} \in \{0,1\}^d$, and $\delta \in [0,1]$. We call $S \subseteq [d]$ a δ-*relevant* set for Ψ and $\mathbf{x}$, if

$$P_{\mathbf{y}}(\Psi(\mathbf{y}) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S) \geq \delta.$$

Remark 4.6 The same concept had been previously introduced as *precision* of the set S [RSG18; Nar+19]. We were unaware of this convention and will continue to use δ-relevance throughout this work. The nomenclature of precision will return in chapter 8, where we consider the average precision over a whole data set.

We hark back to example 4.4 to illustrate this concept.

Example 4.7 Let again $\Psi := (x_1 \vee x_2) \wedge (x_3 \vee \neg x_4)$ and $\mathbf{x} = \{0, 0, 1, 1\}$, resulting in $\Psi(\mathbf{x}) = 0$. Now, choosing $S = \{1\}$ we get a $\frac{3}{4}$-relevant set, since the probability of the second clause to be true is $1 - \left(\frac{1}{2}\right)^2 = \frac{3}{4}$.

For δ close to one this means that the input $\mathbf{x}$ supported on S already determines the output $\Psi(\mathbf{x})$ with high probability. It is clear that $S = [d]$ is always 1-relevant, and any subset $S \subseteq [d]$ is at least 2^{-d}-relevant. Now, the question arises whether for a given δ there exists a δ-relevant set of a certain size. Similarly, one could ask to find the smallest δ-relevant set. This set would then be composed of the most important variables for the function value $\Psi(\mathbf{x})$. This introduces a trade-off since a larger δ will generally require a larger set S.

4.2 Complexity Analysis

Algorithmic problem solving in real-world scenarios often requires reasoning in an uncertain environment. This necessity leads to the investigation of probabilistic satisfiability problems and probabilistic computational complexity classes such as PP and $\mathsf{NP}^{\mathsf{PP}}$. One prototypical

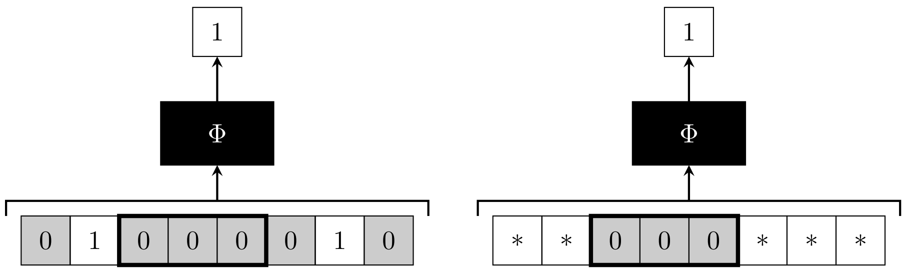


Figure 4.2: The Boolean function $\Psi\colon \{0,1\}^8 \to \{0,1\}$ decides if a binary input string contains a substring of three consecutive zeros. A relevant subset of input variables for an exemplary input is highlighted by a box. In this simple case the three consecutive zeros are relevant, because it is sufficient to know them to predict the decision made by Ψ, independent of all other input variables. Note, that in this example the relevant set is not unique, as there are two sets of three consecutive zeros.

example, the E-Maj-Sat problem [LGM98; LMP01], is an extension of the classical satisfiability problem that includes an element of model counting. The class of $\mathsf{NP}^{\mathsf{PP}}$-complete problems contains many relevant artificial intelligence (AI) problems such as probabilistic conformant planning [LGM98], calculating maximum expected utility (MEU) solutions [CJ08], and maximum a posteriori (MAP) hypotheses [Par02].

We connect these probabilistic reasoning tasks to our problem of interpreting the decisions of neural network classifiers by showing that they can be mapped onto each other in polynomial time. In general, since the classifier function is fixed within each problem instance, such complexity results carry over to all types of classifiers that are able to efficiently represent Boolean circuits. A prominent example are ReLU-neural networks with weights and biases in $\{-1,0,1\}$. They can emulate Boolean circuits of comparable width and depth [MB17; Par96], as illustrated in fig. 4.1.

Definition 4.8 For $\delta \in (0,1]$ we define the δ-Relevant-Input problem as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^d \to \{0,1\}$, $\mathbf{x} \in \{0,1\}^d$, and $k \in \mathbb{N}$, $1 \leq k \leq d$.

Decide: Does there exist $S \subseteq [d]$ with $|S| \leq k$ such that S is δ-relevant for Ψ and $\mathbf{x}$?

Note that in our formulation δ is not a part of the problem instance, but instead each choice of δ represents a problem class, similar to k-SAT. We show that the problem is hard for any fixed δ. The minimisation formulation of the above decision problem can be defined in the obvious way.

Definition 4.9 For $\delta \in (0,1]$ we define the Min-δ-Relevant-Input problem as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^d \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^d$.

Task: Find the minimal $k \in \mathbb{N}$ such that there exists $S \subseteq [d]$ with $|S| \leq k$ and S is δ-relevant for Ψ and $\mathbf{x}$.

The majority of the remainder of the paper will deal with analysing the computational complexity of δ-Relevant-Input, Min-δ-Relevant-Input, and related variants thereof.

Our first main contribution shows that the δ-Relevant-Input problem is generally hard to solve.

Theorem 4.10 *For $\delta \in (0,1)$ the δ-Relevant-Input problem is $\mathsf{NP}^{\mathsf{PP}}$-complete.*

Intuitively, the NP-part of the problem complexity arises from the necessity to check all subsets $S \subseteq [d]$ as possible candidates for being δ-relevant. The PP-part of the complexity arises from the fact that for any given set S checking if it is δ-relevant is by itself a hard (in fact PP-hard)[1] problem. The problem class $\mathsf{NP}^{\mathsf{PP}}$ is beyond the scope of conventional computing. In particular, Min-δ-Relevant-Input is at least as hard to solve as the corresponding decision problem, which makes it unfeasible to solve exactly. However, in applications it is rarely required to exactly find the smallest relevant set. It would be desirable to obtain good approximate solutions within feasible computational complexity.

We present two potential ways for simplifying the problem by allowing approximations: First, we relax the requirement that a solutions set has to be exactly δ-relevant. Secondly, we allow an approximation of the the minimal relevant set in terms of its size. The former would address the PP part whereas the latter would address the NP aspect.

Calculating probabilities or expectation values may be hard in theory, yet it is often easy to calculate them (approximately) in practice, e.g. by sampling. Checking whether a logical proposition is satisfied with probability more than δ by sampling only fails if the true probability can be arbitrarily close to δ both from above and below. These edge cases cause the hardness of the problem, but in our scenario we do not necessarily care about their resolution. We make this notion formal by stating a promise version of our problem where, if the true probability is smaller than δ, it will be smaller by at least γ with $0 \leq \gamma < \delta$. We refer to this as the γ-Gapped-δ-Relevant-Input problem and it is formally defined in section 4.4. We will see that for positive γ this reduces the problem complexity from $\mathsf{NP}^{\mathsf{PP}}$ to $\mathsf{NP}^{\mathsf{BPP}}$. The associated optimisation problem is called Min-γ-Gapped-δ-Relevant-Input and made formal in the same section. Unfortunately, even in this simplified case, it remains NP-hard to approximate the size of the optimal set S within any reasonable approximation factor.

Theorem 4.11 *Let $\delta \in (0,1)$ and $\gamma \in [0,\delta)$. Then, for any $\alpha \in (0,1)$ there is no polynomial time approximation algorithm for Min-γ-Gapped-δ-Relevant-Input with an approximation factor of $d^{1-\alpha}$ unless $\mathsf{P} = \mathsf{NP}$.*

This amounts to the following insight:

> *Any efficient algorithm cannot reliably answer* Q1 *or approximately answer* Q2 *within any non-trivial approximation factor unless* $\mathsf{P} = \mathsf{NP}$.

The complete proofs of both main theorems as well formal definitions, detailed discussions, and analyses of the problem variants are given in section 4.3 and section 4.4. Already here, we can draw an important corollary which follows from theorem 4.11 for the special case $\gamma = 0$.

[1]Checking if a subset is one-relevant is in coNP instead of PP. Thus, we excluded $\delta = 1$ in theorem 4.10.

Corollary 4.12 Let $\delta \in (0, 1)$. Then, for any $\alpha \in (0, 1)$ there is no polynomial time approximation algorithm for MIN-δ-RELEVANT-INPUT with an approximation factor of $d^{1-\alpha}$ unless $\mathsf{P} = \mathsf{NP}$.

4.2.1 Related Approaches

This work is not the first in the direction of explaining binary classifiers through a probabilistic setup. We will look at the following concepts, how they are related and whether our hardness results can carry over.

Sufficient Explanations The authors in [Kho+19] introduced sufficient explanations for binary decision functions obtained from thresholding a continuous prediction model, e.g. a logistic regression classifier. As in our approach, the authors consider a probabilistic version of the prime implicant problem. In this case, the classification decision is required to remain unchanged in expectation instead of for all possible assignments to the non-fixed variables. More precisely, let $f\colon \mathcal{X} \to [0, 1]$ be a continuous prediction model on a domain $\mathcal{X}$ (e.g. a logistic regression model), $\theta\colon [0, 1] \to \{0, 1\}$ be a binarisation function (e.g. thresholding at 0.5), and $\mathcal{D}$ be a distribution on $\mathcal{X}$. A variable x_i of an input $\mathbf{x} \in \mathcal{X}$ is called a supporting variable, if

$$\begin{cases} \mathbb{E}_{\mathbf{y}\sim\mathcal{D}}\Big(f(\mathbf{y})\,\Big|\,\mathbf{y}_{\{i\}^c} = \mathbf{x}_{\{i\}^c}\Big) \leq f(\mathbf{x}) & \text{if } \theta(f(\mathbf{x})) = 1, \\ \mathbb{E}_{\mathbf{y}\sim\mathcal{D}}\Big(f(\mathbf{y})\,\Big|\,\mathbf{y}_{\{i\}^c} = \mathbf{x}_{\{i\}^c}\Big) > f(\mathbf{x}) & \text{if } \theta(f(\mathbf{x})) = 0. \end{cases}$$

In other words, randomising x_i conditioned on fixing all other variables does not increase the classification margin in expectation. A sufficient explanation is a cardinality minimal subset S of all supporting variables satisfying

$$\theta(\mathbb{E}_{\mathbf{y}\sim\mathcal{D}}(f(\mathbf{y})\,|\,\mathbf{y}_S = \mathbf{x}_S)) = \theta(f(\mathbf{x})).$$

For an already binary function f and $\mathcal{D} = \mathcal{U}\Big(\{0, 1\}^d\Big)$, this approach is essentially the same as finding small $\frac{1}{2}$-relevant sets. The only difference is that sufficient explanations only consider subsets of supporting variables, while we make no such distinction. This is however a minor difference and we conjecture that our hardness results carry over.

Anchors Anchors were introduced recently by [RSG18] as local model-agnostic explanations. Given a generic function $f\colon \mathcal{X} \to \mathcal{Z}$ from a domain $\mathcal{X}$ to a codomain $\mathcal{Z}$ (for example a set of class labels) and a threshold $\delta \in [0, 1]$, an anchor for an input $\mathbf{x} \in \mathcal{X}$ is some predicate $A\colon \mathcal{X} \to \{0, 1\}$ satisfying

$$A(\mathbf{x}) = 1 \qquad \text{and} \qquad P_{\mathbf{y}\sim D_{\mathbf{x}}}(f(\mathbf{y}) = f(\mathbf{x})\,|\,A(\mathbf{y})) \geq \delta,$$

where $D_{\mathbf{x}}$ is a local distribution in the neighbourhood of $\mathbf{x}$. The description of feasible predicates A is rather vague, however the predicates explicitly considered by [RSG18] are of the form

$$A(\mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{y}_S = \mathbf{x}_S, \\ 0 & \text{otherwise,} \end{cases}$$

for some subset S of the variables in $\mathcal{X}$, just as in our formulation. Choosing the domain $\mathcal{X} = \{0,1\}^d$ and codomain $\mathcal{Y} = \{0,1\}$, the only difference to our δ-Relevant-Input problem is that we consider a global uniform distribution instead of local perturbations around $\mathbf{x}$.In [RSG18] it was suggested to search for an anchor with the largest possible coverage, defined as $\mathrm{cov}(A) = P_{\mathbf{y}\sim D_{\mathbf{x}}}(A(\mathbf{y}))$. For the uniform distribution this is exactly equivalent to searching for the smallest set S. We conjecture that our hardness results carry over to the problem of finding anchors for many possible perturbation distributions $D_{\mathbf{x}}$.

Shapley Values Let us recall the definition of Shapley values from section 1.2. For a characteristic function $\nu\colon 2^{[d]} \to \mathbb{R}$ the Shapley value of the i-th variable (i-th player) is defined as

$$\varphi_{i,\nu} = \sum_{S\subseteq[d]\setminus\{i\}} \frac{|S|!(d-|S|-1)!}{d!}(\nu(S\cup\{i\}) - \nu(S)).$$

In our scenario the value of a subset of variables S can be measured by the expected difference in Ψ when fixing variables in S and randomising the remaining variables. In [Kon+10] the authors proposed to use

$$\nu(S) = \frac{1}{2^{d-|S|}} \sum_{\substack{\mathbf{y}\in\{0,1\}^d \\ \mathbf{y}_S=\mathbf{x}_S}} \Psi(\mathbf{y}) - \mathbb{E}_{\mathbf{y}}(\Psi(\mathbf{y}))$$

for the analysis of classifier decisions, which uses the expectation of the completely randomised classifier score as a reference value to determine the coalition value. We observe that

$$\begin{aligned} P_{\mathbf{y}}(\Psi(\mathbf{y}) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S) &= 1 - \frac{1}{2^{d-|S|}} \sum_{\substack{\mathbf{y}\in\{0,1\}^d \\ \mathbf{y}_S=\mathbf{x}_S}} |\Psi(\mathbf{y}) - \Psi(\mathbf{x})| \\ &= 1 - |\nu(S) + c| \end{aligned}$$

with $c = \mathbb{E}_{\mathbf{y}}(\Psi(\mathbf{y})) - \Psi(\mathbf{x})$, hence $S \subseteq [d]$ is δ-relevant for Ψ and $\mathbf{x}$ exactly if $|\nu(S)+c| \leq 1-\delta$.

Despite this relation between δ-relevant sets and the characteristic function ν our problem formulation is considerably different from the Shapley value approach. The task considered in this paper is not to distribute the value of coalitions amongst the variables but to find (small) coalitions that are guaranteed to have a certain value.

4.3 Proof of $\mathsf{NP}^{\mathsf{PP}}$-Completeness

In the last section we stated our completeness and inapproximability results for the δ-relevant input problem and its γ-gapped version. This section presents the proof of theorem 4.10 which will be split into two parts. We will show that δ-Relevant-Input is $\mathsf{NP}^{\mathsf{PP}}$-hard in section 4.3.1 and that it is contained in $\mathsf{NP}^{\mathsf{PP}}$ in section 4.3.2.

4.3.1 δ-Relevant-Input is $\mathsf{NP}^{\mathsf{PP}}$-hard

We now give the first part of the proof of theorem 4.10. This is done by constructing a polynomial-time reduction of a $\mathsf{NP}^{\mathsf{PP}}$-complete problem to δ-Relevant-Input. The canonical complete problem for $\mathsf{NP}^{\mathsf{PP}}$ is E-Maj-Sat [LGM98].

Definition 4.13 The E-Maj-Sat problem is defined as follows.

Given: A Boolean function $\Psi\colon \{0,1\}^d \to \{0,1\}$ in conjunctive normal form (CNF) and $k \in \mathbb{N}$, $1 \le k \le d$.

Decide: Does there exist $\mathbf{x} \in \{0,1\}^k$ such that $P_{\mathbf{y}}\left(\Psi(\mathbf{y}) \,\middle|\, \mathbf{y}_{[k]} = \mathbf{x}\right) > \frac{1}{2}$?

In other words, E-Maj-Sat asks whether there is an assignment to the first k variables of Ψ such that the majority of assignments to the remaining $d-k$ variables satisfies Ψ. There are three hurdles to take if we want to reduce this to δ-Relevant-Input.

1. Instead of freely assigning values to a subset of variables we are given an assignment to all variables and can only choose which to fix and which to randomise.
2. Instead of assigning values to a given set of k variables we can freely choose the set S of size at most k.
3. Instead of checking whether the majority of assignments satisfies Ψ we check if the fraction of satisfying assignments is greater than or equal to some δ.

We address each of these hurdles and give a chain of polynomial-time reductions

$$\text{E-Maj-Sat} \preceq_p \text{IP1} \preceq_p \text{IP2} \preceq_p \delta\text{-Relevant-Input} \tag{4.1}$$

in three steps with intermediate auxiliary problems IP1 and IP2. The following observations will turn out to be useful.

Remark 4.14 Let Ψ and Γ be Boolean functions, not necessarily of different variables. Then,

$$\begin{aligned} P(\Gamma) = 0 \quad &\Rightarrow \quad P(\Psi \oplus \Gamma) = P(\Psi), \\ P(\Gamma) = 1 \quad &\Rightarrow \quad P(\Psi \oplus \Gamma) = 1 - P(\Psi), \end{aligned}$$

and if Ψ and Γ are independent, i.e. $P(\Psi \wedge \Gamma) = P(\Psi)P(\Gamma)$, also

$$P(\Gamma) = \frac{1}{2} \quad \Rightarrow \quad P(\Psi \oplus \Gamma) = \frac{1}{2}.$$

Lemma 4.15 We consider two Boolean functions $\mathrm{EQ}\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ and $\Gamma\colon \{0,1\}^k \times \{0,1\}^k \times \{0,1\} \to \{0,1\}$ be defined as

$$\mathrm{EQ}(\mathbf{u},\mathbf{v}) = \bigwedge_{i=1}^{k} \neg(u_i \oplus v_i) \qquad \text{and} \qquad \Gamma(\mathbf{u},\mathbf{v},t) = \left(\bigvee_{i=1}^{k} (u_i \oplus v_i)\right) \wedge t.$$

Then, for any $\Psi\colon \{0,1\}^k \times \{0,1\}^{d-k} \to \{0,1\}$ and $A\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ with

$$P(A(\mathbf{u},\mathbf{v}) \wedge \mathrm{EQ}(\mathbf{u},\mathbf{v})) > 0 \quad \text{and} \quad P(A(\mathbf{u},\mathbf{v}) \wedge \neg\,\mathrm{EQ}(\mathbf{u},\mathbf{v})) > 0,$$

we have

$$P(\Psi(\mathbf{u},\mathbf{r}) \oplus \Gamma(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v})) > \frac{1}{2} \quad \Longleftrightarrow \quad P(\Psi(\mathbf{u},\mathbf{r}) \,|\, A(\mathbf{u},\mathbf{v}), \mathrm{EQ}(\mathbf{u},\mathbf{v})) > \frac{1}{2}.$$

The condition EQ determines whether $\mathbf{u} = \mathbf{v}$ or not. As soon as there exists an i with $u_i \neq v_i$, $\Gamma(\mathbf{u},\mathbf{v},t)$ has the value of t, which is 1 with probability $\frac{1}{2}$. That means by modulo-adding Γ to Ψ we only have to consider the cases where $\mathbf{u} = \mathbf{v}$ to decide whether the majority of assignments to Ψ evaluates to true. This is independent from any additional condition $A(\mathbf{u},\mathbf{v})$.

Proof. We can rewrite $\Gamma(\mathbf{u},\mathbf{v},t) = (\neg\,\mathrm{EQ}(\mathbf{u},\mathbf{v})) \wedge t$ and therefore

$$P(\Gamma \,|\, \mathrm{EQ}) = 0 \qquad \text{and} \qquad P(\Gamma \,|\, \neg\,\mathrm{EQ}) = \frac{1}{2}.$$

Since $\Psi \,|\, A$ and $\Gamma \,|\, A$ are conditionally independent given $\neg\,\mathrm{EQ}$ (in this case Γ depends on t only), we obtain from remark 4.14 that

$$P(\Psi \oplus \Gamma \,|\, A, \mathrm{EQ}) = P(\Psi \,|\, A, \mathrm{EQ}) \qquad \text{and} \qquad P(\Psi \oplus \Gamma \,|\, A, \neg\,\mathrm{EQ}) = \frac{1}{2}.$$

Therefore,

$$\begin{aligned}P(\Psi \oplus \Gamma \mid A) &= P(\Psi \oplus \Gamma \mid A, \mathrm{EQ})P(\mathrm{EQ}) + P(\Psi \oplus \Gamma \mid A, \neg\,\mathrm{EQ})P(\neg\,\mathrm{EQ}) \\ &= P(\Psi \mid A, \mathrm{EQ})P(\mathrm{EQ}) + \frac{1}{2}(1 - P(\mathrm{EQ})) \\ &= \frac{1}{2} + \left(P(\Psi \mid A, \mathrm{EQ}) - \frac{1}{2}\right)P(\mathrm{EQ}).\end{aligned}$$

This directly implies $P(\Psi \oplus \Gamma \mid A) > \frac{1}{2}$ if and only if $P(\Psi \mid A, \mathrm{EQ}) > \frac{1}{2}$. □

4.3.1.1 Fixing or Randomising Variables

Let us now come to the first step of the reductive chain (4.1). In this, we translate the possibility of freely assigning the first k variables into the choice of fixing or randomising variables from a given assignment. This choice is however still restricted to the first k variables.

Definition 4.16 We define the Intermediate Problem 1 (IP1) as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^d \to \{0,1\}$, $\mathbf{x} \in \{0,1\}^d$ and $k \in \mathbb{N}$, $1 \le k \le d$.

Decide: Does there exist $S \subseteq [k]$ such that $P_{\mathbf{y}}(\Psi(\mathbf{y}) \mid \mathbf{y}_S = \mathbf{x}_S) > \frac{1}{2}$?

In other words, IP1 asks the questions whether there exists a subset of the first k variables of Ψ such that fixing these to the values given by $\mathbf{x}$ implies that the majority of assignments to the remaining variables satisfies Ψ.

Lemma 4.17 E-Maj-Sat $\preceq_p$ IP1, in particular IP1 is $\mathsf{NP}^{\mathsf{PP}}$-hard.

Proof. Let $\{\Psi, k\}$ be an E-Maj-Sat instance. We will construct $\{\Psi', \mathbf{x}', k'\}$ that is a *Yes*-instance for IP1 if and only if $\{\Psi, k\}$ is a *Yes*-instance for E-Maj-Sat. For convenience we split the d variables of Ψ into the first k variables and the remaining $d-k$ variables and denote this $\Psi(\mathbf{x}) = \Psi(\mathbf{u}, \mathbf{r})$. The main idea is to duplicate the first k variables and choose $\mathbf{x}'$ in such a way that fixing the original variables or their duplicates corresponds to assigning zeros or ones in the E-Maj-Sat instance respectively. More precisely, we define

- $\Psi'\colon \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{d-k} \times \{0,1\} \to \{0,1\}$ as

$$\Psi'(\mathbf{u}, \mathbf{v}, \mathbf{r}, t) = \Psi(\mathbf{u}, \mathbf{r}) \oplus \left(\bigvee_{i=1}^{k} (u_i \oplus v_i) \wedge t\right),$$

- $\mathbf{x}' = (\mathbf{0}_k, \mathbf{1}_k, \mathbf{0}_{d-k}, 0) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{d-k} \times \{0,1\}$,
- $k' = 2k$.

This is a polynomial time construction. With Γ defined as in lemma 4.15 we can rewrite $\Psi'(\mathbf{u}, \mathbf{v}, \mathbf{r}, t) = \Psi(\mathbf{u}, \mathbf{r}) \oplus \Gamma(\mathbf{u}, \mathbf{v}, t)$.

Necessity: Assume that $\{\Psi, k\}$ is a *Yes*-instance for E-MAJ-SAT. Then, there exists an assignment $\mathbf{u}^* \in \{0,1\}^k$ to the first k variables of Ψ such that $P_{\mathbf{r}}(\Psi(\mathbf{u}^*, \mathbf{r})) > \frac{1}{2}$. Now, choose $S' = \{\, i \in \{1, \ldots, k\} \,:\, u_i^* = 0 \,\} \cup \left\{\, i \in \{k+1, \ldots, 2k\} \,:\, u_{i-k}^* = 1 \,\right\} \subseteq [k'] = [2k]$. Let $A\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ be given by

$$A(\mathbf{u}, \mathbf{v}) = \left(\bigwedge_{i \in S' \cap \{1, \ldots, k\}} \neg u_i \right) \wedge \left(\bigwedge_{i \in S' \cap \{k+1, \ldots, 2k\}} v_{i-k} \right)$$

and EQ as in lemma 4.15. Note that A depends on S' and thus implicitly on $\mathbf{u}^*$. In fact, $A(\mathbf{u}, \mathbf{v}) = 1$ holds if and only if both $u_i^* = 0$ implies $u_i = 0$ and $u_i^* = 1$ implies $v_i = 1$ for all $i \in [k]$. In particular, we have $A(\mathbf{u}, \mathbf{u}) = 1$ if an only if $\mathbf{u} = \mathbf{u}^*$. Also $\mathrm{EQ}(\mathbf{u}, \mathbf{v}) = 1$ if and only if $\mathbf{u} = \mathbf{v}$. Thus, we have

$$\begin{aligned} P_{\mathbf{r}}(\Psi(\mathbf{u}^*, \mathbf{r})) &= P(\Psi(\mathbf{u}, \mathbf{r}) \,|\, \mathbf{u} = \mathbf{u}^*) \\ &= P(\Psi(\mathbf{u}, \mathbf{r}) \,|\, A(\mathbf{u}, \mathbf{u})) \\ &= P(\Psi(\mathbf{u}, \mathbf{r}) \,|\, A(\mathbf{u}, \mathbf{v}), \mathrm{EQ}(\mathbf{u}, \mathbf{v})), \end{aligned}$$

and by the choice of $\mathbf{x}'$, A, and S' we get

$$\begin{aligned} P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) &= P(\Psi'(\mathbf{u}, \mathbf{v}, \mathbf{r}, t) \,|\, A(\mathbf{u}, \mathbf{v})) \\ &= P(\Psi(\mathbf{u}, \mathbf{r}) \oplus \Gamma(\mathbf{u}, \mathbf{v}, t) \,|\, A(\mathbf{u}, \mathbf{v})). \end{aligned}$$

We use lemma 4.15 together with $P_{\mathbf{r}}(\Psi(\mathbf{u}^*, \mathbf{r})) > \frac{1}{2}$ to conclude $P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) > \frac{1}{2}$ which shows that $\{\Psi', \mathbf{x}', k'\}$ is a *Yes*-instance for IP1.

Sufficiency: Now, conversely, assume that $\{\Psi', \mathbf{x}', k'\}$ is a *Yes*-instance for IP1. Then, there exists $S' \subseteq [k'] = [2k]$ such that $P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) > \frac{1}{2}$. Following the same grouping of variables as before, we write $\mathbf{x}' = (\mathbf{u}', \mathbf{v}', \mathbf{r}', t')$. We can translate this into a satisfying assignment $\mathbf{u}^*$ for E-MAJ-SAT where $u_i^* = 0$ when $i \in S$ and $u_i^* = 1$ when $i + k \in S'$. For that, we need two statements to be true. First, not both i and $i + k$ can be in S'. And second, if neither i nor $i + k$ are in S', then there is always the possibility of adding one of them to S' and still satisfy $P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) > \frac{1}{2}$.

To prove the first statement, assume towards a contradiction that there exists an $i \in [k]$ with $i \in S'$ and $i + k \in S'$. Since $\mathbf{u}' = \mathbf{0}_k$ and $\mathbf{v}' = \mathbf{1}_k$, we have that $(\mathbf{u}, \mathbf{v})_{S'} = (\mathbf{u}', \mathbf{v}')_{S'}$ implies $u_i = 0 \neq 1 = v_i$ and hence

$$P_{\mathbf{u},\mathbf{v},t}(\Gamma(\mathbf{u}, \mathbf{v}, t) \,|\, (\mathbf{u}, \mathbf{v})_{S'} = (\mathbf{u}', \mathbf{v}')_{S'}) = P_{\mathbf{u},\mathbf{v},t}(t \,|\, (\mathbf{u}, \mathbf{v})_{S'} = (\mathbf{u}', \mathbf{v}')_{S'}) = \frac{1}{2}.$$

Thus, remark 4.14 would imply $P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) = \frac{1}{2}$, which contradicts the assumption that $\{\Psi', \mathbf{x}', k'\}$ is a *Yes*-instance for IP1.

For the second statement, assume there exists an $i \in [k]$ with neither $i \in S'$ nor $i + k \in S'$. Then $A(\mathbf{u}, \mathbf{v})$ is a condition on $\mathbf{u}$ and $\mathbf{v}$ that does not include the variables u_i and v_i. Therefore,

$$\begin{aligned}
P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v})) = {} & \frac{1}{4} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 0, v_i = 0) \\
& + \frac{1}{4} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 0, v_i = 1) \\
& + \frac{1}{4} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 1, v_i = 0) \\
& + \frac{1}{4} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 1, v_i = 1).
\end{aligned}$$

For the second and third summand we have $u_i \neq v_i$, thus using remark 4.14 again, we get

$$P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 0, v_i = 1) = \frac{1}{2} = P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 1, v_i = 0),$$

and obtain

$$\begin{aligned}
P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v})) = {} & \frac{1}{4} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 0, v_i = 0) \\
& + \frac{1}{4} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 0, v_i = 1) \\
& + \frac{1}{4} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 0, v_i = 1) \\
& + \frac{1}{4} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 1, v_i = 1) \\
= {} & \frac{1}{2} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), u_i = 0) \\
& + \frac{1}{2} P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v}), v_i = 1).
\end{aligned}$$

Altogether, if $P_{\mathbf{u},\mathbf{v},t}(\Psi'(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v})) > \frac{1}{2}$, then at least one of the additional conditions $u_i = 0$ or $v_i = 1$ must also yield a probability greater than $\frac{1}{2}$. This implies that if $P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) > \frac{1}{2}$ then either i or $i + k$ can be added to S' while keeping the probability greater than $\frac{1}{2}$.

So, without loss of generality, we can assume that for each $i \in [k]$ exactly one of the cases $i \in S'$ or $i + k \in S'$ occurs. Then, we can define $\mathbf{u}^* \in \{0,1\}^k$ as $u_i^* = 0$ if $i \in S'$ and $u_i^* = 1$ otherwise. We observe that S' and $\mathbf{u}^*$ are exactly as in the previous step and the rest of the proof follows analogously. Again we use lemma 4.15 and conclude from $P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) > \frac{1}{2}$ that $P_{\mathbf{r}}(\Psi(\mathbf{u}^*, \mathbf{r})) > \frac{1}{2}$. This shows that $\{\Psi, k\}$ is a *Yes*-instance for E-MAJ-SAT. □

4.3.1.2 Allowing for all variables to be chosen

We continue with the second step of the reductive chain (4.1). Instead of choosing from the first k variables we are free to chose from all d variables but at most k many.

Definition 4.18 We define the INTERMEDIATE PROBLEM 2 (IP2) as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^d \to \{0,1\}$, $\mathbf{x} \in \{0,1\}^d$ and $k \in \mathbb{N}$, $1 \le k \le d$.

Decide: Does there exist $S \subseteq [d]$ with $|S| \le k$ such that $P_{\mathbf{y}}(\Psi(\mathbf{y}) \,|\, \mathbf{y}_S = \mathbf{x}_S) > \frac{1}{2}$?

In other words, IP2 asks the question whether there exists a subset of at most k variables of Ψ such that fixing these to the values given by $\mathbf{x}$ implies that the majority of the possible assignments to the remaining variables satisfies Ψ.

Lemma 4.19 We have IP1 $\preceq_p$ IP2. In particular, IP2 is $\mathsf{NP}^{\mathsf{PP}}$-hard.

Proof. Let $\{\Psi, \mathbf{x}, k\}$ be an IP1 instance. We will construct $\{\Psi', \mathbf{x}', k'\}$ that is a *Yes*-instance for IP2 if and only if $\{\Psi, \mathbf{x}, k\}$ is a *Yes*-instance for IP1. For convenience, we split the d variables of Ψ into the first k variables and the remaining $d-k$ variables and denote this $\Psi(\mathbf{x}) = \Psi(\mathbf{u}, \mathbf{r})$. The main idea is to extend Ψ with clauses that force the set S to be chosen from the first k variables. More precisely, we define

- $\Psi'\colon \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{d-k} \times \{0,1\}^{d-k} \times \{0,1\}^{d-k} \to \{0,1\}$ with

$$\Psi'(\mathbf{u}, \mathbf{v}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) = \Psi(\mathbf{u}, \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3) \wedge \left(\bigwedge_{i=1}^{k} ((u_i \oplus \neg x_i) \vee v_i) \right),$$

  where $\mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3$ is understood component-wise,

- $\mathbf{x}' = \left(\mathbf{x}_{[k]}, \mathbf{1}_k, \mathbf{x}_{[k]^c}, \mathbf{x}_{[k]^c}, \mathbf{x}_{[k]^c} \right) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{d-k} \times \{0,1\}^{d-k} \times \{0,1\}^{d-k}$,
- $k' = k$.

This is a polynomial time construction.

Necessity: Assume that $\{\Psi, \mathbf{x}, k\}$ is a *Yes*-instance for IP1. Then, there exists $S \subseteq [k]$ such that $P_{\mathbf{y}}(\Psi(\mathbf{y}) \,|\, \mathbf{y}_S = \mathbf{x}_S) > \frac{1}{2}$. Now, choose

$$S' = S \cup \{\, i \in \{k+1, \dots, 2k\} \;:\; i - k \notin S \,\}.$$

Then, $|S'| = |S| + (k - |S|) = k = k'$ and for each $i \in [k]$ exactly one of the cases $i \in S'$ or $i + k \in S'$ occurs. The former corresponds to fixing $u_i = x'_i = x_i$ and the latter to fixing $v_i = 1$. Therefore,

$$P_{(\mathbf{u},\mathbf{v})}\left(\bigwedge_{i=1}^{k} ((u_i \oplus \neg x_i) \vee v_i) \,\middle|\, (\mathbf{u}, \mathbf{v})_{S'} = \mathbf{x}'_{S'} \right) = 1,$$

which means, conditioned on $(\mathbf{u}, \mathbf{v})_{S'} = \mathbf{x}'_{S'}$, the probability of satisfying Ψ' only depends on $\Psi(\mathbf{u}, \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3)$. Now, since the random vector $\mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3$ is independent of this condition, it has the exact same distribution as the random vector $\mathbf{r}$, and we obtain

$$\begin{aligned} P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) &= P(\Psi'(\mathbf{u}, \mathbf{v}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \,|\, (\mathbf{u}, \mathbf{v})_{S'} = \mathbf{x}'_{S'}) \\ &= P\Big(\Psi'(\mathbf{u}, \mathbf{v}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \,\Big|\, \mathbf{u}_S = \mathbf{x}_S, \mathbf{v}_{[k]\setminus S} = \mathbf{1}_{k-|S|}\Big) \\ &= P(\Psi(\mathbf{u}, \mathbf{r}) \,|\, \mathbf{u}_S = \mathbf{x}_S) \\ &= P_{\mathbf{y}}(\Psi(\mathbf{y}) \,|\, \mathbf{y}_S = \mathbf{x}_S) > \frac{1}{2}. \end{aligned} \tag{4.2}$$

Hence, $\{\Psi', \mathbf{x}', k'\}$ is a *Yes*-instance for IP2.

Sufficiency: Now, conversely, assume that $\{\Psi', \mathbf{x}', k'\}$ is a *Yes*-instance for IP2. Then, there exists a set $S' \subseteq [2k + 3(d-k)]$ with $|S'| \leq k' = k$ and

$$P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) > \frac{1}{2}.$$

First, we show that S' can contain at most two indices that are not in $[2k]$. For any $i \in [k]$, consider the term $(u_i \oplus \neg x_i) \vee v_i$, which is true if $u_i = x_i$ or $v_i = 1$. This could be assured by $i \in S'$ or $i + k \in S'$ respectively. Otherwise, $P_{u_i,v_i}((u_i \oplus \neg x_i) \vee v_i) = \frac{3}{4}$. Let $N = |\{i \in [k] \,|\, i \notin S' \wedge i + k \notin S'\}|$, then

$$P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) \leq P\left(\bigwedge_{i=1}^{k} ((u_i \oplus \neg x_i) \vee v_i) \,\middle|\, (\mathbf{u}, \mathbf{v})_{S'} = \mathbf{x}'_{S'}\right) = \left(\frac{3}{4}\right)^N,$$

and since $\left(\frac{3}{4}\right)^3 < \frac{1}{2}$ but $P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) > \frac{1}{2}$, we know that $N \leq 2$.

Therefore, at most two variables out of $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$ can be fixed and thus $\mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3$ conditioned on $\mathbf{y}'_{S'} = \mathbf{x}'_{S'}$ has the same distribution as $\mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3$ without the condition. So, without loss of generality, we can even assume $S' \cap [2k]^c = \emptyset$.

Similarly, if $i \in S'$, we have $P((u_i \oplus \neg x_i) \vee v_i \,|\, (\mathbf{u}, \mathbf{v})_{S'} = \mathbf{x}'_{S'}) = 1$ and additionally having $i + k \in S'$ could not increase the probability of satisfying Ψ'. Hence, we can assume $i + k \notin S'$ in this case. Contrary, if $i \notin S'$, we have

$$P((u_i \oplus \neg x_i) \vee v_i \,|\, (\mathbf{u}, \mathbf{v})_{S'} = \mathbf{x}'_{S'}) = \frac{1}{2} + \frac{1}{2} P(v_i \,|\, (\mathbf{u}, \mathbf{v})_{S'} = \mathbf{x}'_{S'}),$$

which is one if $i + k \in S'$ and $\frac{3}{4}$ otherwise. So including $i + k$ in S' does not decrease the probability.

Altogether, without loss of generality, we can assume $S' \subseteq [2k]$, $|S'| = k$ and for each $i \in [k]$ exactly one of the cases $i \in S'$ or $i + k \in S'$ occurs. We now choose $S = S' \cap [k]$. Then, the rest of the proof proceeds exactly as in (4.2), and we conclude

$$P_{\mathbf{y}}(\Psi(\mathbf{y}) \,|\, \mathbf{y}_S = \mathbf{x}_S) = P_{\mathbf{y}'}(\Psi'(\mathbf{y}'_{S'}) \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) > \frac{1}{2},$$

implying that $\{\Psi, \mathbf{x}, k\}$ is a *Yes*-instance for IP1. □

4.3.1.3 Changing the Probability Threshold

Now, we want to change the probability threshold from $\frac{1}{2}$ to an arbitrary number $\delta \in (0,1)$ and show that the hardness does not depend on δ. Our reduction will depend on whether $\delta > \frac{1}{2}$ or $\delta < \frac{1}{2}$ since we either have to raise or lower the probability threshold. To raise the probability threshold, we make use of the following lemma.

Lemma 4.20 (Raising the probability, $>$ to $\geq$) Given $0 \leq \delta_1 < \delta_2 < 1$, for any $d \in \mathbb{N}$ there exists a monotone function $\Pi\colon \{0,1\}^n \to \{0,1\}$ such that for all $\Psi\colon \{0,1\}^d \to \{0,1\}$ we have

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) > \delta_1 \quad \Longleftrightarrow \quad P_{(\mathbf{y},\mathbf{r})}(\Psi(\mathbf{y}) \vee \Pi(\mathbf{r})) \geq \delta_2$$

with $n \in \mathcal{O}(d^2)$. The function Π can be constructed in $\mathcal{O}(n)$ time.

The constructive proof of lemma 4.20 can be found in appendix B.1. An analogous lemma is used to lower the probability threshold.

Lemma 4.21 (Lowering the probability, $>$ to $\geq$) Given $0 < \delta_1 \leq \delta_2 \leq 1$, for any $d \in \mathbb{N}$ there exists a monotone function $\Pi\colon \{0,1\}^n \to \{0,1\}$ such that for all $\Psi\colon \{0,1\}^d \to \{0,1\}$ we have

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) > \delta_2 \quad \Longleftrightarrow \quad P_{(\mathbf{y},\mathbf{r})}(\Psi(\mathbf{y}) \wedge \Pi(\mathbf{r})) \geq \delta_1$$

with $n \in \mathcal{O}(d^2)$. The function Π can be constructed in $\mathcal{O}(n)$ time.

Lastly, we introduce an auxiliary operation that allows us to make $\Psi(\mathbf{x})$ true for the initial assignment $\mathbf{x}$, while not changing the overall probability for random assignments.

Lemma 4.22 (Neutral Operation) Given $0 < \delta < 1$, for any $d \in \mathbb{N}$ there exists a monotone function $\Gamma_{d,\delta} : \{0,1\}^r \to \{0,1\}$ and positive integer $n_{d,\delta}$ with $r + n_{d,\delta} \in \mathcal{O}(d^2)$ such that for all $\Psi\colon \{0,1\}^d \to \{0,1\}$ we have

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) \geq \delta \quad \Longleftrightarrow \quad P_{(\mathbf{y},\mathbf{r},\mathbf{t})}\left((\Psi(\mathbf{y}) \wedge \Gamma_{d,\delta}(\mathbf{r})) \vee \left(\bigwedge_{i=1}^{n} t_i\right)\right) \geq \delta$$

for all $n \geq n_{d,\delta}$. The function $\Gamma_{d,\delta}$ can be consturcted in $\mathcal{O}(r)$ time.

The constructive proof of lemma 4.22 can be found in appendix B.4. Now, we are able to prove the following lemma.

Lemma 4.23 For $\delta \in (0,1)$ we have IP2 $\preceq_p$ δ-Relevant-Input. In particular, the δ-Relevant-Input problem is $\mathsf{NP}^{\mathsf{PP}}$-hard.

Proof. We start with the reduction for the case of $\delta \in (\frac{1}{2}, 1)$. Let $\{\Psi, \mathbf{x}, k\}$ be an IP2 instance. We will construct $\{\Psi', \mathbf{x}', k'\}$ that is a *Yes*-instance for δ-Relevant-Input if and only if $\{\Psi, \mathbf{x}, k\}$ is a *Yes*-instance for IP2. Let $\Pi\colon \{0,1\}^\ell \to \{0,1\}$ be as in lemma 4.20 for $\delta_1 = \frac{1}{2}$ and $\delta_2 = \delta$. Let $\Gamma = \Gamma_{d+\ell,\delta}$ and $n_{d+\ell,\delta}$ be defined according to lemma 4.22 and set $n = n_{d+\ell,\delta} + k$. We define

- $\Psi' \colon \{0,1\}^d \times \{0,1\}^\ell \times \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$,

  $$(\mathbf{y},\mathbf{r},\mathbf{s},\mathbf{t}) \mapsto ((\Psi(\mathbf{y}) \vee \Pi(\mathbf{r})) \wedge \Gamma(\mathbf{s})) \vee (\textstyle\bigwedge_{i=1}^n t_i)$$

- $\mathbf{x}' = (\mathbf{x}, \mathbf{0}_\ell, \mathbf{0}_m, \mathbf{1}_n) \in \{0,1\}^d \times \{0,1\}^\ell \times \{0,1\}^m \times \{0,1\}^n$,
- $k' = k$.

This is a polynomial time construction. By the choice of Ψ' and $\mathbf{x}'$, we guarantee $\Psi'(\mathbf{x}') = 1$ regardless of the value of $\Psi(\mathbf{x})$ since $\bigwedge_{i=0}^n 1 = 1$.

Necessity: Assume that $\{\Psi, \mathbf{x}, k\}$ is a *Yes*-instance for IP2 with satisfying set S. Then set $S' = S$ and from the definition of Π and n we get

$$\begin{aligned}
& & P_{\mathbf{y}}(\Psi(\mathbf{y}) \,|\, \mathbf{y}_S = \mathbf{x}_S) > \frac{1}{2} \\
&\iff & P_{(\mathbf{u},\mathbf{r})}(\Psi(\mathbf{u}) \vee \Pi(\mathbf{r}) \,|\, \mathbf{u}_S = \mathbf{x}_S) \geq \delta \\
&\iff & P_{(\mathbf{u},\mathbf{r},\mathbf{s},\mathbf{t})}\left((\Psi(\mathbf{u}) \vee \Pi(\mathbf{r})) \wedge \Gamma(\mathbf{s}) \vee \left(\bigwedge_{i=1}^n t_i \right) \,\middle|\, \mathbf{u}_S = \mathbf{x}_S \right) \geq \delta \\
&\iff & P_{\mathbf{y}'}(\Psi'(\mathbf{y}') = \Psi'(\mathbf{x}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) \geq \delta.
\end{aligned}$$

Hence, $\{\Psi', \mathbf{x}', k'\}$ is a *Yes*-instance for δ-Relevant-Input.

Sufficiency: Now assume that $\{\Psi', \mathbf{x}', k'\}$ is a *Yes*-instance for δ-Relevant-Input. Then there exists a subset S' with $|S'| \leq k' = k$ and $P_{\mathbf{y}'}(\Psi'(\mathbf{y}') = \Psi'(\mathbf{x}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) \geq \delta$. Since Π and Γ are monotone and their initial input assignments are $\mathbf{0}_\ell$ and $\mathbf{0}_m$, including any of their variables in S' does not increase the probability that Ψ' evaluates to $\Psi'(\mathbf{x}') = 1$. Thus, without loss of generality, we can assume that S' does no include variables from Γ. At most $k' = k$ of the n variables in the conjunction from lemma 4.22 can be included in S', which by the choice of n does not affect whether the overall probability threshold of δ is reached or not. Thus,

$$P_{\mathbf{y}'}(\Psi'(\mathbf{y}') = \Psi'(\mathbf{x}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) \geq \delta \quad \Longrightarrow \quad P_{\mathbf{y}'}\left(\Psi'(\mathbf{y}') = \Psi'(\mathbf{x}') \,\middle|\, \mathbf{y}'_{S' \cap [d]} = \mathbf{x}'_{S' \cap [d]}\right) \geq \delta.$$

We set $S = S' \cap [d]$. Clearly $|S| \leq |S'| = k' = k$. Then analogous to before,

$$P_{\mathbf{y}'}(\Psi'(\mathbf{y}') = \Psi'(\mathbf{x}') \,|\, \mathbf{y}'_S = \mathbf{x}'_S) \geq \delta \quad \iff \quad P_{\mathbf{y}}(\Psi(\mathbf{y}) \,|\, \mathbf{y}_S = \mathbf{x}_S) > \frac{1}{2},$$

implying that $\{\Psi, \mathbf{x}, k\}$ is a *Yes*-instance for IP2.

The reduction for $\delta \in (0, \frac{1}{2}]$ can be done analogously by using lemma 4.21 instead of lemma 4.20 and we omit the details for brevity. $\square$

4.3.2 δ-Relevant-Input is contained in $\mathsf{NP}^{\mathsf{PP}}$

We now come to the second part of the proof of theorem 4.10. We will show that δ-Relevant-Input is indeed contained in $\mathsf{NP}^{\mathsf{PP}}$, meaning that it can be solved in polynomial time by a non-deterministic Turing machine with access to a PP-oracle. The following lemmas, very similar to lemmas 4.20 and 4.21, will be useful.

Lemma 4.24 (Raising the probability, $\geq$ to $>$) Given $0 \leq \delta_1 \leq \delta_2 < 1$, for any $d \in \mathbb{N}$ there exists a monotone function $\Pi\colon \{0,1\}^n \to \{0,1\}$ such that for all $\Psi\colon \{0,1\}^d \to \{0,1\}$ we have

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) \geq \delta_1 \quad \Longleftrightarrow \quad P_{(\mathbf{y},\mathbf{r})}(\Psi(\mathbf{y}) \vee \Pi(\mathbf{r})) > \delta_2$$

with $n \in \mathcal{O}(d^2)$. The function Π can be constructed in $\mathcal{O}(n)$ time.

The constructive proof of lemma 4.24 can be found in appendix B.1.

Lemma 4.25 (Lowering the probability, $\geq$ to $>$) Given $0 < \delta_1 < \delta_2 \leq 1$, for any $d \in \mathbb{N}$ there exists a monotone function $\Pi\colon \{0,1\}^n \to \{0,1\}$ such that for all $\Psi\colon \{0,1\}^d \to \{0,1\}$ we have

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) \geq \delta_2 \quad \Longleftrightarrow \quad P_{(\mathbf{y},\mathbf{r})}(\Psi(\mathbf{y}) \wedge \Pi(\mathbf{r})) > \delta_1$$

with $n \in \mathcal{O}(d^2)$. The function Π can be constructed in $\mathcal{O}(n)$ time.

The constructive proof of lemma 4.25 can be found in appendix B.2.

Lemma 4.26 For $\delta \in (0,1)$ the δ-Relevant-Input problem is contained in $\mathsf{NP}^{\mathsf{PP}}$.

We will prove this for $\delta \in \left(\frac{1}{2}, 1\right)$ by lowering the probability threshold from δ to $\frac{1}{2}$. The case $\delta \in \left(0, \frac{1}{2}\right]$ can be treated analogously by raising the threshold.

Proof. Let $\{\Psi, \mathbf{x}, k\}$ be an instance of δ-Relevant-Input. It suffices to show that the decision problem whether a given set $S \subseteq [d]$ is δ-relevant for Ψ and $\mathbf{x}$ is in PP. Without loss of generality we can assume $\Psi(\mathbf{x}) = 1$. Otherwise, we could consider $\neg\Psi$ instead. Now, choose $\Pi\colon \{0,1\}^n \to \{0,1\}$ as in lemma 4.25 for $\delta_1 = \frac{1}{2}$ and $\delta_2 = \delta$. Then,

$$P_{\mathbf{y}}(\Psi(\mathbf{y}) \,|\, \mathbf{y}_S = \mathbf{x}_S) \geq \delta \quad \Longleftrightarrow \quad P_{(\mathbf{y},\mathbf{r})}(\Psi(\mathbf{y}) \wedge \Pi(\mathbf{r}) \,|\, \mathbf{y}_S = \mathbf{x}_S) > \frac{1}{2}.$$

A probabilistic Turing machine can now draw a random assignment $(\mathbf{y}, \mathbf{r})$ conditioned on $\mathbf{y}_S = \mathbf{x}_S$ and evaluate $\Psi(\mathbf{y}) \wedge \Pi(\mathbf{r})$. Thus, the machine will answer *Yes* with probability strictly greater than $\frac{1}{2}$ if and only if S is δ-relevant. This means the subproblem of checking a set for δ-relevance is contained in PP.

A non-deterministic Turing-machine with a PP-oracle can thus guess a set $S \subseteq [d]$ with $|S| \leq k$ and, using the oracle, check whether it is δ-relevant. $\square$

4.4 Variations of the Problem Formulation

We want to consider two variations of the δ-Relevant-Input problem. The first variation relaxes the requirement to check if a candidate set S is exactly δ-relevant or not by introducing a probability gap γ. In short, we then ask if a δ-relevant set of size k exists or if all sets of size k are not even $(\delta - \gamma)$-relevant.

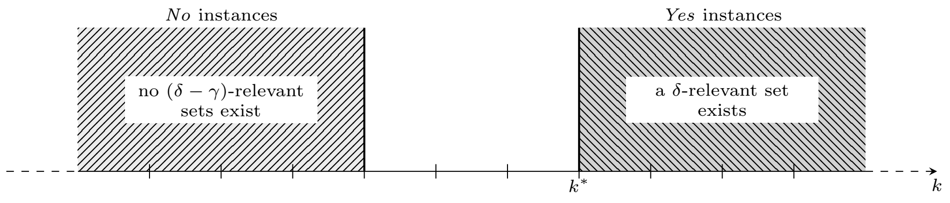

No instances
Yes instances
no $(\delta-\gamma)$-relevant sets exist
a δ-relevant set exists
k^*
k

Figure 4.3: Visualization of the γ-Gapped-δ-Relevant-Input problem for some fixed Ψ and $\mathbf{x}$ and for various k. In the unmarked region in the centre no δ-relevant set exists but $\widetilde{\delta}$-relevant sets could exist for any $\widetilde{\delta} < \delta$, in particular also for $\widetilde{\delta} = \delta - \gamma$. In this region we do not expect an answer for the gapped problem. The solution k^* of the ungapped optimisation problem Min-δ-Relevant-Input is the left boundary of the *Yes*-instance region.

The second variation concerns the optimisation version of the problem. Here, we introduce a set size gap and relax the requirement to find the smallest δ-relevant set. Instead, for $k < m$ we ask if a δ-relevant set of size k exists or if all relevant sets must be of size at least m.

We show that these problems remain hard to solve (even in combination, that is with both a gap in probability and set size). This can be used to show that no polynomial time approximation algorithm for Min-δ-Relevant-Input with approximation factor better than the trivial factor d can exists unless $\mathsf{P} = \mathsf{NP}$. Due to the connection between Boolean circuits and neural networks, as described in section 4.5.2, this inapproximability result shows theoretical limitations of interpretation methods for neural network decision.

4.4.1 The Probability Gap

As explained in the problem formulation, see section 4.2, probabilities and expectation values may be hard to calculate in theory, yet are often easy to approximate in practice via sampling. The edge cases where the true probability can be arbitrarily close to the threshold δ cause the hardness of problems in PP.

It seems impractical to defend the hardness of the δ-Relevant-Input problem with the exact evaluation of probabilities. Therefore, we introduce a variant of the problem including a probability gap. This can be seen as a promise problem with the promise that all sets S are either δ-relevant or not even $(\delta-\gamma)$-relevant. Alternatively, this can be seen as the δ-Relevant-Input problem where we want to answer *Yes* if a δ-relevant set of size k exists but only want to answer *No* if all sets of size k are not even $(\delta-\gamma)$-relevant. For cases in between we do not expect an answer at all or do not care about the exact answer. This is illustrated in fig. 4.3.

Definition 4.27 For $\delta \in (0,1]$ and $\gamma \in [0,\delta)$ we define the γ-Gapped-δ-Relevant-Input problem as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^d \to \{0,1\}$, $\mathbf{x} \in \{0,1\}^d$, and $k \in \mathbb{N}$, $1 \le k \le d$.

Decide:

Yes: There exists $S \subseteq [d]$ with $|S| \le k$ and S is δ-relevant for Ψ and $\mathbf{x}$.

No: All $S \subseteq [d]$ with $|S| \le k$ are not $(\delta-\gamma)$-relevant for Ψ and $\mathbf{x}$.

For $\gamma = 0$ we exactly retrieve the original δ-RELEVANT-INPUT problem, but for $\gamma > 0$ this is an easier question.

Lemma 4.28 For $\delta \in (0,1)$ and $\gamma \in (0,\delta)$ the γ-GAPPED-δ-RELEVANT-INPUT problem is contained in $\mathsf{NP}^{\mathsf{BPP}}$.

Proof. Let $\{\Psi, \mathbf{x}, k\}$ be an instance of γ-GAPPED-δ-RELEVANT-INPUT. It suffices to show that the decision problem whether a given set $S \subseteq [d]$ is either δ-relevant (*Yes*) or not $(\delta-\gamma)$-relevant (*No*) for Ψ and $\mathbf{x}$ is in BPP. To see this, we describe an explicit algorithm with bounded error probability:

Draw $n = \left\lceil \frac{2\ln(3)}{\gamma^2} \right\rceil$ independent random binary vectors $\mathbf{b}^{(i)} \in \{0,1\}^{d-|S|}$ for $i \in [n]$ from the uniform distribution on $\{0,1\}^{d-|S|}$ and define $\mathbf{y}^{(i)} \in \{0,1\}^d$ as $\mathbf{y}_S^{(i)} = \mathbf{x}_S$ and $\mathbf{y}_{S^c}^{(i)} = \mathbf{b}^{(i)}$. Set

$$\xi = \frac{1}{n}\sum_{i=1}^{n} \xi_i, \quad \text{where} \quad \xi_i = \begin{cases} 1, & \text{if } \Psi(\mathbf{x}) = \Psi\left(\mathbf{y}^{(i)}\right) \\ 0, & \text{if } \Psi(\mathbf{x}) \neq \Psi\left(\mathbf{y}^{(i)}\right) \end{cases} \quad \text{for} \quad i = 1, \ldots, n.$$

Then, answer *No* if $\xi < \delta - \frac{\gamma}{2}$ and *Yes* if $\xi \geq \delta - \frac{\gamma}{2}$.

The random variables ξ_i are independently and identically Bernoulli distributed variables with

$$p = \mathbb{E}[\xi_i] = \mathbb{E}[\xi] = P_{\mathbf{y}}(\Psi(\mathbf{y}_S) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S).$$

Therefore, S is δ-relevant if $p \geq \delta$ and not $(\delta-\gamma)$-relevant if $p < \delta - \gamma$. We use Hoeffding's inequality [Hoe94] to bound the error probability of the algorithm. Firstly, assume $p \geq \delta$. Then, we make an error if $\xi < \delta - \frac{\gamma}{2}$, which implies $p - \xi > \frac{\gamma}{2}$. The probability for this event can be bounded by

$$P\left(p - \xi > \frac{\gamma}{2}\right) \leq e^{-\frac{n\gamma^2}{2}} \leq \frac{1}{3}.$$

Secondly, assume $p < \delta - \gamma$. Then, we can bound the probability that $\xi \geq \delta - \frac{\gamma}{2}$, and thus $\xi - p > \frac{\gamma}{2}$, by

$$P\left(\xi - p > \frac{\gamma}{2}\right) \leq e^{-\frac{n\gamma^2}{2}} \leq \frac{1}{3}.$$

Altogether the algorithm answers correctly with probability $\frac{2}{3}$, showing that the problem lies in BPP.

A non-deterministic Turing machine with BPP-oracle can thus guess a set $S \subseteq [d]$ with $|S| \leq k$ and, using the oracle, check if it is δ-relevant or not $(\delta-\gamma)$-relevant . □

Similar to the original problem formulation, we can also state an optimisation version of the gapped problem. In this case, we relax the optimality condition on the set size k by allowing also sizes in the region between *Yes*- and *No*-instances of γ-GAPPED-δ-RELEVANT-INPUT (cf. fig. 4.3). In other words, we want to find any k that is large enough so that it is not a *No*-instance for the gapped problem but not larger than the optimal solution of the ungapped minimization problem. Strictly speaking, this results in a search problem and not an optimisation problem. However, problems of this type can be referred to as weak optimisation problems [GLS88].

Definition 4.29 For $\delta \in (0,1]$ and $\gamma \in [0,\delta)$ we define the Min-γ-Gapped-δ-Relevant-Input problem as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^d \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^d$.

Find: $k \in \mathbb{N}$, $1 \leq k \leq d$ such that

(i) There exists $S \subseteq [d]$ with $|S| = k$ and S is $(\delta - \gamma)$-relevant for Ψ and $\mathbf{x}$.

(ii) All $S \subseteq [d]$ with $|S| < k$ are not δ-relevant for Ψ and $\mathbf{x}$.

Note that both for γ-Gapped-δ-Relevant-Input and Min-γ-Gapped-δ-Relevant-Input a solution for γ_1 will always also be a solution for $\gamma_2 > \gamma_1$. Specifically, being able to solve the ungapped problems introduced in section 4.2 provides a solution to the gapped problems for any $\gamma > 0$.

4.4.2 The Set Size Gap (Approximability)

Even the gapped version of the minimisation problem is hard to approximate. We prove this by introducing another intermediate problem which we show to be NP-hard but which would be in P if there exists a “good” polynomial time approximation algorithm for Min-γ-Gapped-δ-Relevant-Input. As mentioned above, strictly speaking Min-γ-Gapped-δ-Relevant-Input is not an optimisation but a search problem. In order to give a meaning to the concept of approximation factors we use the following convention.

Definition 4.30 An algorithm for Min-γ-Gapped-δ-Relevant-Input has an approximation factor $c \geq 1$ if, for any instance $\{\Psi, \mathbf{x}\}$, it produces an approximate solution k such that there exists a true solution $\widetilde{k}$ (satisfying both conditions in definition 4.29) with $\widetilde{k} \leq k \leq c\widetilde{k}$.

An algorithm that always produces the trivial approximate solution $k = d$ achieves an approximation factor d. We will show that it is generally hard to obtain better factors. More precisely, for any $\alpha > 0$ an algorithm achieving an approximation factor $d^{1-\alpha}$ can not be in polynomial time unless P = NP.

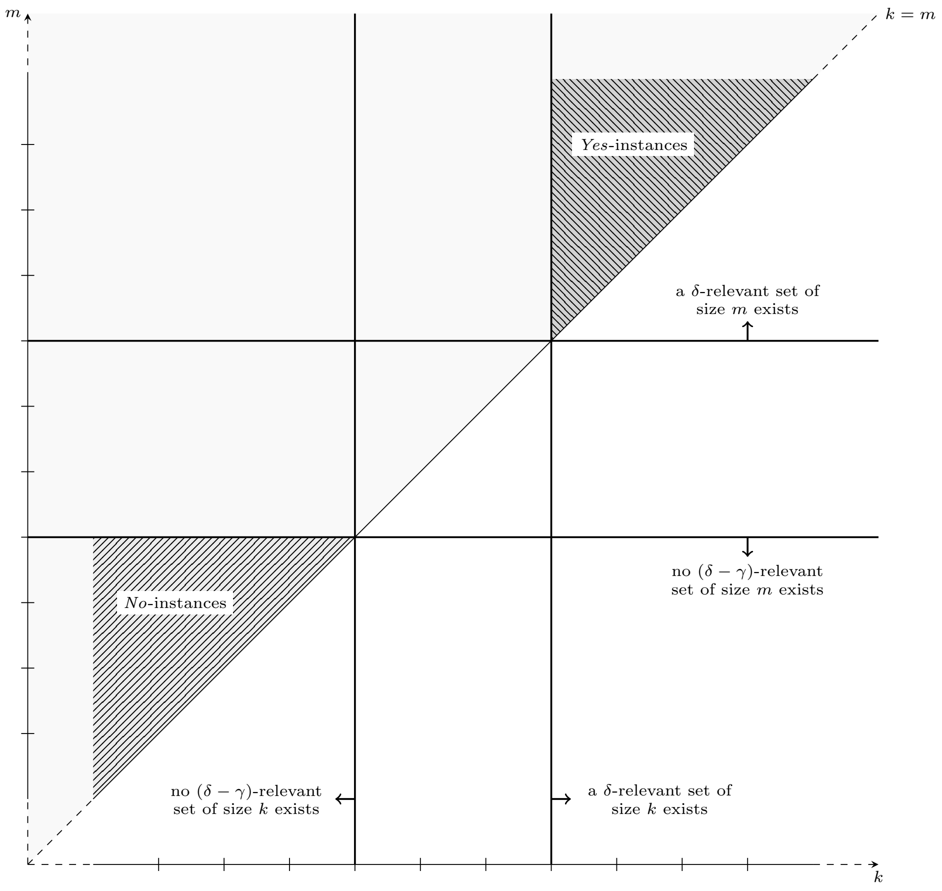
Definition 4.31 For $\delta \in (0,1]$ and $\gamma \in [0,\delta)$ we define the Intermediate Problem 3 (IP3) as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^d \to \{0,1\}$, $\mathbf{x} \in \{0,1\}^d$, and $k, m \in \mathbb{N}$, $1 \leq k \leq m \leq d$.

Decide:

Yes: There exists $S \subseteq [d]$ with $|S| \leq k$ and S is δ-relevant for Ψ and $\mathbf{x}$.

No: All $S \subseteq [d]$ with $|S| \leq m$ are not $(\delta - \gamma)$-relevant for Ψ and $\mathbf{x}$.



m
$k = m$
Yes-instances
a δ-relevant set of size m exists
No-instances
no $(\delta - \gamma)$-relevant set of size m exists
no $(\delta - \gamma)$-relevant set of size k exists
a δ-relevant set of size k exists
k

Figure 4.4: Visualization of the Intermediate Problem 3 for some fixed Ψ and $\mathbf{x}$ and for various k and m. As before we do not expect an answer for this problem in the unmarked regions. The restriction to the diagonal $k = m$ corresponds to the γ-Gapped-δ-Relevant-Input problem (cf. fig. 4.3).

The restriction to the case $k = m$ is exactly the γ-Gapped-δ-Relevant-Input problem. However, here we also allow the case $k < m$ with a gap in the set sizes. This is illustrated in fig. 4.4.

Lemma 4.32 For $\delta \in (0, 1)$ and $\gamma \in [0, \delta)$ we have SAT $\preceq_p$ IP3, in particular, in this case IP3 is NP-hard.

The idea for this proof is rather simple. Given a SAT-formula Ψ with d variables, we replace each variable by a conjunction of sufficiently many variables, i.e.

$$u_i = \bigwedge_{j=1}^{q} u_i^{(j)},$$

initially set to one. Fixing all $u_i^{(j)}$ effectively sets u_i to one. Randomising all $u_i^{(j)}$ effectively sets u_i to zero with high probability. If we now disjoin the resulting formula with a polynomially large conjunction of independent variables, i.e.

$$\Psi\left(\bigwedge_{j=1}^{q} u_1^{(j)}, \dots, \bigwedge_{j=1}^{q} u_d^{(j)}\right) \vee \left(\bigwedge_{i=1}^{M} v_i\right),$$

initially also set to one, then any satisfying assignment for Ψ yields a δ-relevant set of size at most dq by effectively setting $\mathbf{u}$ to the satisfying assignment. On the other hand, if Ψ is not satisfiable a $(\delta-\gamma)$-relevant set has to include almost all of the additional M variables. Choosing M sufficiently larger than dq results in the desired set size gap. We now make this argument formal.

Proof. Given a SAT instance in conjunctive normal form (CNF), let $\Psi\colon \{0,1\}^d \to \{0,1\}$ be the Boolean circuit representation corresponding to the CNF formula. From now on we will not distinguish between Ψ and the CNF formula that it represents. We will construct $\{\Psi', \mathbf{x}', k', m'\}$ that is a *Yes*-instance for IP3 if and only if Ψ is a *Yes*-instance for SAT. Let

$$q = \left\lceil \log_2\left(\frac{d}{1-\delta}\right)\right\rceil \quad \text{and} \quad p = \left\lfloor \log_2\left(\frac{1}{\delta-\gamma}\right)\right\rfloor + 1.$$

We set

- $k' = dq$,
- $m' \geq k'$ arbitrary but at most polynomial in d,
- $\Psi'\colon \{0,1\}^{d\times q} \times \{0,1\}^{m'+p} \to \{0,1\}$ with

  $$\Psi'(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)}, \mathbf{v}) = \Psi\left(\bigwedge_{j=1}^{q} \mathbf{u}^{(j)}\right) \vee \left(\bigwedge_{i=1}^{m'+p} v_i\right),$$

  where each $\mathbf{u}^{(j)} \in \{0,1\}^d$ and the conjunction within Ψ is understood component-wise,
- $\mathbf{x}' = \mathbf{1}_{dq+m'+p}$.

This is a polynomial time construction. By the choice of Ψ' and $\mathbf{x}'$ we guarantee $\Psi'(\mathbf{x}') = 1$ regardless of the satisfiability of Ψ.

Necessity: Let Ψ be a *Yes*-instance for SAT. This means that there exists $\mathbf{x} \in \{0,1\}^d$ with $\Psi(\mathbf{x}) = 1$. Let $S = \{\, i \in [d] : x_i = 1 \,\}$ and $S' = S \times [q]$. Then, $|S'| \leq k'$. Denote

$$A(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)}) = \bigwedge_{(i,j)\in S'} u_i^{(j)}.$$

Hence, S' is δ-relevant for Ψ' and $\mathbf{x}'$ if $P\Big(\Psi'(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)}, \mathbf{v}) \,\Big|\, A(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)})\Big) \geq \delta$. We have

$$\begin{aligned} P\Big(\Psi'(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)}, \mathbf{v}) \,\Big|\, A(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)})\Big) &\geq P\left(\Psi\left(\bigwedge_{j=1}^{q} \mathbf{u}^{(j)}\right) \,\middle|\, A(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)})\right) \\ &\geq P\left(\bigwedge_{j=1}^{q} \mathbf{u}^{(j)} = \mathbf{x} \,\middle|\, A(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)})\right). \end{aligned}$$

From this, with a union bound, we obtain

$$\begin{aligned} P\left(\bigwedge_{j=1}^{q} \mathbf{u}^{(j)} = \mathbf{x} \,\middle|\, A(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)})\right) &= 1 - P\left(\neg \bigwedge_{j=1}^{q} \mathbf{u}^{(j)} = \mathbf{x} \,\middle|\, A(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)})\right) \\ &= 1 - P\left(\exists i \in S^c : \bigwedge_{j=1}^{q} u_i^{(j)}\right) \\ &\geq 1 - |S^c| 2^{-q} \\ &\geq \delta, \end{aligned}$$

which shows that $\{\Psi', \mathbf{x}', k', m'\}$ is a *Yes*-instance for IP3.

Sufficiency: Now, conversely, let Ψ be a *No*-instance for SAT. Then, for any subset $S' \subseteq [dq + m' + p]$ with $|S'| \leq m'$ we have

$$\begin{aligned} P_{\mathbf{y}'}(\Psi'(\mathbf{y}') = \Psi'(\mathbf{x}') \,|\, \mathbf{y}'_{S'} = \mathbf{x}'_{S'}) &= P_{\mathbf{y}'}(\Psi'(\mathbf{y}') \,|\, \mathbf{y}'_{S'} = \mathbf{1}) \\ &= P_{(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)}, \mathbf{v})}\left(\bigwedge_{i=1}^{m'+p} v_i \,\middle|\, (\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(q)}, \mathbf{v})_{S'} = \mathbf{1}\right) \\ &\leq 2^{-(m'+p-|S'|)} \\ &\leq 2^{-p} \\ &< \delta - \gamma. \end{aligned}$$

This shows that S' is not $(\delta - \gamma)$-relevant for Ψ' and $\mathbf{x}'$, hence $\{\Psi', \mathbf{x}', k', m'\}$ is a *No*-instance for IP3. □

Recall the second main theorem of the paper which shows the inapproximability of the MIN-γ-GAPPED-δ-RELEVANT-INPUT problem.

Theorem 4.11 *Let $\delta \in (0, 1)$ and $\gamma \in [0, \delta)$. Then, for any $\alpha \in (0, 1)$ there is no polynomial time approximation algorithm for MIN-γ-GAPPED-δ-RELEVANT-INPUT with an approximation factor of $d^{1-\alpha}$ unless $\mathsf{P} = \mathsf{NP}$.*

The proof idea is to choose the m' in the previous proof large enough such that even an approximation algorithm that promises only a rough approximation factor could still be used to solve an NP-hard problem.

Proof. We prove this by showing that the existence of such an approximation algorithm would allow us to decide IP3 in polynomial time for certain instances. These can be chosen as in the proof of lemma 4.32, which in turn implies that we could decide SAT in polynomial time. This is only possible if $\mathsf{P} = \mathsf{NP}$.

Given a SAT instance as a CNF formula, let $\Psi \colon \{0, 1\}^d \to \{0, 1\}$ be a Boolean circuit representation of the CNF formula and $\{\Psi', \mathbf{x}', k', m'\}$ an equivalent IP3 instance as in the proof of lemma 4.32. As before, we will not further distinguish between the SAT formula in CNF and the circuit Ψ representing it. We have seen that there is some freedom in the choice of m' as long as it satisfies $k' \leq m'$ and is at most polynomial in d. We will choose it in such a way, that any approximate solution k with approximation factor ${d'}^{1-\alpha}$ would allow us to decide $\{\Psi', \mathbf{x}', k', m'\}$ by checking whether $k < m'$ or $k > m'$. For this we set $m' = \left\lceil \max(2k'({k'}^{1-\alpha} + p^{1-\alpha}), (2k')^{\frac{1}{\alpha}} + 1) \right\rceil$ with $p = \left\lfloor \log_2\left(\frac{1}{\delta-\gamma}\right) \right\rfloor + 1$ as before. Recall that $k' = dq$ with $q = \left\lceil \log_2\left(\frac{d}{1-\delta}\right) \right\rceil$, so clearly m' is polynomial in d and $k' \leq m'$. Further, we have $m' > (2k')^{\frac{1}{\alpha}}$ so $1 - k'{m'}^{-\alpha} > \frac{1}{2}$ and therefore

$$m'(1 - k'{m'}^{-\alpha}) > \frac{m'}{2} \geq k'({k'}^{1-\alpha} + p^{1-\alpha}).$$

Now, let $d' = k' + m' + p$ denote the number of variables of Ψ'. By the subadditivity of the map $z \mapsto z^{1-\alpha}$ we finally obtain

$$k'{d'}^{1-\alpha} = k'(k' + m' + p)^{1-\alpha} \leq k' \left({k'}^{1-\alpha} + {m'}^{1-\alpha} + p^{1-\alpha}\right) < m'.$$

It remains to show that an IP3 instance with $m' > k'{d'}^{1-\alpha}$ can be decided by an approximation algorithm for Min-γ-Gapped-δ-Relevant-Input with approximation factor ${d'}^{1-\alpha}$. Assume such an algorithm exists and let k be an approximate solution. Then, there exists a true solution $\widetilde{k}$ with $\widetilde{k} \leq k \leq {d'}^{1-\alpha}\widetilde{k}$.

Firstly, assume that $\{\Psi', \mathbf{x}', k', m'\}$ is a *Yes*-instance for IP3. Then, there is a δ-relevant set of size k'. However, no set smaller than $\widetilde{k}$ can be δ-relevant. This implies $\widetilde{k} \leq k'$ and therefore $k \leq {d'}^{1-\alpha}k' < m'$.

Secondly, assume that $\{\Psi', \mathbf{x}', k', m'\}$ is a *No*-instance for IP3. Then, all sets of size at most m' are not $(\delta - \gamma)$-relevant. However, there exists a $(\delta - \gamma)$-relevant set of size $\widetilde{k}$. This implies $k \geq \widetilde{k} > m'$.

Altogether, checking whether $k < m'$ or $k > m'$ decides $\{\Psi', \mathbf{x}', k', m'\}$. □

4.5 Discussion

We want to briefly discuss the scope of our analysis and the implications for algorithms that explain the predictions of classifiers such as neural networks. One could ask whether a solution set S to the Min-δ-Relevant-Input problem is in itself already a good abductive explanation for a classifier prediction.

We are not arguing that a solution set alone is enough to explain the decision of a classifier. The solution sets have some limitations that are discussed in the following subsection. We rather argue that any good explanation should contain a solution of the Min-δ-Relevant-Input problem. and thus answer questions *Q1* and *Q2*. The evaluation methods for explanations of [Sam+17] and [FV17] indicate that practitioners agree and design algorithms that should solve Min-δ-Relevant-Input in practice. Yet, our hardness results indicate that efficient methods cannot be proven to achieve this under all circumstances.

4.5.1 Stability and Uniqueness of δ-Relevant Sets

One should note that, like prime implicant explanations, the solution sets of the Min-δ-Relevant-Input problem are generally not unique. However, this behaviour is expected since an input can contain redundant information and each part of it alone can already be sufficient for the prediction. Even for instances with unique solution sets, these can depend sensitively on the probability threshold δ, i.e. slight changes in δ can lead to very different (and possibly not even overlapping) solution sets.

Further, the concept of δ-relevance is not monotone, in the sense that if S_1 is δ-relevant then $S_2 \supseteq S_1$ does not need to be δ-relevant as well. Again, this behaviour is expected, since there can be negative evidence in some of the variables. For example, think of a cat-vs-dog image classifier and an input containing both a cat and a dog. A set of pixels including the cat will get less relevant for the prediction "cat" if we add more pixels covering the dog to it. In fact, this non-monotonicity property was essential for the constructions used in our proof of the inapproximability theorem.

4.5.2 Choice of Function class

In our analysis, we considered Boolean circuit classifiers and binary input variables. As discussed in the introduction, the classifier is fixed in each problem instance, thus any class of classifiers that can efficiently describe Boolean circuits is also subject to our hardness results. This includes ReLU neural networks as well as Bayesian networks.[2]

We might consider restricting the allowed function classes to achieve computational feasibility of the problem. Although model counting can be done efficiently for various classes of representations of Boolean functions, e.g. BDDs [Bry86], deterministic Decomposable Negation Normal Forms (d-DNNF) [Dar00] and Sentential Decision Diagrams (SDD) [Dar11], this alone does not solve the inapproximability of our problem as we proved in section 4.4.2. Going further, we do not see a straightforward way to extend the prime implicant finding algorithm of [CM92] for BDDs to our problem setting with $\delta < 1$. The basic observation underlying the algorithm is that a set of variables not containing x_j is an implicant for $\Psi(\mathbf{x})$ exactly if it is an implicant for both $\Psi(x_1, \dots, x_j = 0, \dots, x_d)$ and $\Psi(x_1, \dots, x_j = 1, \dots, x_d)$. This is not true for δ-relevance.

The work of Izza et al. has revealed that subset-minimal δ-relevant sets can be computed with polynomially many calls to an NP-oracle by restricting the function class to decision trees [Izz+21].

4.5.3 Choice of Distribution

We restricted our analysis to the case of using the uniform distribution over the binary cube to randomise non-relevant variables. One could also consider more data-adapted or even empirical distributions. However, this may obscure insights about the classifiers reasoning. As an example, consider a faulty image classifier for boats that recognises water instead. If the data-adapted distribution only models images of boats on water, then marking the boat as relevant will always lead to random completions including water around the boat. Consequently, the prediction will remain unchanged even though the boat is not the true underlying reason for the classifier prediction. The classifier appears to work correctly, even though it will fail for images showing other objects on water. Instead, if the distribution

[2]The conditional probabilities describing the Bayesian network can represent truth tables of logical operators. This allows them to emulate Boolean circuits [Par02].

used for the random completion is oblivious to the correlation between boats and water, the function output will only remain constant when the water is fixed, revealing the true relevant region.

4.5.4 Implications

Our analysis shows that practical guarantees for the interpretation of neural networks are infeasible, as long as the networks are powerful enough to represent arbitrary logical functions. It is thus necessary to further restrict the problem setting. However, the hardness instances we construct can already be represented by neural networks with a fixed number of layers and bounded weights. Excluding these instances by further restricting these coarse hyperparameters would go against the idea of neural networks. This only leaves the option of more subtle restrictions on the neural networks and the inputs that depend on the actual data structures on which the networks have been trained. These, however, are not yet well enough understood.

In the next chapter, we will extend this analysis to continuous functions such as neural networks. Most notably, we extend the analysis to the uniform distribution on the continuous hypercube, which is a more natural choice for functions on a continuous domain. We will see that our inapproximability results can be carried over to this setting, strengthening the points we raise in this chapter.

5 Rate-Distortion Explanations

Since most neural network classifiers operate on a continuous domain, we are going to extend our notion of probabilistic prime implicants from the Boolean setting. This results in a rate-distortion formalism (see for example [Ber03] for an overview) where the rate corresponds to the number of input components that are considered relevant and the distortion corresponds to the expected change in the classification if only the relevant part of the input was known. We start by giving the formal description of the rate-distortion framework and its relation to prime implicants.

5.1 A Rate-Distortion Trade-off

The concept of probabilistic-prime implicants was defined for Boolean functions, but neural networks are ultimately continuous. Our definition of δ-relevance introduces a natural trade-off between the probability threshold δ and the minimal set size $|S|$ necessary to achieve it. This can be interpreted as a *rate-distortion trade-off*. We will borrow the language of rate-distortion theory in order to describe the concepts needed for our continuous extension.

Given $\mathbf{x} \in \{0,1\}^d$ and $S \subset [d]$ we write the shorthand $\mathbf{y} \sim \mathcal{U}_S$ when

$$\begin{aligned} \mathbf{y}_S &= \mathbf{x}_S, \\ \mathbf{y}_{S^c} &\sim \mathcal{U}\Big(\{0,1\}^{d-|S|}\Big), \end{aligned}$$

keeping the dependence of the random Variable on $\mathbf{x}$ implicit. Then, we can rewrite the δ-relevance condition definition 4.5 as

$$\begin{aligned} & \mathbb{P}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[\Psi(\mathbf{y}) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S] \geq \delta \\ \iff \quad & \mathbb{E}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[|\Psi(\mathbf{y}) - \Psi(\mathbf{x})| \,|\, \mathbf{y}_S = \mathbf{x}_S] \leq 1-\delta \\ \iff \quad & \mathbb{E}_{\mathbf{y}\sim\mathcal{U}_S}[|\Psi(\mathbf{y}) - \Psi(\mathbf{x})|] \leq 1-\delta. \end{aligned} \tag{5.1}$$

The right hand side $1-\delta$ can be seen as a distortion measure bounding the expected change in the classifier prediction. This formulation through an expectation is well suited to be generalised to a continuous setting. Also, other probability distributions as well as other distance measures than the absolute difference might be of interest.

Let now $\Phi\colon [0,1]^d \to [0,1]$ be a general classifier function, $\mathcal{V}$ be a probability distribution on $[0,1]^d$, and $\mathbf{n} \sim \mathcal{V}$ a random vector. We define the *obfuscation* of a signal $\mathbf{x} \in [0,1]^d$ with respect to $S \subseteq [d]$ and $\mathcal{V}$ as a random vector $\mathbf{y}$ that is deterministically defined on S as $\mathbf{y}_S = \mathbf{x}_S$ and distributed on the complement according to $\mathbf{y}_{S^c} = \mathbf{n}_{S^c}$. Explicitely

$$\mathbf{y} \sim \mathcal{V}_S \overset{\text{def}}{\iff} \begin{cases} \mathbf{y}_S &= \mathbf{x}_S, \\ \mathbf{y}_{S^c} &= \mathbf{n}_{S^c} \quad \text{where } \mathbf{n} \sim \mathcal{V}. \end{cases}$$

As above, we write $\mathcal{V}_S$ for the resulting distribution of $\mathbf{y}$. This enables us to define the distortion of S with respect to $\Phi, \mathbf{x}$ and $\mathcal{V}$ as

$$D(S, \Phi, \mathbf{x}, \mathcal{V}) = \mathbb{E}_{\mathbf{y}\sim\mathcal{V}_S}\left[\operatorname{dist}(\Phi(\mathbf{x}), \Phi(\mathbf{y}))\right],$$

where $\operatorname{dist}\colon [0,1]^2 \to [0,\infty)$ is some distance measure, e.g. absolute difference or squared difference. We will use the abbreviated notation $D(S)$, whenever Φ, $\mathbf{x}$, and $\mathcal{V}$ are clear from the context.

The relationship between set size and distortion is described by the rate-distortion function, defined as

$$R(\epsilon, \Phi, \mathbf{x}, \mathcal{V}) = \min\{\, |S| \,:\, S \subseteq [d],\, D(S, \Phi, \mathbf{x}, \mathcal{V}) \leq \epsilon \,\}. \tag{5.2}$$

The distortion limit ϵ takes the role of $1-\delta$ from before. Again, we use the abbreviation $R(\epsilon)$ if the context is clear.

The idea of formulating relevance in a rate-distortion viewpoint can also be motivated with the hypothetical setup illustrated in Figure 5.1. The terminology is borrowed from information theory where rate-distortion is used to analyse lossy data compression. In that sense, the set of relevant components can be thought of as a compressed description of the signal with the expected deviation from the classification score being a measure for the reconstruction error.

This framework is used to state a clearly defined objective that relevance maps should fulfil: Given a distortion limit ϵ, the goal is to find a set S achieving the minimum in (5.2), i.e., evaluating the rate-distortion function. This generalises the problem of finding small δ-relevant sets to functions on continuous domains. Thus, evaluating the rate-distortion function amounts to answering the questions *Q1* and *Q2*. We will show that no efficient algorithm can always fulfil this objective. Still, it can be used to numerically evaluate the quality of relevance maps that were produced by heuristic algorithms, as discussed in Sections 5.3 and 7.2.

5.2 Computational Complexity

We have shown the inapproximability of small relevant sets for binary functions, represented as Boolean circuits, and the uniform distribution $\mathcal{U}\left(\{0,1\}^d\right)$ on $\{0,1\}^d$ in section 4.2—a result we now generalise for classifiers on a continuous domain.

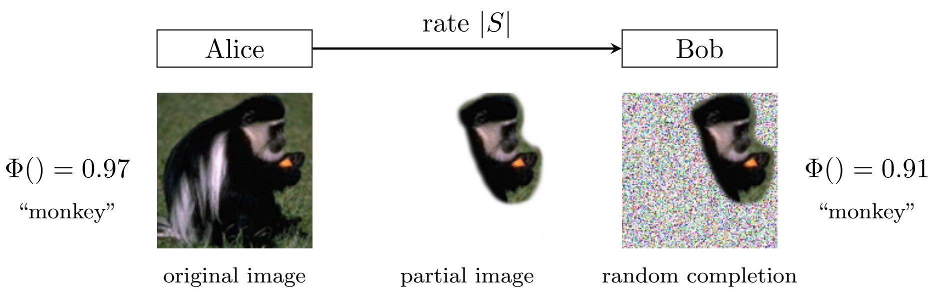


Figure 5.1: We motivate the rate-distortion viewpoint from the following hypothetical scenario: two people, Alice and Bob, have access to the same neural network classifier. Alice classified an image as a "monkey" and wants to convey this to Bob. She is only allowed to send a limited number of pixels to Bob, who will complete the image with random values. Alice's best chance of convincing Bob is to transmit those pixels that are most relevant for the class "monkey" and ensure a small difference between their classification scores in expectation.

5.2.1 Approximating the Rate-Distortion Function

For a fixed distribution $\mathcal{V}$ on $[0,1]^d$, a family of classifiers functions $\mathcal{F} \subseteq \left\{\Phi\colon [0,1]^d \to [0,1]\right\}$, and distortion limit $\epsilon \geq 0$ we say that an algorithm to evaluate the rate-distortion function achieves the approximation factor $c \geq 1$ if for any signal $\mathbf{x} \in [0,1]^d$ and classifier $\Phi \in \mathcal{F}$ it computes a set S of size

$$R(\epsilon, \Phi, \mathbf{x}, \mathcal{V}) \leq |S| \leq cR(\epsilon, \Phi, \mathbf{x}, \mathcal{V}),$$

satisfying $D(S, \Phi, \mathbf{x}, \mathcal{V}) \leq \epsilon$. In other words S can be larger than the optimal size $R(\epsilon, \Phi, \mathbf{x}, \mathcal{V})$ by at most a factor c. The approximation factor d can trivially be achieved by simply taking $S = [d]$, i.e., taking all input components as relevant. We will show for the important case of neural network classifiers that anything beyond this is computationally hard. There is no efficient algorithm that can do significantly better than the trivial factor d.

5.2.2 Neural Network Functions

Let $L \in \mathbb{N}$ denote the number of layers of a neural network, $d_1, \ldots, d_{L-1} \in \mathbb{N}$ and $d_0 = d$, $d_L = 1$. Further let $(\mathbf{W}_1, \mathbf{b}_1), \ldots, (\mathbf{W}_L, \mathbf{b}_L)$ with $\mathbf{W}_i \in \mathbb{R}^{d_i \times d_{i-1}}$, $\mathbf{b}_i \in \mathbb{R}^{d_i}$ for $i \in [L]$ be weight matrices and bias vectors. From now on we consider $\mathcal{F}$ to contain functions of the form

$$\Phi(\mathbf{x}) = \mathbf{W}_L \varrho(\ldots \varrho(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) \ldots) + \mathbf{b}_L,$$

with the rectified linear unit (ReLU) activation $\varrho(x) = \max\{0, x\}$. A neural network $\Phi\colon [0,1]^d \to [0,1]$ is said to *interpolate* a binary classifier $\Psi\colon \{0,1\}^d \to \{0,1\}$ if Φ restricted to $\{0,1\}^d$ is equal to Ψ. We will make use of the fact that ReLU neural networks can interpolate arbitrary Boolean circuits with comparable depth and width [MB17; Par96].

5.2.3 Main Result

We now come to our main result, showing the hardness of approximating the rate-distortion function for neural network classifiers. For this, we assume that the distance measure satisfies the two properties

(P1) dist is bounded on $[0,1]^2$ by a constant $M > 0$, (P2) $\operatorname{dist}(0,0) = \operatorname{dist}(1,1) = 0$ and $\operatorname{dist}(1,0) = \operatorname{dist}(0,1) = 1$,

which can be guaranteed (if necessary by appropriate scaling) for most common positive definite and symmetric distance measures, such as absolute or squared difference. For ease of presentation, we state our result only for the uniform distribution. However, we want to remark that it is also valid for more general distributions, see Section 6.5 for a discussion.

Theorem 5.1 *Let $\mathcal{V} = \mathcal{U}\left([0,1]^d\right)$ and assume $\mathsf{P} \neq \mathsf{NP}$. Then for any $\epsilon \in (0,1)$ and $\alpha \in (0,1]$ there does not exist a polynomial time approximation algorithm for $R(\epsilon, \Phi, \mathbf{x}, \mathcal{V})$ with approximation factor $d^{1-\alpha}$.*

We will prove this by reducing an NP-hard problem from the binary setting considered in [Wäl+21] to the problem of evaluating the rate-distortion function. Let us quickly recall some notions from the binary case.

Definition 5.2 (see chapter section 4.2) For $\delta \in (0,1]$ and $\gamma \in [0,\delta)$ the Min-γ-Gapped-δ-Relevant-Input problem is defined as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^d \to \{0,1\}$ and a variable assignment $\mathbf{x} \in \{0,1\}^d$.

Find: $k \in \mathbb{N}$, $1 \leq k \leq d$ such that

1. there exists a set $S \subseteq [d]$ with $|S| = k$ and S is $(\delta - \gamma)$-relevant for Ψ and $\mathbf{x}$,
2. all sets $S \subseteq [d]$ with $|S| < k$ are not δ-relevant for Ψ and $\mathbf{x}$.

An algorithm for Min-γ-Gapped-δ-Relevant-Input is said to have an approximation factor $c \geq 1$ if, for any instance $\{\Psi, \mathbf{x}\}$, it produces an approximate solution k such that there exists a true solution $\tilde{k}$ (satisfying both conditions in Definition 4.29) with $\tilde{k} \leq k \leq c\tilde{k}$. We showed that for any $\alpha > 0$ no polynomial time approximation algorithm for Min-γ-Gapped-δ-Relevant-Input with approximation factor $d^{1-\alpha}$ exists, unless $\mathsf{P} = \mathsf{NP}$ [Wäl+21].

The idea for the proof of Theorem 5.1 can be summarised in a few steps: Given a Boolean circuit Ψ we choose an interpolating ReLU network Φ_0. For any $\eta > 0$ there exists a fixed size ReLU network Φ_η that transforms the uniform distribution $\mathcal{U}\left([0,1]^d\right)$ into the binary distribution $\mathcal{U}\left(\{0,1\}^d\right)$ up to a small error depending explicitly on η, such that $\Phi = \Phi_0 \circ \Phi_\eta$ still interpolates Ψ. The difference of the distortions $D\left(S, \Phi, \mathbf{x}, \mathcal{U}\left([0,1]^d\right)\right)$ and $D\left(S, \Psi, \mathbf{x}, \mathcal{U}\left(\{0,1\}^d\right)\right)$ can be shown to depend explicitly on η as well. Moreover, the binary distortion is directly related to the probability lower bounded by δ in Definition 4.5. Thus, it can be shown that for the right choice of η any approximation algorithm for the rate-distortion function would also be an approximation algorithm for the Min-γ-Gapped-δ-Relevant-Input problem with the same approximation factor. The inapproximability result in [Wäl+21] thus carries over to the continuous setting.

For brevity, we introduce the notation

$$
\begin{aligned}
D^b_{\Phi,\mathbf{x}}(S) &= D\left(S, \Phi, \mathbf{x}, \mathcal{U}\left(\{0,1\}^d\right)\right),\\
D^c_{\Phi,\mathbf{x}}(S) &= D\left(S, \Phi, \mathbf{x}, \mathcal{U}\left([0,1]^d\right)\right),
\end{aligned}
$$

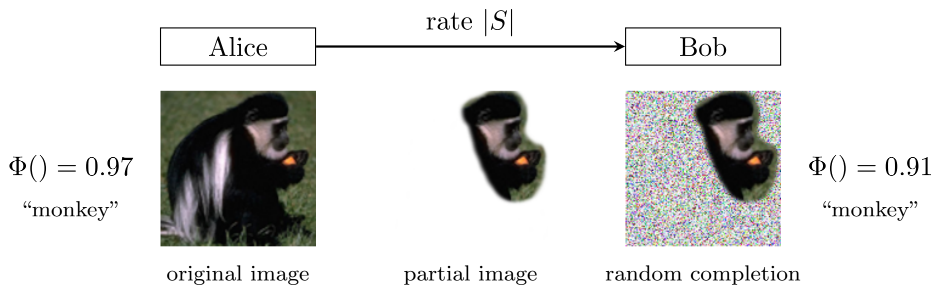


Figure 5.2: Illustration of the binarisation function Φ_η for one component. The event E_S in the proof of Lemma 5.4 describes those $\mathbf{y} \in [0,1]^d$ for which at least one component not indexed by S falls into the gray marked region of width η.

for the binary and the continuous distortion respectively. For $0 < \eta \leq 1$ we set $\Phi_\eta(\mathbf{x}) = \varphi\Big(\frac{1}{\eta}\Big(\mathbf{x} - \frac{1-\eta}{2}\mathbf{1}_d\Big)\Big)$ with

$$\varphi(x) = \begin{cases} 0, & x \leq 0, \\ x, & 0 < x \leq 1, \\ 1, & x > 1, \end{cases}$$

and observe that Φ_η interpolates the identity on $\{0,1\}^d$ and can be realised by two ReLU layers of size $\mathcal{O}(d)$, cf. Figure 5.2. Before we come to the main proof, we state two more useful results.

Lemma 5.3 Let $\Psi\colon \{0,1\}^d \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^d$. Then for any $S \subseteq [d]$ we have

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[\Psi(\mathbf{y}) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S] = 1 - D^b_{\Psi,\mathbf{x}}(S).$$

This follows from a direct calculation analogous to (5.1) using the property (P2) of the distance function.

Lemma 5.4 Let $\Psi\colon \{0,1\}^d \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^d$. Then for any $\Phi_0\colon [0,1]^d \to [0,1]$ interpolating Ψ, $S \subseteq [d]$, and $0 < \eta \leq 1$ we have for $\Phi = \Phi_0 \circ \Phi_\eta$ that

$$D^b_{\Phi,\mathbf{x}}(S) = D^b_{\Phi_0,\mathbf{x}}(S) = D^b_{\Psi,\mathbf{x}}(S)$$

as well as

$$\Big|D^c_{\Phi,\mathbf{x}}(S) - D^b_{\Psi,\mathbf{x}}(S)\Big| \leq Md\eta.$$

The proofs for both can be found in appendix C.1. We now come to the proof of the main theorem.

Proof of Theorem 5.1. Given $\epsilon \in (0,1)$ we choose $\delta \in (0,1)$ and $\gamma \in (0,\delta)$ such that $\epsilon = 1 - \delta + \frac{\gamma}{2}$. Let $\{\Psi, \mathbf{x}\}$ be an instance of Min-γ-Gapped-δ-Relevant-Input. Let $\Phi_0\colon [0,1]^d \to [0,1]$ be a neural network that interpolates Ψ, set $\eta = \frac{\gamma}{2Md}$ and $\Phi = \Phi_0 \circ \Phi_\eta$. We show that $R\Big(\epsilon,\Phi,\mathbf{x},\mathcal{U}\big([0,1]^d\big)\Big)$ is a solution for the MIN-γ-GAPPED-δ-RELEVANT-INPUT problem instance $\{\Psi,\mathbf{x}\}$, i.e., it fulfils both conditions in Definition 4.29. To see this, let

$$S^* \in \operatorname{argmin}\Big\{|S| : S \subseteq [d], D\Big(S,\Phi,\mathbf{x},\mathcal{U}\big([0,1]^d\big)\Big) \le \epsilon\Big\},$$

and hence $|S^*| = R\Big(\epsilon,\Phi,\mathbf{x},\mathcal{U}\big([0,1]^d\big)\Big)$. Lemma 5.4 yields

$$\begin{aligned} 1 - D^b_{\Psi,\mathbf{x}}(S^*) &\ge 1 - \Big(D^c_{\Phi,\mathbf{x}}(S^*) + Md\eta\Big) \\ &\ge 1 - \epsilon - \frac{\gamma}{2} = \delta - \gamma, \end{aligned}$$

and together with Lemma 5.3 we get

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[\Psi(\mathbf{y}) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_{S^*} = \mathbf{x}_{S^*}] \ge \delta - \gamma,$$

showing that the first condition in Definition 4.29 is satisfied.

Similarly, for any S with $|S| < R\Big(\epsilon,\Phi,\mathbf{x},\mathcal{U}\big([0,1]^d\big)\Big)$ we know $D\Big(S,\Phi,\mathbf{x},\mathcal{U}\big([0,1]^d\big)\Big) > \epsilon$. Thus by Lemma 5.4

$$\begin{aligned} 1 - D^b_{\Psi,\mathbf{x}}(S) &\le 1 - \Big(D^c_{\Phi,\mathbf{x}}(S) - Md\eta\Big) \\ &< 1 - \epsilon + \frac{\gamma}{2} = \delta, \end{aligned}$$

and again using Lemma 5.3 we obtain

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[\Psi(\mathbf{y}) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_{S} = \mathbf{x}_{S}] < \delta,$$

showing that the second condition in Definition 4.29 is satisfied as well.

Hence, any algorithm approximating the rate-distortion function $R\Big(\epsilon,\Phi,\mathbf{x},\mathcal{U}\big([0,1]^d\big)\Big)$ can also be used as an approximation algorithm for MIN-γ-GAPPED-δ-RELEVANT-INPUT with the same approximation factor. For the latter it is known that achieving the factor $d^{1-\alpha}$ is NP-hard for any $\alpha > 0$, which completes the proof. □

5.2.4 Non-Uniform Distributions

We use the uniform distribution $\mathcal{U}\big([0,1]^d\big)$ as a reference or baseline distribution $\mathcal{V}$ in our proof for the hardness result on approximating the rate-distortion function. This can easily be extended to more general probability measures μ on $[0,1]^d$. We can choose μ as a product of any independent one-dimensional measures μ_i for the individual input components as long as for every $i \in [d]$ and $0 < \eta \le 1$ there exist lower and upper thresholds $a_i, b_i \in [0,1]$ with $a_i < b_i$ such that

$$\mu_i([0,a_i]) = \mu_i([b_i,1]) \qquad \text{and} \qquad \mu_i([a_i,b_i]) \le \eta.$$

This is possible for all probability measures on $[0,1]$ without point masses, such as truncated Gaussian or exponential distributions, as they have continuous distribution functions. In that case we simply have to adapt the function Φ_η to $\mathbf{x} \mapsto \varphi((\mathbf{x}-\mathbf{a}) \oslash (\mathbf{b}-\mathbf{a}))$ and proceed with the remaining proof as before.

We want to stress that this is a worst-case analysis and does not imply that the task is always infeasible in practical applications. Yet, performance guarantees cannot be proven as long as the neural networks considered are powerful enough to represent arbitrary logical functions, which is the case for ReLU networks.

Hence, we have to rely on heuristic solution strategies and have to evaluate them numerically at how well they answer questions *Q1* and *Q2*. However, as noted before, most existing heuristics produce continuous relevance scores instead of a strict partition into relevant and irrelevant sets. We explain how to reconcile this difference in the following section and how to use the rate-distortion framework to evaluate any continuous map numerically.

5.2.5 Conditional versus Marginal Distributions

We want to make another remark regarding the choice of the distributions $\mathcal{V}_S$ used in our rate-distortion framework. We define the obfuscation $\mathbf{y}$ to be deterministically given by $\mathbf{y}_S = \mathbf{x}_S$ on S and distributed according to $\mathbf{y}_{S^c} = \mathbf{n}_{S^c}$ with $\mathbf{n} \sim \mathcal{V}$ on the complement S^c. This means that the resulting distribution $\mathcal{V}_S$ of $\mathbf{y}$ corresponds to $\mathcal{V}$ marginalised over all components in S. One might be tempted to condition on the given components $\mathbf{x}_S$ instead of marginalising. But this could actually be detrimental to uncover how the classifier operates. Let us illustrate this with an example. Consider a classifier that is trained to detect ships, but actually only learned to detect the water surrounding the ship, as in [Lap+16b]. The classifier can achieve high accuracy as long as the data set only contains ships on water and no other objects surrounded by water. Now assume we have a relevance map selecting a subset of pixels showing a ship as relevant. If we complete the rest of the image with random values from a conditional distribution, we will most likely see water in the completion, as most images with a ship will also have water surrounding it. The classifier would correctly classify the completed image with high probability. The potentially small subset of pixels containing the ship will thus give a small distortion and will be considered relevant. However, this result is not useful to uncover the underlying workings of the network. It does not tell us that the network does not recognise ships but only the surrounding water. Using a very data adapted and restricted conditional distribution compensates the shortcoming of the network. That is why we advocate for using a less data adapted marginal distribution. In fact, we believe that using mostly uninformed distributions like uniform or truncated Gaussian distributions is beneficial for uncovering the network's reasoning.

5.3 Relevance Scores and Orderings

It seems not immediately clear how the continuous relevance scores calculated by many established explanation methods relate to the partitions into a set of relevant and non-relevant input components that were discussed in Sections 5.1 and 5.2. However, we argue that the exact numerical values of such a continuous relevance map are generally meaningless. Instead, it is the *ordering* of the input components according to their relevance scores that is of importance. Let $\pi\colon [d] \to [d]$ be a permutation that describes a relevance ordering in the sense that $\pi(k)$ is the k-th most relevant input component and $\pi([k])$ are the k most relevant input components.

This ordering can be seen as a greedy approach to solve one of the following two questions[1] for varying ϵ.

Productive Formulation: If we want to *preserve* the class prediction $\Phi(\mathbf{x})$ up to a maximal distortion of $D(S) \leq \epsilon$, which is the smallest set S of components we should fix?

Destructive Formulation: If we want to *destroy* the class prediction $\Phi(\mathbf{x})$ with minimal distortion $D(S^c) \geq \epsilon$, which is the smallest set S of components we should obfuscate?

Both formulations might be equally valid depending on the application. Though seemingly equivalent, these questions do generally not have the same answer. Consider, for example, the case of redundancy, e.g., a picture with two monkeys that was classified as containing a monkey. In the productive scenario it can be sufficient to include just one of the monkeys in S, while in the destructive scenario one should try to obfuscate both monkeys equally.

All existing quantitative evaluation methods for relevance maps implicitly use one of these formulations: they are based on obfuscating or perturbing parts of the input components that are deemed most or least relevant and measure the change in the classification score. Zeiler and Fergus consider obfuscations by a constant baseline value [ZF14b], Samek et al. use obfuscations by random values [Sam+17], and Fong and Vedaldi use both types of obfuscations as well as perturbations by blurring [FV17].

In this work we focus entirely on the productive formulation, but we conjecture that a hardness result comparable to Theorem 5.1 also holds for the destructive case. The size of the optimal solution for the productive formulation is described by our rate-distortion function $R(\epsilon)$. A good relevance ordering is one that provides good approximations to the optimal size, when we greedily include input components in descending order of their relevance until the distortion limit is satisfied. The rate function associated to an ordering π is thus given by

$$R_\pi(\epsilon) = \min\{\, k \in [d] : \; D(\pi([k])) \leq \epsilon \,\}.$$

Clearly $R(\epsilon) \leq R_\pi(\epsilon)$ holds for any ordering π. But we can evaluate relevance maps by how well the rate function associated to the induced relevance ordering approximates the optimal rate $R(\epsilon)$. It would be desirable to obtain meaningful upper bounds on the approximation error. Unfortunately, we have seen that no non-trivial approximation bound can be given for any efficient method of calculating relevance maps. They cannot be proven to perform systematically better than a random ordering and do not provably find small relevant sets, even when they exist. Nevertheless, the ordering based rate functions R_π can still be used for comparing different relevance maps to each other. This results in a comparison test very similar to the test in [Sam+17]. We present our own approach to obtaining relevance maps in the next section and then come back to the relevance orderings for comparing it to other established methods in Section 7.2.

5.4 Rate-Distortion Explanation

Finding the optimal partition into S and S^c for varying distortion limits $D(S) \leq \epsilon$ is a hard combinatorial optimisation problem. To make this problem somewhat approachable, we have

[1]Fong and Vedaldi refer to these two formulations as a preservation and deletion game respectively [FV17].

to overtake two hurdles. The first is efficiently calculate the distortion $D(S)$. The second is to optimise over all possible S. We tackle the latter problem via convex relaxation of set membership that we solve with gradient descent, and the former with a technique called *Assumed Density filtering.* Both are heuristic approaches without known error bounds, but as discussed, these are either trivial or unfeasible in any way. We call this approach to obtaining relevance scores for classifier decisions *Rate-Distortion Explanation (RDE).*

5.4.1 Convex Relaxation

As we have seen in our complexity analysis, combinatorial optimisation over the all possible sets S is numerically infeasible. A common approach to handle combinatorial explosion is the following. We rewrite the set S as a binary vector $\mathbf{s}$, where

$$s_i = \begin{cases} 0 & i \notin S, \\ 1 & i \in S, \end{cases}$$

and are now looking for an optimum over the set $\{0,1\}^d$. For this nonconvex space we make a convex relaxation, by instead searching over the convex hull, which is $[0,1]^d$. In other words, instead of binary relevance decisions (*relevant* versus *non-relevant*) encoded by the set S, we allow for a continuous relevance score for each component, encoded by a vector $\mathbf{s} \in [0,1]^d$. We redefine the obfuscation of $\mathbf{x}$ with respect to $\mathbf{s}$ as a component-wise convex combination

$$\mathbf{y} = \mathbf{x} \odot \mathbf{s} + \mathbf{n} \odot (\mathbf{1}_d - \mathbf{s}) \tag{5.3}$$

of $\mathbf{x}$ and $\mathbf{n} \sim \mathcal{V}$. As before we write $\mathcal{V}_\mathbf{s}$ for the resulting distribution of $\mathbf{y}$. This is a generalisation of the obfuscation introduced in Section 5.1 which can be recovered by choosing $\mathbf{s}$ equal to one on S and zero on S^c. The natural relaxation of the set size $|S|$ is the norm $\|\mathbf{s}\|_1 = \sum_{i=1}^d |s_i|$. Our new relaxed problem amounts to

$$\text{minimise} \quad \|\mathbf{s}\|_1 \quad \text{subject to} \quad D(\mathbf{s}) \leq \epsilon,\ \mathbf{s} \in [0,1]^d. \tag{5.4}$$

Optimising over the non-convex distortion-constraint is numerically challenging, since projections back into the feasible region cannot be done in closed form. For this reason one might consider the rate-constrained problem instead:

$$\text{minimise} \quad D(\mathbf{s}) \quad \text{subject to} \quad \|\mathbf{s}\|_1 \leq k,\ \mathbf{s} \in [0,1]^d, \tag{5.5}$$

where k is the maximally allowed rate. Instead of the hard constraints on either $D(\mathbf{s})$ or $\|\mathbf{s}\|_1$, we use the continuous rate minimisation problem in its Lagrangian formulation

$$\text{minimise} \quad D(\mathbf{s}) + \lambda\|\mathbf{s}\|_1 \quad \text{subject to} \quad \mathbf{s} \in [0,1]^d \tag{5.6}$$

with a regularisation parameter $\lambda > 0$. The three formulations are of course connected, however the relation between λ, k, and ϵ that would lead to the same solutions is far from trivial, cf. Appendix C.3 for a discussion on different variants of handling the constraint.

To recover the binary decision between relevant and irrelevant parameters, we would canonically use thresholding of the continuous scores. To calculate the rate-distortion function the threshold must be chosen in a way that ensures the distortion-constrain. If π is the resulting ordering of the continuous scores, and $R_\pi(\epsilon)$ is the rate function associated with that ordering then, for a given ϵ, S^* consists of the $R_\pi(\epsilon)$ highest scoring parameters. As discussed before, the continuous scores now serve as a greedy approach to evaluate the rate-distortion function for different distortion-constraints, an interpretation that we transfer to the other relevance methods as well. Depending on the activation function, the distortion does not need to be differentiable. However, the ReLU activation is differentiable everywhere, except at zero. As commonly done during the training of neural networks, we simply use (projected) gradient descent to find a stationary point of (5.4). For this, we need an efficient way of evaluating $D(\mathbf{s})$ and its gradient.

5.4.2 Calculating the Distortion

We consider the squared distance $\operatorname{dist}(a,b) = (a-b)^2$ and define the expected distortion as before. In this case it can be rewritten in its bias-variance decomposition

$$D(\mathbf{s}) = \mathbb{E}_{\mathbf{y}\sim\mathcal{V}_\mathbf{s}}\left[(\Phi(\mathbf{x})-\Phi(\mathbf{y}))^2\right] = (\Phi(\mathbf{x}) - \mathbb{E}_{\mathbf{y}\sim\mathcal{V}_\mathbf{s}}[\Phi(\mathbf{y})])^2 + \mathbb{V}_{\mathbf{y}\sim\mathcal{V}_\mathbf{s}}[\Phi(\mathbf{y})], \tag{5.7}$$

where $\mathbb{V}$ denotes the covariance matrix.

The expected distortion is determined by the first and second moment of the output layer distribution. The exact calculation of expectation values and variances for arbitrary functions is in itself already a hard problem. One possibility to overcome this issue is to approximate the expectation by a sample mean. However, depending on the dimension d and the distribution $\mathcal{V}$ sampling might be numerically expensive. Thus, we focus on a second possibility, which takes the specific structure of Φ more into account.

From (5.3) it is straight-forward to obtain the first and second moment

$$\begin{aligned}\mathbb{E}_{\mathbf{y}\sim\mathcal{V}_\mathbf{s}}[\mathbf{y}] &= \mathbf{x}\odot\mathbf{s} + \mathbb{E}[\mathbf{n}]\odot(\mathbf{1}_d - \mathbf{s}),\\ \mathbb{V}_{\mathbf{y}\sim\mathcal{V}_\mathbf{s}}[\mathbf{y}] &= \operatorname{diag}(\mathbf{1}_d-\mathbf{s})\mathbb{V}[\mathbf{n}]\operatorname{diag}(\mathbf{1}_d-\mathbf{s}),\end{aligned}$$

of the input distribution $\mathcal{V}_\mathbf{s}$. It remains to transfer the moments from the input to the output layer of Φ. To address the challenge of efficiently approximating the expectation values in (5.7) we utilise the layered structure of Φ and propagate the distribution of the neuron activations through the network. This propagation can only be done approximately, as we prove in the next chapter 6.

The approximation method we use is called *assumed density filtering* (ADF), see for example [Min01; BK98], which has recently been used for ReLU neural networks in the context of uncertainty quantification [GR18].

Assumed Density Filtering In a nutshell, at each layer we assume a Gaussian distribution for the input, transform it according to the layers weights $\mathbf{W}$, biases $\mathbf{b}$, and activation function ϱ, and project the output back to the nearest Gaussian distribution (w.r.t. KL-divergence). This amounts to matching the first two moments of the distribution [Min01]. We now state the

ADF rules for a single network layer. Applying these repeatedly gives us a way to propagate moments through all layers and obtain an explicit expression for the distortion.

Let $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be normally distributed with some mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. An affine linear transformation preserves Gaussianity and acts on the mean and covariance in the well-known way, i.e.,

$$\mathbb{E}_{\mathbf{z}}[\mathbf{W}\mathbf{z} + \mathbf{b}] = \mathbf{W}\boldsymbol{\mu} + \mathbf{b} \quad \text{and} \quad \mathbb{V}_{\mathbf{z}}[\mathbf{W}\mathbf{z} + \mathbf{b}] = \mathbf{W}\boldsymbol{\Sigma}\mathbf{W}^*. \tag{5.8}$$

The ReLU non-linearity ϱ presents a difficulty as it changes a Gaussian distribution into a non-Gaussian one. Let f and F be the probability density and cumulative distribution function of the univariate standard normal distribution, respectively. Further, let $\boldsymbol{\sigma}$ be the vector of the diagonal entries of $\boldsymbol{\Sigma}$ and $\boldsymbol{\eta} = \boldsymbol{\mu} \oslash \boldsymbol{\sigma}$. Then, as in [GR18, Eq. 10a], we obtain

$$\mathbb{E}_{\mathbf{z}}[\varrho(\mathbf{z})] = \boldsymbol{\sigma} \odot f(\boldsymbol{\eta}) + \boldsymbol{\mu} \odot F(\boldsymbol{\eta}).$$

Unfortunately, the off-diagonal terms of the covariance matrix of $\varrho(\mathbf{z})$ are thought to have no closed form solution [FA14]. Either, we make the additional assumption that the network activations within each layer are uncorrelated. This amounts to propagating only the diagonal $\mathbb{V}_{\text{diag}}$ of the covariance matrices through the network, simplifies (5.8) to

$$\mathbb{V}_{\text{diag},\mathbf{z}}[\mathbf{W}\mathbf{z} + \mathbf{b}] = (\mathbf{W} \odot \mathbf{W})\boldsymbol{\sigma},$$

and results, as also seen in [GR18, Eq. 10b], in

$$\mathbb{V}_{\text{diag},\mathbf{z}}[\varrho(\mathbf{z})] = \boldsymbol{\mu} \odot \boldsymbol{\sigma} \odot f(\boldsymbol{\eta}) + \left(\boldsymbol{\sigma}^2 + \boldsymbol{\mu}^2\right) \odot F(\boldsymbol{\eta}) - \mathbb{E}_{\mathbf{z}}[\varrho(\mathbf{z})]^2.$$

Or, we use an approximation for the full covariance matrix

$$\mathbb{V}_{\mathbf{z}}[\varrho(\mathbf{z})] \approx \mathbf{N}\boldsymbol{\Sigma}\mathbf{N}, \tag{5.9}$$

with $\mathbf{N} = \operatorname{diag}(F(\boldsymbol{\eta}))$. This ensures positive semi-definiteness and symmetry. Depending on the network size it is usually infeasible to compute the full covariance matrix at each layer. However, if we choose a symmetric low-rank approximation factorisation $\mathbb{V}_{\mathbf{y}\sim\mathcal{V}_{\mathbf{s}}}[\mathbf{y}] \approx \mathbf{Q}\mathbf{Q}^\top$ at the input layer with $\mathbf{Q} \in \mathbb{R}^{d\times r}$ for $r \ll d$ (for example half of a truncated singular value decomposition), then the symmetric update (5.9) allows us to propagate only one of the factors through the layers. The full covariance is then solely recovered at the output layer. This immensely reduces the computational cost and memory requirement. Altogether, combining the affine linear with the non-linear transformation, implies how to propagate the first two moments through a ReLU neural network in the ADF framework. Gradients can be obtained via backpropagation.

5.5 Discussion

The introduction of the convex relaxation as well as the Assumed Density Filtering both constitute approximations for which we cannot bound the error compared to the original formulation. However, since we have proved that the original problem cannot be efficiently approximated within any non-trivial factor, we focus purely on numerical convenience in this case.

Even with these approximations the minimisation is still a non-convex problem. A solution produced by projected gradient descent will still depend heavily on the starting point, step size and number of iterations. Before we come to our numerical investigation in chapter 7, however, we want to take a brief detour to justify our use of ADF.

The application of ADF is premised on the assumption that their is no smarter distribution family that in fact is invariant to the transformation of the neural network layers. We prove this fact in the following chapter.

6 ReLU-Invariant Distributions

It is an important theoretical as well as computational challenge to calculate the distribution of outputs of a neural network from a given distribution of inputs. In chapter 5 this problem arises in our formulation for the continuous version of δ-relevant sets. However, this task is not only important for explainable AI, but also for uncertainty quantification and Bayesian learning.

6.1 Problem Formulation

We want to describe the distribution μ_{out} of

$$f(\mathbf{x}) \quad \text{with} \quad \mathbf{x} \sim \mu_{\text{in}},$$

where f is a neural network function and $\mathbf{x}$ is a random input vector distributed according to some probability measure μ_{in}. In other words, we want to determine the push-forward

$$\mu_{\text{out}} = f_* \mu_{\text{in}} \tag{6.1}$$

of μ_{in} with respect to the transformation f. Given the fact that neural networks are highly non-linear functions this generally has no closed form solution. Obtaining approximations through numerical integration methods is expensive due to the high dimensionality and complexity of the function.

This lead to the application of other approximation schemes, such as assumed density filtering [GR18; Mac+19a] (ADF) and in the case of Bayesian neural networks the more general framework of expectation propagation [JNV14; SHM14] (EP) which contains ADF as a special case. Both methods were originally developed for general large-scale Bayesian inference problems [Min01; BK98].

ADF makes use of the feed-forward network structure: If we write $f = f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1$ as a composition of its layers f_j, then (6.1) decomposes as

$$\begin{aligned} \mu_0 &= \mu_{\text{in}}, \\ \mu_j &= (f_j)_* \mu_{j-1} \quad \text{for} \quad j = 1, \dots, L, \\ \mu_{\text{out}} &= \mu_L. \end{aligned} \tag{6.2}$$

The task can then be described as how to propagate distributions through neural network layers.

If each μ_j was in the same family of probability distributions as the original μ_{in} then we could find efficient layer-wise propagation rules for each layer f_j to obtain μ_{out}. Such a family, if expressable with a possibly large but finite number of parameters, would be extremely useful for applications involving uncertainty quantification or the need for explainable predictions.

ADF commonly propagates Gaussians, or more generally exponential families, which are not invariant under the action of neural network layers with non-linear activation. Hence, the method relies on a projection step onto the chosen family of probability distributions after each network layer. In other words, (6.2) is replaced by

$$\begin{aligned} \mu_0 &= \operatorname{proj}(\mu_{\text{in}}), \\ \mu_j &= \operatorname{proj}((f_j)_* \mu_{j-1}) \quad \text{for} \quad j = 1, \dots, L, \\ \mu_{\text{out}} &= \mu_L, \end{aligned}$$

where $\operatorname{proj}(\cdot)$ is a suitable projection. In this case only $\mu_{\text{out}} \approx f_* \mu_{\text{in}}$ holds and there is no guarantee how good the approximation is after multiple layers. This heuristic is justified by the implicit assumption that no invariant families exist.

Due to the flexibility and richness of the class of neural network functions, it seems intuitively clear that such an invariant family of distributions cannot be realised in a meaningful way—a fact that until now, to the best of our knowledge, remained unproven.

In this chaper we give a complete characterisation of families of probability distributions that are invariant under the action of ReLU neural network layers. In fact, we prove that all possible invariant families belong to a set of degenerate cases that are either practically irrelevant or amount to sampling.

Hence, the implicit assumption made by the ADF and EP frameworks is justified. For each of the degenerate cases we give an explicit example of an invariant family of distributions.

Our novel proof technique is based on very intuitive geometric constructions and ideas from metric dimension theory. This intuitive argument is made rigorous using properties of the Hausdorff dimension of metric spaces. This provides a theoretical underpinning for statistical inference schemes such as ADF for neural networks.

6.2 Characterisation of Invariant Families of Distributions

Before we rigorously state and prove our main result, we start by introducing some notation and terminology and precisely specify what we mean by *parametrised families* of probability distributions and by their *invariance* with respect to layers of neural networks.

6.2.1 Preliminaries

We denote by $\mathcal{D}(\mathbb{R}^d)$ the set of Radon[1] Borel probability measures on $\mathbb{R}^d$. We equip $\mathcal{D}(\mathbb{R}^d)$ with the Prokhorov metric [Pro56].

Definition 6.1 (Family of Distributions) Let $\Omega \subseteq \mathbb{R}^n$ and $p\colon \Omega \to \mathcal{D}(\mathbb{R}^d)$. We call p an n-parameter family of probability distributions. It is called a continuous family, if p is continuous.

It might seem more intuitive to refer to the set $\{p(\theta)\}_{\theta\in\Omega}$ as the family of distributions. However, a set of distributions can be parametrised in multiple ways and even the number n of parameters describing such a set is not unique. Whenever we speak about a family of distributions, we never think of it as a mere set of distributions but always attach it to a fixed chosen parametrisation p. All our results are stated in terms of this parametrisation.

Example 6.2 (2-dimensional Gaussian Family) Consider the parameter space

$$\Omega = \mathbb{R}^2 \times \left\{(\sigma_1,\sigma_2,\sigma_3) \in \mathbb{R}^3 \ : \ \sigma_1,\sigma_2 \geq 0 \text{ and } \sigma_1\sigma_2 - \sigma_3^2 \geq 0\right\},$$

and

$$p\colon \Omega \to \mathcal{D}(\mathbb{R}^2)\colon (\mu_1,\mu_2,\sigma_1,\sigma_2,\sigma_3) \mapsto \mathcal{N}\left(\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \begin{bmatrix}\sigma_1 & \sigma_3\\ \sigma_3 & \sigma_2\end{bmatrix}\right),$$

where $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ denotes the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Then p is a continuous 5-parameter family of 2-dimensional Gaussian distributions.

Definition 6.3 (Invariance) Let $p\colon \Omega \to \mathcal{D}(\mathbb{R}^d)$ be a family of probability distributions and let $f\colon \mathbb{R}^d \to \mathbb{R}^d$ be any measurable function. Then the family is called f-invariant, if for any $\theta \in \Omega$ the pushforward of the measure $p(\theta)$ under f is again in the family, i.e. there exists $\omega \in \Omega$ such that $p(\omega) = f_*p(\theta)$. For a collection $\mathcal{F} \subseteq \{\, f\colon \mathbb{R}^d \to \mathbb{R}^d : f \text{ measurable}\,\}$ of measurable functions it is called $\mathcal{F}$-invariant, if it is f-invariant for all $f \in \mathcal{F}$.

We are interested in the special case of invariance with respect to ReLU layers of neural networks. The rectified linear unit (ReLU) [NH10; GBB11], defined as $\varrho(x) = \max\{0, x\}$, has emerged as the dominant choice for activation functions as of today [RZL17].

[1]In fact $\mathbb{R}^d$ is a separable complete metric space, thus every Borel probability measure is automatically Radon.

Definition 6.4 (ReLU-Invariance) Let $p\colon \Omega \to \mathcal{D}(\mathbb{R}^d)$ be a family of distributions. The family is called ReLU-invariant, if it is $\mathcal{F}$-invariant for the collection

$$\mathcal{F} = \{\, f\colon \mathbb{R}^d \to \mathbb{R}^d\colon \mathbf{x} \mapsto \varrho(\mathbf{W}\mathbf{x} + \mathbf{b}) \;:\; \mathbf{W} \in \mathbb{R}^{d\times d}, \mathbf{b} \in \mathbb{R}^d \,\},$$

where $\varrho(x) = \max\{0, x\}$ is applied componentwise.

We observe that if a family p is f-invariant and g-invariant for two functions f and g such that the composition $g \circ f$ is well-defined, then it is also $(g \circ f)$-invariant. In particular, a ReLU-invariant family is invariant for all ReLU networks of any depths L.

6.2.2 The Main Results

We can now state the main theorem of this work.

Theorem 6.5 *Let $\Omega \subseteq \mathbb{R}^n$ and $p\colon \Omega \to \mathcal{D}(\mathbb{R}^d)$ be a continuous and ReLU-invariant n-parameter family of probability distributions. Then at least one of the following restrictions has to hold:*

R1. Restricted Dimension: $d = 1$,

R2. Restricted Support: $\operatorname{supp}(p(\theta))$ *is finite for all* $\theta \in \Omega$,

R3. Restricted Regularity: p *is not locally Lipschitz continuous.*

This can be interpreted as follows: Besides some rather degenerate cases there can not be any family of probability distributions that is invariant with respect to the layers of a ReLU neural network. The three restrictions characterise which kind of degenerate cases can occur.

Restriction (R1) Neural networks with only one neuron per layer are not really powerful function classes, so that the ReLU-invariance is not a strong limitation in this case. However, already two dimensions are enough to unlock the expressive power of neural networks.

Restriction (R2) Under mild regularity assumptions on the parametrisation the only ReLU-invariant distributions in dimensions $d \geq 2$ are finite mixtures of Dirac distributions. This amounts to sampling distributions of finite size, which of course can work in many scenarios but are often too computationally expensive in high-dimensions.

Restriction (R3) Continuity alone is not a strong enough assumptions on the parametrisation. This is due to the fact that the class of continuous functions is too flexible and includes non-intuitive examples such as space-filing curves. These can be used to construct examples of invariant distributions. However such a kind of parametrisation has to be be rather wild and would be impractical to work with. A slightly stronger assumption like local Lipschitz continuity is enough to exclude these degenerate examples.

We complement theorem 6.5 by providing a kind of reverse statement showing that the three restrictions are individually necessary.

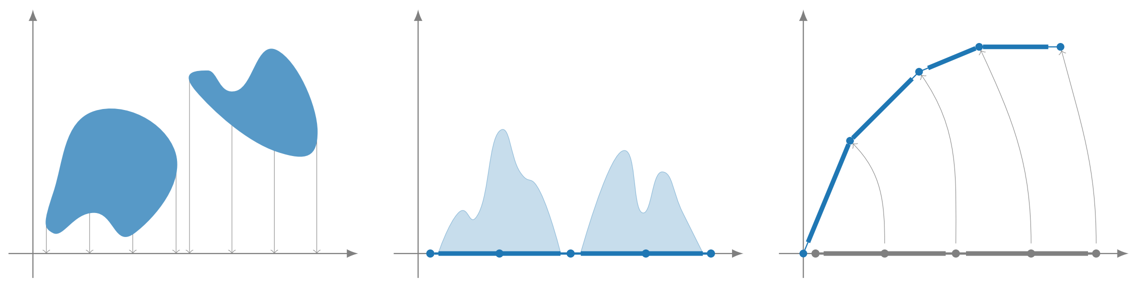


Figure 6.1: Main steps of transforming a generic probability distribution to a distribution supported on a polygonal chain. The original distribution (left) is projected onto a single dimension and partitioned into intervals (centre) that each contain a sufficiently large portion of the probability mass (shown as the lightly shaded region). Finally it is "bent" into a polygonal chain (right). The number of segments of the chain and its segment lengths can be chosen arbitrarily.

Theorem 6.6 *For each of the restrictions (R1), (R2), and (R3) from theorem 6.5 there exists a continuous ReLU-invariant n-parameter family of probability distributions that is subject to that restriction but avoids the other two.*

The proof for theorem 6.5 will be given in section 6.3 and the proof for the reverse result theorem 6.6 in section 6.4.

6.3 Proof of the Main Result (Theorem 6.5)

The main idea for proving theorem 6.5 is to use the layers of a neural network to transform simple, more or less arbitrary probability distributions to complicated distributions that need an arbitrarily high number of parameters to be described. But a family of parametrised distributions necessarily has a finite number of parameters leading to a contradiction if the family is assumed to be invariant under neural network layers. We present a high-level construction before going into detail.

More precisely, ReLU neural network layers can be used to transform arbitrary probability distributions to distributions that are supported on certain polygonal chains, which we call arcs. This is visualised in fig. 6.1. These arcs can be described by the lengths of their line segments. However, the number of segments can be made larger than the number of parameters describing the family of distributions as long as the support of the initial distribution is large enough. From this a contradiction can be derived. There are only three ways to circumvent this, corresponding to the three restrictions in theorem 6.5. Firstly, restricting the allowed neural network layers so that transformations to the arcs are not possible. Secondly, restricting the support of the distributions so that the number of line segments they can be transformed to is limited. Thirdly, using a wild parametrisation that leverages ideas similar to space-filling curves in order to use only few parameters to describe a set that effectively would require more parameters.

The idea of mapping (a subset of) the parametrisation domain to distributions supported on polygonal arcs, then to the respective arcs, and finally to segment lengths is schematically shown in fig. 6.2. We will give exact definitions of all involved spaces and mappings below.

$$\begin{tikzcd}
\Omega \supseteq \Omega_{\delta,m} \arrow[r, "\Xi"] \arrow[d, "p|_{\Omega_{\delta,m}}"'] & \mathbb{R}_+^{m-1} \times \{1\} \subseteq \mathbb{R}_+^m \\
\mathcal{D}(\mathbb{R}^d) \supseteq \mathcal{D}_{\delta,m}(\mathbb{R}^d) \arrow[r, "\mathrm{arc}"'] & \mathcal{A}_m \arrow[u, "\mathrm{scal}"']
\end{tikzcd}$$

Figure 6.2: Schematic overview of the spaces and functions involved in our proof.

6.3.1 Details of the Proof

We will now present the main steps of the proof of theorem 6.5. The proofs of the individual steps will be deferred in order to not distract from the main idea. We will assume towards a contradiction that all three restrictions in theorem 6.5 are not satisfied.

Assumption 6.7 Let $d \geq 2$, $\Omega \subseteq \mathbb{R}^n$, and $p\colon \Omega \to \mathcal{D}(\mathbb{R}^d)$ be a locally Lipschitz continuous and ReLU-invariant n-parameter family of probability distributions such that there exists a $\theta \in \Omega$ for which $\mathrm{supp}(p(\theta))$ is not finite.

From this we will derive a contradiction. The proof is based on the idea of transforming arbitrary probability distributions to distributions supported on certain polygonal chains, which we call arcs.

Definition 6.8 (Arcs) An m-arc is a subset $A \subseteq \mathbb{R}^2$ defined as a polygonal chain, i.e. a connected piecewise linear curve,

$$A = \bigcup_{i=1}^{m} \mathrm{conv}(\{\mathbf{v}_{i-1}, \mathbf{v}_i\}),$$

determined by $m+1$ vertices $\{\mathbf{v}_0, \dots, \mathbf{v}_m\}$, that satisfy

$$\mathbf{v}_i = \mathbf{v}_{i-1} + r_i \begin{bmatrix} \sin(i\phi_m) \\ \cos(i\phi_m) \end{bmatrix}, \quad i = 1, \dots, m$$

for some length scales $r_1, \dots, r_m > 0$ and the angle $\phi_m = \frac{\pi}{2m}$.

We denote the vertex set of an m-arc A as $\mathrm{vert}(A) = \{\mathbf{v}_0, \dots, \mathbf{v}_m\}$, the set of its line segments as $\mathrm{segm}(A) = \{\ell_1, \dots, \ell_m\}$, where $\ell_i = \mathrm{conv}(\{\mathbf{v}_{i-1}, \mathbf{v}_i\})$, and the set of its scaling factors as $\mathrm{scal}(A) = \{r_1, \dots, r_m\}$.

It will turn out to be useful to remove some ambiguity from the set of m-arcs by standardising their behaviour at the start and end vertices. Some examples can be seen in fig. 6.3.

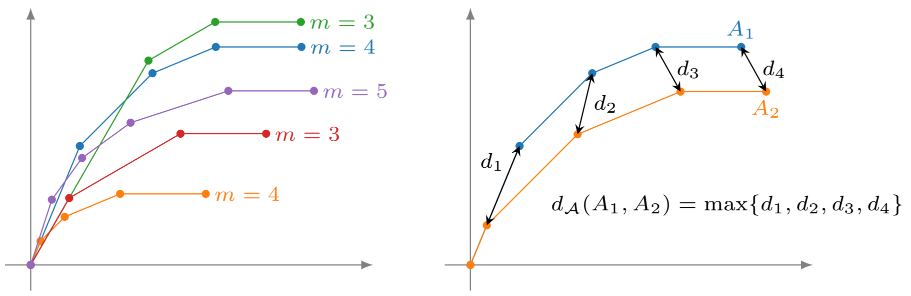


Figure 6.3: Examples of several standard m-arcs for varying m (left). Standardised arcs are always contained within the non-negative orthant and end on a horizontal line segment. The distance between two m-arcs is the maximal Euclidean distance of corresponding vertices (right).

Definition 6.9 (Standard Arcs) An m-arc A with vertices $\{\mathbf{v}_0, \ldots, \mathbf{v}_m\}$ and length scales $\{r_1, \ldots, r_m\}$ is called standardised (or a standard m-arc), if it starts at the origin and has a normalised last line segment, i.e. if $\mathbf{v}_0 = \mathbf{0}$ and $r_m = 1$. The set of all standard m-arcs is denoted $\mathcal{A}_m$ and equipped with the metric

$$d_{\mathcal{A}}(A_1, A_2) = \max_{i=1,\ldots,m} \|\operatorname{vert}(A_1)_i - \operatorname{vert}(A_2)_i\|_2.$$

The metric $d_{\mathcal{A}}$ is induced by a mixed (ℓ_2, ℓ_∞)-norm and thus indeed a proper metric. There is a one-to-one correspondence between the sets $\mathcal{A}_m$ and $\mathbb{R}_+^{m-1} \times \{1\} \cong \mathbb{R}_+^{m-1}$ via the scaling factors.

Definition 6.10 (Arc-Supported Measures) For $\delta > 0$, a probability measure $\mu \in \mathcal{D}(\mathbb{R}^2)$ is said to be δ-distributed on a standard m-arc $A \in \mathcal{A}_m$ if $\operatorname{supp}(\mu) \subseteq A$ and for each line segment $\ell \in \operatorname{segm}(A)$ we have $\mu(\ell) \geq \delta$.

For $d > 2$, a probability measure $\mu \in \mathcal{D}(\mathbb{R}^d)$ is said to be δ-distributed on a standard m-arc $A \in \mathcal{A}_m$ if $\operatorname{supp}(\mu) \subseteq \operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\}$ and by identifying $\operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\}$ with $\mathbb{R}^2$ it is δ-distributed on a standard m-arc in the above sense.

We denote the probability measures that are δ-distributed on standard m-arcs by $\mathcal{D}_{\delta,m}(\mathbb{R}^d) \subseteq \mathcal{D}(\mathbb{R}^d)$.

Lemma 6.11 Let $\mu \in \mathcal{D}_{\delta,m}(\mathbb{R}^d)$ be δ-distributed on a standard m-arc, then this arc is unique.

Proof. Towards a contradiction assume that μ is δ-distributed on $A_1, A_2 \in \mathcal{A}_m$ and $A_1 \neq A_2$. Let $\left\{\ell_1^j, \ldots, \ell_m^j\right\} = \operatorname{segm}(A_j)$ and $\left\{r_1^j, \ldots, r_m^j\right\} = \operatorname{scal}(A_j)$ denote the line segments and scaling factors of A_1 and A_2 respectively. Since $A_1 \neq A_2$ and $r_m^1 = r_m^2 = 1$ there is a smallest index $1 \leq i \leq m-1$ such that $r_i^1 \neq r_i^2$. Without loss of generality assume $r_i^1 < r_i^2$. Then ℓ_{i+1}^2 lies outside of the convex hull of A_1 and in particular

$$\ell_{i+1}^2 \cap A_1 = \emptyset.$$

But this contradicts the fact that both $\mu(\ell_{i+1}^2) \geq \delta$ and $\operatorname{supp}(\mu) \subseteq A_1$ must hold. $\square$

Lemma 6.11 shows that arc-supported measures induce unique standard m-arcs. We denote the corresponding function mapping a measure $\mu \in \mathcal{D}_{\delta,m}(\mathbb{R}^d)$ to its induced arc $A_\mu \in \mathcal{A}_m$ by $\operatorname{arc}\colon \mathcal{D}_{\delta,m}(\mathbb{R}^d) \to \mathcal{A}_m \colon \mu \mapsto A_\mu$. For an n-parameter family of distributions, $p\colon \Omega \to \mathcal{D}(\mathbb{R}^d)$, we denote by

$$\Omega_{\delta,m} = \left\{ \theta \in \Omega \,:\, p(\theta) \in \mathcal{D}_{\delta,m}(\mathbb{R}^d) \right\} \subseteq \Omega$$

the set of parameters mapping to arc-supported measures. Without further assumptions on p this set might well be empty. We will see however, that this can not happen for ReLU-invariant families as long as they contain at least one measure with a support of cardinality at least m.

We are now ready to start discussing the main steps of the proof of theorem 6.5. It relies on the following three results.

Lemma 6.12 (Surjectivity) Under assumption 6.7 and for any $m \in \mathbb{N}$ there exists a $\delta > 0$ such that the map $\Xi\colon \Omega_{\delta,m} \subseteq \mathbb{R}^n \to \mathbb{R}_+^{m-1} \times \{1\}$ given by $\Xi = \operatorname{scal} \circ \operatorname{arc} \circ p|_{\Omega_{\delta,m}}$ is surjective. In particular, in this case the domain $\Omega_{\delta,m}$ is non-empty.

The proof for this is given in appendix D.1. In short, we show that any measure with non finite support can be transformed by ReLU layers into a measure supported on an arbitrary standard arc. Then using the scaling factors to identify arcs with $\mathbb{R}_+^{m-1} \times \{1\}$ yields the claim.

Lemma 6.13 (Local Lipschitz continuity) Under assumption 6.7 and for any $m \in \mathbb{N}$ let $\delta > 0$ and $\Xi\colon \Omega_{\delta,m} \subseteq \mathbb{R}^n \to \mathbb{R}_+^{m-1} \times \{1\}$ be as in lemma 6.12. Then Ξ is locally Lipschitz continuous.

The proof for this is given in appendix D.2. In short, we show that all three partial functions scal, arc, and p are locally Lipschitz continuous, hence also their composition Ξ is.

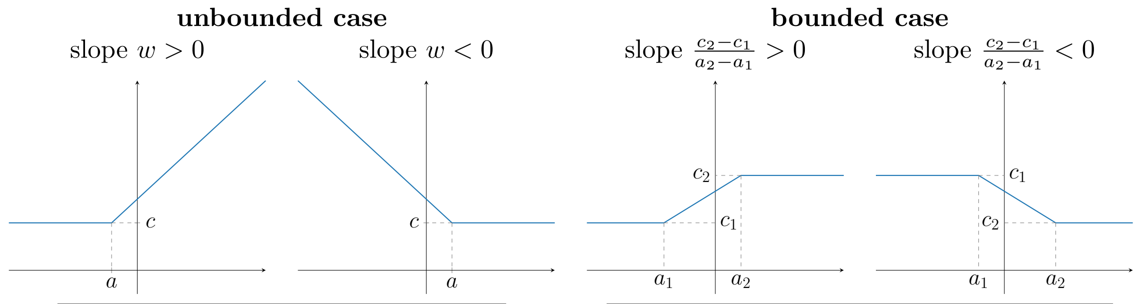
As a final piece before the main theorem we need a rather general result on maps between Euclidean spaces.

Lemma 6.14 Let $B \subseteq \mathbb{R}^n$ and $f\colon B \to \mathbb{R}^m$ be locally Lipschitz continuous. If the image $f(B)$ is a Borel set with non-empty interior in $\mathbb{R}^m$, then $m \leq n$. In particular, if f maps surjectively onto $\mathbb{R}^m$ or $\mathbb{R}_+^m$, then $m \leq n$.

The full proof of this is given in appendix D.5. It essentially uses the fact that locally Lipschitz continuous maps cannot increase the Hausdorff dimension. Since the Hausdorff dimension of B is at most n and the Hausdorff dimension of $f(B)$ is m, the claim follows.

We now have all the ingredients to prove the main theorem.

Proof of theorem 6.5. Towards a contradiction let assumption 6.7 hold. Let $m > n+1$ be arbitrary. By lemma 6.12 there exists a $\delta > 0$ so that $\Omega_{\delta,m}$ is non-empty and $\Xi\colon \Omega_{\delta,m} \to \mathbb{R}_+^{m-1} \times \{1\}$ is surjective. By lemma 6.13 it is also locally Lipschitz continuous. Since $\mathbb{R}_+^{m-1} \times \{1\}$ is isometric to $\mathbb{R}_+^{m-1}$ and $m-1 > n$ this contradicts lemma 6.14. Hence, one of the assumptions in assumption 6.7 cannot hold, which proves theorem 6.5. □



$$f(x) = \begin{cases} c, & w(x-a) \leq 0 \\ c + w(x-a), & w(x-a) > 0 \end{cases}$$

$$f(x) = \begin{cases} c_1 & x \leq a_1 \\ c_2, & x \geq a_2 \\ c_1 + \frac{c_2-c_1}{a_2-a_1}(x-a_1), & a_1 < x < a_2 \end{cases}$$

Figure 6.4: ReLU networks in dimension $d = 1$ can only take one of five distinct forms. We show the four non-constant cases. This includes two unbounded cases (left) and two bounded cases (right).

6.4 Proof of the Reverse Result (Theorem 6.6)

We will now come to the proof of theorem 6.6 and show that exploiting any of the three restrictions in theorem 6.5 indeed allows us to find ReLU-invariant families of distributions. We split the proof into three parts and construct examples for each of the three cases. Here, we briefly provide the main proof ideas and refer to appendices D.3 and D.4 for the detailed and formal proofs.

6.4.1 Families of Distributions in One Dimension

The first part of theorem 6.6 corresponds to restriction (R1). We explicitly construct a family of ReLU-invariant probability distributions on $\mathbb{R}$ with a locally Lipschitz continuous parametrisation map.

The key observation is that ReLU neural networks in one dimension are a rather restricted class of functions unlike networks in higher dimensions. In fact, every one-dimensional ReLU neural network of arbitrary depth is either constant or takes one of four forms, illustrated in fig. 6.4, and can therefore be rewritten as a three layer network. Thus, we only require a constant number of parameters to describe the set of all one-dimensional ReLU networks. It is then straightforward to obtain a ReLU-invariant family of probability distributions by explicitly making these parameters part of the family's parametrisation.

In fact, we can use a *prototype* measure $\mu_0 \in \mathcal{D}(\mathbb{R})$ with non-finite support to obtain a parametrisation map

$$p \colon \mathbb{R}^6 \to \mathcal{D}(\mathbb{R}) \colon \theta \mapsto (f_\theta)_* \mu_0,$$

where f_θ denotes a ReLU network with weights and biases $\theta = (w_1, b_1, w_2, b_2, w_3, b_3) \in \mathbb{R}^6$. This parametrisation is shown to be locally Lipschitz continuous and ReLU-invariant in appendix D.3.

6.4.2 Families of Distributions with Finite Support

The second part of theorem 6.6 corresponds to (R2). We show that ReLU-invariant families of probability distributions exist in which all distributions are finitely supported. A finitely supported Radon probability measure $\mu \in \mathcal{D}(\mathbb{R}^d)$ can be expressed as a mixture of finitely many Dirac measures,

$$\mu = \sum_{i=1}^{N} c_i \delta_{\mathbf{a}_i},$$

for some $N \in \mathbb{N}$, $\mathbf{a}_i \in \mathbb{R}^d$, and $0 \leq c_i \leq 1$ with $\sum_{i=0}^{N} c_i = 1$. For any ReLU network layer $f\colon \mathbb{R}^d \to \mathbb{R}^d\colon \mathbf{x} \mapsto \varrho(\mathbf{W}\mathbf{x} + \mathbf{b})$ we get

$$f_*(\mu) = \sum_{i=1}^{N} c_i \delta_{\varrho(\mathbf{W}\mathbf{a}_i + \mathbf{b})},$$

which is again a mixture of not more than N Dirac measures. Hence, for any $N \in \mathbb{N}$ the set $\mathcal{D}_N = \left\{ \mu \in \mathcal{D}(\mathbb{R}^d) \;:\; |\operatorname{supp}(\mu)| \leq N \right\}$ is ReLU-invariant. It can be described by $n = N(d+1)$ parameters by

$$p_N\colon \underbrace{\mathbb{R}^d \times \cdots \times \mathbb{R}^d}_{N \text{ times}} \times \Delta_N \to \mathcal{D}(\mathbb{R}^d)\colon (\mathbf{a}_1, \ldots, \mathbf{a}_N, \mathbf{c}) \mapsto \sum_{i=1}^{N} c_i \delta_{\mathbf{a}_i},$$

with $\Delta_N = \left\{ \mathbf{c} \in [0,1]^N \;:\; \sum_i c_i = 1 \right\}$. It is clear by the defintion of the Prokhorov metric that p_N is also Lipschitz continuous.

6.4.3 Families of Distributions Without Local Lipschitz Continuity

The third part of theorem 6.6 corresponds to (R3). If we omit the local Lipschitz continuity of the parametrisation we can construct a ReLU-invariant one-parameter family of distributions in $\mathbb{R}^d$ for any dimension $d \in \mathbb{N}$. We proceed similar to the ideas in section 6.4.1, however we need to take care of the fact that the set of all ReLU networks can not easily be described by a constant number of parameters, unlike in the one-dimensional case.

This issue can be overcome by using the fact that there exists a continuous[2] map $\Gamma\colon \mathbb{R} \to \mathbb{R}^\infty$, where $\mathbb{R}^\infty$ denotes the space of all real-valued and eventually vanishing sequences. We use $\mathbb{R}^\infty$ to parametrise all ReLU networks of arbitrary depths and make this a part of the parametrisation p of the distributions. This is done analogously to section 6.4.1, that is

$$p\colon \mathbb{R} \to \mathcal{D}(\mathbb{R})\colon \theta \mapsto (f_{\Gamma(\theta)})_* \mu_0,$$

where $f_{\Gamma(\theta)}$ is a network with weights and biases determined by $\Gamma(\theta)$ and $\mu_0 \in \mathcal{D}(\mathbb{R}^d)$ is some prototype measure. The detailed construction of p and proof of its ReLU-invariance can be found in appendix D.4.

[2]This uses the concept of space-filling curves and cannot be achieved with locally Lipschitz continuous functions.

6.5 Discussion

We presented a complete characterisation of distribution families that are invariant under transformations by layers of ReLU neural networks. The only invariant distributions are either sampling distributions or rather degenerate and elaborately constructed distributions that are infeasible for practical applications. This justifies the use of the ADF approximation scheme for RDE.

We have limited our analysis to the class of functions defined as layers of ReLU neural networks. Similarly, one could ask the same question for other function classes.

Considering different commonly used activation functions, such as tanh, the logistic function, or the Heaviside function, instead of ReLUs is one variation that comes to mind. Since the Heaviside function is not continuous and has a discrete and finite range, it transforms any distribution into a distribution with finite support. Clearly, the only invariant distributions can be sampling distributions. For smooth activation functions, such as tanh and the logistic function, we are not aware of any characterisation of invariant distributions. Extending our proof strategy to this scenario is not straightforward as it relies on specific properties of the ReLU. We leave this question open for future research.

A second variation on the class of functions would be to put restrictions on the weight matrices or bias vectors. This could be either general constraints, for example non-negativity or positive-definiteness, or more specific restrictions by allowing the weight matrices only to be chosen from a small set of allowed matrices. Both kinds of constraints arise in the context of iterative reconstruction methods for solving inverse and sparse coding problems [Hoy02] or in corresponding unrolled and learnable iterative algorithms [GL10; Kob+17]. Here, the restrictions on the weight matrices often stem from physical constraints of the model describing the inverse problem. We comment on both aspects of this variation in appendix D.7. Our investigation shows that for any finite or even countable set of weights and biases there exist indeed parametrisations of distributions that circumvent the three restrictions (R1)–(R3) in theorem 6.5. These, however, are purely of theoretical interest, since they rely on calculating and interpolating infinitely many distributions and are thus not of practical relevance.

We are now fully justified in using heuristics to numerically solve the rate-distortion functional that is the basis for RDE. In the following chapters we explain our numerical approach for different test datasets and compare to the most important relevance methods.

7 Numerical Comparison

RDE is strongly motivated by the formulation of δ-relevance and clearly aims at answering the questions *Q1* and *Q2*. But it remains, like all other efficient relevance mapping methods, a heuristic that cannot provably achieve this goal in all situations.

Several approaches for the evaluation of relevance mapping methods have been proposed. These include analytical tests for certain minimal requirements, e.g., invariance or covariance to simple input transformations, as well as numerical evaluations on benchmark tasks and quantitative post-hoc analyses for more realistic classification tasks. We review and discuss these evaluation approaches and present two novel tests:

We advocate for a quantitative post-hoc analysis complementing the visual evaluation of relevance maps, as also done in [Sam+17; FV17]. Relevance maps coincide with human intuition only if the relevance algorithm performs correctly and the network has learned precisely the reasoning a human would use, which is unclear in many circumstances. In fact, the relevance method should be evaluated on quantitative terms and then be used to access the reasoning of neural networks.

In addition to the quantitative evaluation and comparison tests in [Sam+17; FV17], we propose to use designed classifiers and synthetic data as baseline tests. Here, as a proof of concept, we evaluate the performance of several methods for a classification task on synthetic binary string data, where the optimal relevant sets are known.

In the numerical evaluations we compare RDE to a representative selection of established methods, namely Layer-wise Relevance Propagation (LRP) [Bac+15b], Deep Taylor decompositions [MSM18], Sensitivity Analysis [SVZ14], SmoothGrad [Smi+17], Guided Backprop [Spr+15], SHAP [LL17b], and LIME [RSG16a]. For this we use the Innvestigate[1] [Alb+18b], SHAP[2], and LIME[3] toolboxes with recommended settings. In the following, we keep the description of the numerical experiments and the choice of all hyper-parameters brief and refer to the appendices E.1 to E.4 for all details.[4]

[1]https://github.com/albermax/innvestigate
[2]https://github.com/slundberg/shap
[3]https://github.com/marcotcr/lime
[4]The code for all experiments will be made available at https://github.com/jmaces/rde upon publication.

7.1 Invariance Properties

Kindermans et al. proposed that relevance mapping methods should be invariant or covariant to simple transformations in the input domain, e.g., mean shifts, if the transformation can be compensated by the first network layer [Kin+19]. However they observed that some frequently used relevance mapping methods fail to satisfy this basic requirement. The following result shows that RDE is invariant to shifts and scalings and covariant to permutations of the input domain.

Lemma 7.1 Let $\mathbf{s}^*$ be an RDE relevance score for Φ and $\mathbf{x}$ with respect to $\mathcal{V}$, i.e., a solution of (5.4). Further let $\mathbf{P} \in \{0,1\}^{d\times d}$ be a permutation matrix, $\mathbf{\Lambda} \in (0,1]^{d\times d}$ a diagonal scaling matrix, $\mathbf{m} \in [0,1]^d$ a shift vector, and $\widetilde{\mathbf{x}} = h(\mathbf{x}) = \mathbf{P\Lambda x} + \mathbf{m}$ a transformed input. Let $\widetilde{\Phi}(\widetilde{\mathbf{x}}) = \Phi(\mathbf{\Lambda}^{-1}\mathbf{P}^\top\widetilde{\mathbf{x}} - \mathbf{\Lambda}^{-1}\mathbf{P}^\top\mathbf{m})$ be a classifier that compensates the input domain transform, i.e., $\widetilde{\Phi}(\widetilde{\mathbf{x}}) = \Phi(\mathbf{x})$, and $\widetilde{\mathcal{V}} = h_*(\mathcal{V})$ the transformed (pushforward) distribution. Then $\widetilde{\mathbf{s}}^* = \mathbf{P}\mathbf{s}^*$ is an RDE relevance score for $\widetilde{\Phi}$ and $\widetilde{\mathbf{x}}$ with respect to $\widetilde{\mathcal{V}}$.

Proof. Since $\mathbf{s} \mapsto \mathbf{Ps}$ is a bijective transform on $[0,1]^d$, it suffices to show that

$$\widetilde{D}(\mathbf{Ps}) + \lambda\|\mathbf{Ps}\|_1 = D(\mathbf{s}) + \lambda\|\mathbf{s}\|_1, \quad \text{for all} \quad \mathbf{s} \in [0,1]^d,$$

where $\widetilde{D}(\widetilde{\mathbf{s}}) = D(\widetilde{\mathbf{s}}, \widetilde{\Phi}, \widetilde{\mathbf{x}}, \widetilde{\mathcal{V}})$ denotes the distortion functional of the transformed problem. For the distortion functional, we get

$$\begin{aligned}
\widetilde{D}(\mathbf{Ps}) &= \mathbb{E}_{\widetilde{\mathbf{y}}\sim\widetilde{\mathcal{V}}_{\mathbf{Ps}}}\left[\operatorname{dist}\left(\widetilde{\Phi}(\widetilde{\mathbf{x}}), \widetilde{\Phi}(\widetilde{\mathbf{y}})\right)\right] \\
&= \mathbb{E}_{\widetilde{\mathbf{n}}\sim\widetilde{\mathcal{V}}}\left[\operatorname{dist}\left(\widetilde{\Phi}(\widetilde{\mathbf{x}}), \widetilde{\Phi}(\widetilde{\mathbf{x}} \odot \mathbf{Ps} + \widetilde{\mathbf{n}} \odot (\mathbf{1}_d - \mathbf{Ps})\right)\right] \\
&= \mathbb{E}_{\mathbf{n}\sim\mathcal{V}}\left[\operatorname{dist}\left(\widetilde{\Phi}(\mathbf{P\Lambda x} + \mathbf{m}), \widetilde{\Phi}((\mathbf{P\Lambda x} + \mathbf{m}) \odot \mathbf{Ps} + (\mathbf{P\Lambda n} + \mathbf{m}) \odot \mathbf{P}(\mathbf{1}_d - \mathbf{s})\right)\right] \\
&= \mathbb{E}_{\mathbf{n}\sim\mathcal{V}}\left[\operatorname{dist}\left(\widetilde{\Phi}(\mathbf{P\Lambda x} + \mathbf{m}), \widetilde{\Phi}(\mathbf{P\Lambda}(\mathbf{x} \odot \mathbf{s} + \mathbf{n} \odot (\mathbf{1}_d - \mathbf{s})) + \mathbf{m}\right)\right] \\
&= \mathbb{E}_{\mathbf{n}\sim\mathcal{V}}\left[\operatorname{dist}\left(\Phi(\mathbf{x}), \Phi(\mathbf{x} \odot \mathbf{s} + \mathbf{n} \odot (\mathbf{1}_d - \mathbf{s})\right)\right] \\
&= \mathbb{E}_{\mathbf{y}\sim\mathcal{V}_{\mathbf{s}}}\left[\operatorname{dist}\left(\Phi(\mathbf{x}), \Phi(\mathbf{y})\right)\right] \\
&= D(\mathbf{s}),
\end{aligned}$$

where we used the definition of the pushforward distribution $\widetilde{\mathcal{V}}$ in the third equation, properties of the Hadamard product in the fourth equation, and the definition of $\widetilde{\Phi}$ in the fifth equation. Clearly, the equality also holds for the ℓ_1-norm term. □

In other words, shifting and scaling of the inputs does not affect the RDE relevance scores, whereas a permutation of the input components results in the same permutation of the relevance scores, as can be expected of a reasonable relevance mapping method.

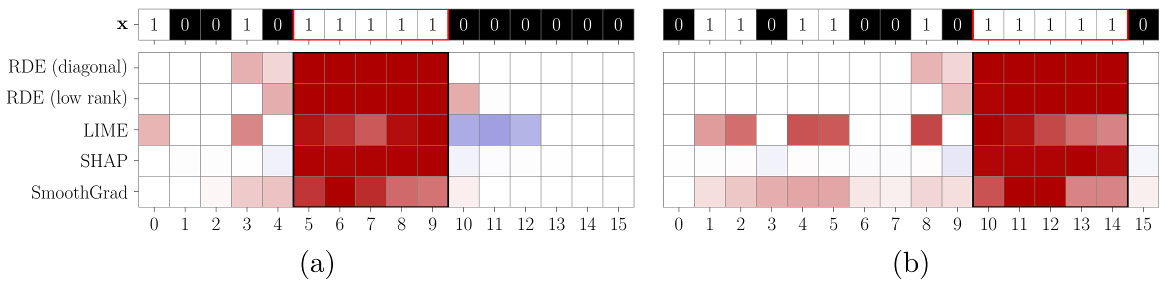


Figure 7.1: Relevance mappings generated by several methods for two binary strings. The left string (a) contains one block of five consecutive ones whereas the right string (b) contains one complete and one incomplete block. The colourmap indicates positive relevances as red and negative relevances as blue. All methods clearly identify the correct block as relevant in the example (a). RDE, SHAP and SmoothGrad identify the correct block as most relevant in the example (b). For SmoothGrad the distinction from the incomplete block is less pronounced.

7.2 Numerical Comparisons

In the following we present a series of numerical investigations, based both on the proposal in [OWT19] and [Sam+17], as well as experiments with a designed network, where the relevant part is know a priori and neural networks trained on established image recognition tasks.

7.2.1 Synthetic Binary Strings

The above invariance test is merely a minimum requirement for relevance mapping methods but far from sufficient. As an additional baseline, we propose to test relevance mapping methods on a synthetic binary classification task where the optimal relevant sets are known. We consider the Boolean function

$$\Psi\colon \{0,1\}^d \to \{0,1\}, \quad \mathbf{x} \mapsto \bigvee_{i=1}^{d-k+1} \bigwedge_{j=i}^{i+k-1} x_j,$$

that checks binary strings of length d for the existence of a block of k consecutive ones.

If an input signal $\mathbf{x}$ contains a unique set of k consecutive ones, then it is clear that these variables are relevant for the classification. More precisely, the smallest rate that can achieve distortion zero is k and in fact any set S containing the block of k consecutive ones will achieve it. On the other hand any smaller set of size $|S| < k$ will have distortion at least $\frac{1}{2}$.

We can construct a ReLU neural network $\Phi\colon [0,1]^d \to [0,1]$ that interpolates Ψ, see Appendix E.1 for details. Relying on the connection between the binary and continuous setting established in Section 4.3 we expect that a relevance mapping method should also find the block of k consecutive ones as most relevant for Φ.

We test this for two input signals of size $d = 16$ each containing a block of $k = 5$ consecutive ones. The first has no disjoint other group of five consecutive variables that is even close to being a block of ones, see Figure 7.1(a). The second also has a disjoint second group of five consecutive variables that almost forms a block of ones (four of the five are ones), see Figure 7.1(b).

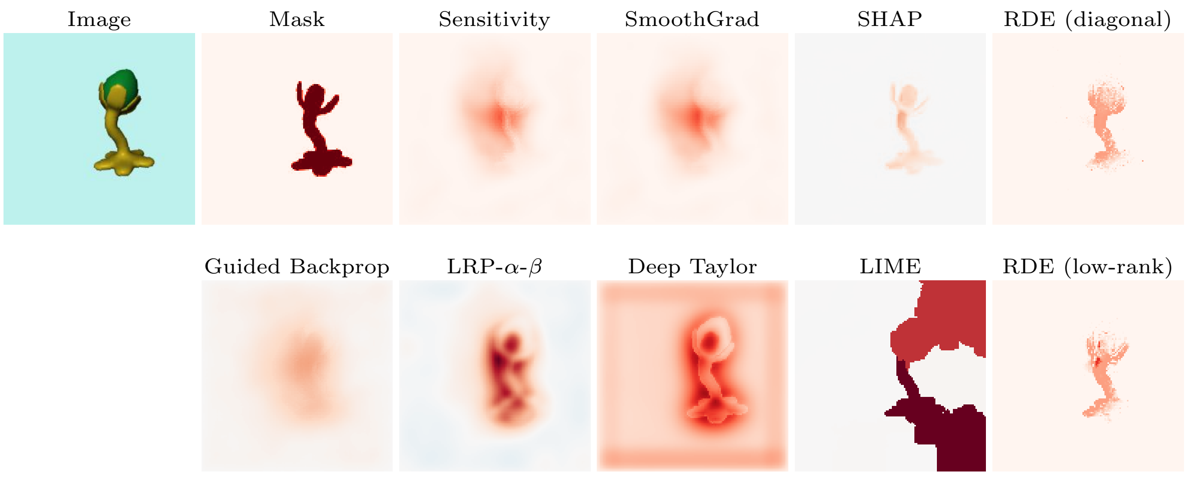


Figure 7.2: Relevance mappings generated by several methods for an image from the an8flower benchmark dataset classified as *yellow stem* by our network. The colourmap indicates positive relevances as red and negative relevances as blue.

In this setting we compare RDE to SmoothGrad, SHAP, and LIME. All other methods are excluded from the comparison as they produce constant relevance mappings for this particular neural network and thus fail to identify the relevant block.

RDE, SHAP, and SmoothGrad identify the correct block as relevant in both cases, whereas LIME identifies the correct block in the first case but gets distracted by the incomplete block in the second case, see Figure 7.1.

7.2.2 An8flower Benchmark Dataset

Oramas et al. proposed the benchmark dataset *an8flower* for the evaluation of relevance mapping methods for image classifiers [OWT19]. It consists of synthetic images of plants in various positions and rotations that are to be classified by the colour of their flower or stem. The dataset also provides binary masks marking the coloured part of the plant that is responsible for its class. These masks are considered as the “ground truth” explanations in [OWT19]. Consequently, relevance mapping methods are evaluated by measuring the correlation between their relevance scores and the respective binary mask.

The dataset is synthetic, however the classifier Φ is trained on data samples and not constructed, unlike in the synthetic binary string experiment. Hence, the true underlying reasoning of the classifier remains unknown, and it is unclear whether the provided masks can represent a true explanation of the classifiers decisions. Nevertheless, we want to include this test in our analysis, as it can be seen as an intermediate step between the synthetic binary string setup (fully constructed) and the post-hoc relevance ordering test discussed in the next section (based on real-world data). We trained a convolutional neural network (three convolution layers each followed by average-pooling and finally two fully-connected layers and softmax output) on the an8flower dataset end-to-end up to a test accuracy of 0.99.

The relevance mappings for one example image of a plant with yellow stem are shown in fig. 7.2. The mappings are calculated for the pre-softmax score of the class with the highest activation. The performance of the relevance mapping methods with respect to two different similarity measures averaged over 12 examples from the dataset (one for each class) is

	Sensitivity	SmoothGrad	SHAP	Guided Backprop	LRP-α-β	Deep Taylor	LIME	RDE (diag)	RDE (low-rank)
Pearson Corr.	0.14 ± 0.11	0.13 ± 0.11	0.14 ± 0.10	0.09 ± 0.12	0.21 ± 0.24	-0.02 ± 0.26	0.20 ± 0.23	0.23 ± 0.16	$\mathbf{0.27 \pm 0.16}$
Jaccard Index	0.11 ± 0.07	0.12 ± 0.07	0.12 ± 0.07	0.06 ± 0.04	0.10 ± 0.08	0.04 ± 0.05	0.12 ± 0.10	0.14 ± 0.09	$\mathbf{0.16 \pm 0.08}$

Table 7.1: Similarity between relevance mappings generated by several methods and the respective binary masks for the an8flower dataset with respect to Pearson correlation coefficient and Jaccard index. Values closer to 1 mean more similar in both measures. Results show the mean ± standard deviation over 12 images from the test set (1 image per class).

summarised in table 7.1. Both variants of RDE achieve the best results for both the Pearson correlation coefficient as well as the Jaccard index, with a slight advantage for the low-rank RDE variant over the diagonal RDE variant.

7.2.3 Relevance Ordering Test

Next, we generate relevance mappings for greyscale images of handwritten digits from the MNIST dataset [LeC+98] as well as colour images from the STL-10 dataset [CNL11]. The true optimal relevant sets are not known for image classifiers trained on these tasks. Therefore, in the spirit of Section 5.3, we propose a variant of the *relevance ordering*-based test introduced in [Sam+17] for a fair comparison of the methods.

We sort components according to their relevance score (breaking ties randomly). Then, starting with a completely random signal, we replace increasingly large parts of it by the original input, and observe the change in the classifier score. This is then averaged over multiple random input samples. A good relevance mapping will lead to a fast convergence to the classification score of the original signal when the most relevant components are fixed first. In other words the distortion quickly drops to zero. The described process allows us to approximately evaluate the rate-distortion function. It corresponds to calculating the inverse of the rate function $R_\pi(\epsilon)$ associated to the relevance ordering π, as discussed in Section 5.3.

7.2.3.1 MNIST Experiment

We trained a convolutional neural network (three convolution layers each followed by average-pooling and finally two fully-connected layers and softmax output) end-to-end up to a test accuracy of 0.99.

The relevance mappings for one example image of the digit six are shown in Figure 7.3. The mappings are calculated for the pre-softmax score of the class with the highest activation. Both variants of our proposed method generate similar results and highlight an area at the top that distinguishes the digit six from, for example, the digits zero and eight. The relevance-ordering test results are shown in Figure 7.5 (left). We observe that the expected distortion drops fastest for our proposed method indicating that the most relevant components were correctly identified.

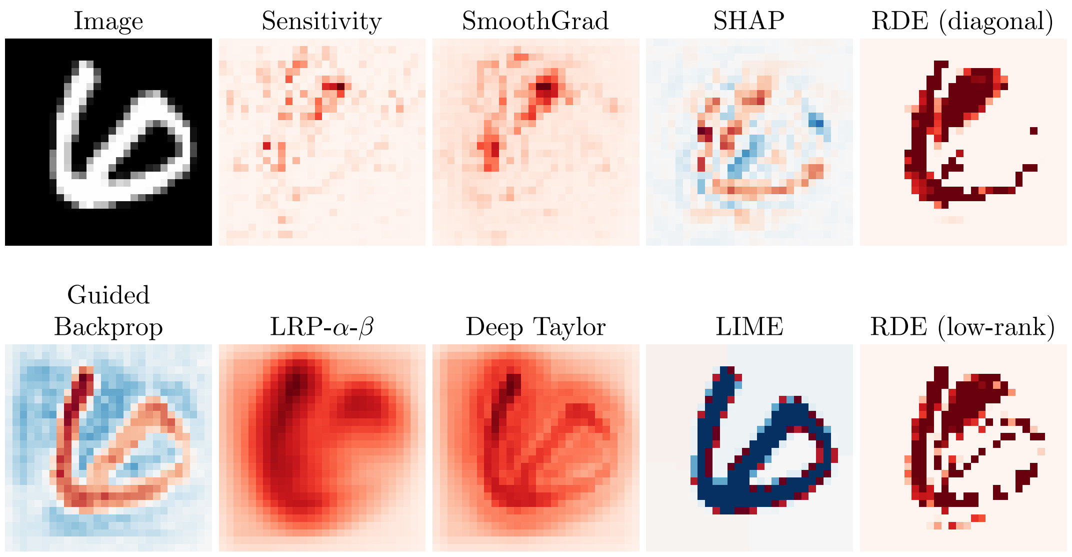


Figure 7.3: Relevance mappings generated by several methods for an image from the MNIST dataset classified as *digit six* by our network. The colourmap indicates positive relevances as red and negative relevances as blue.

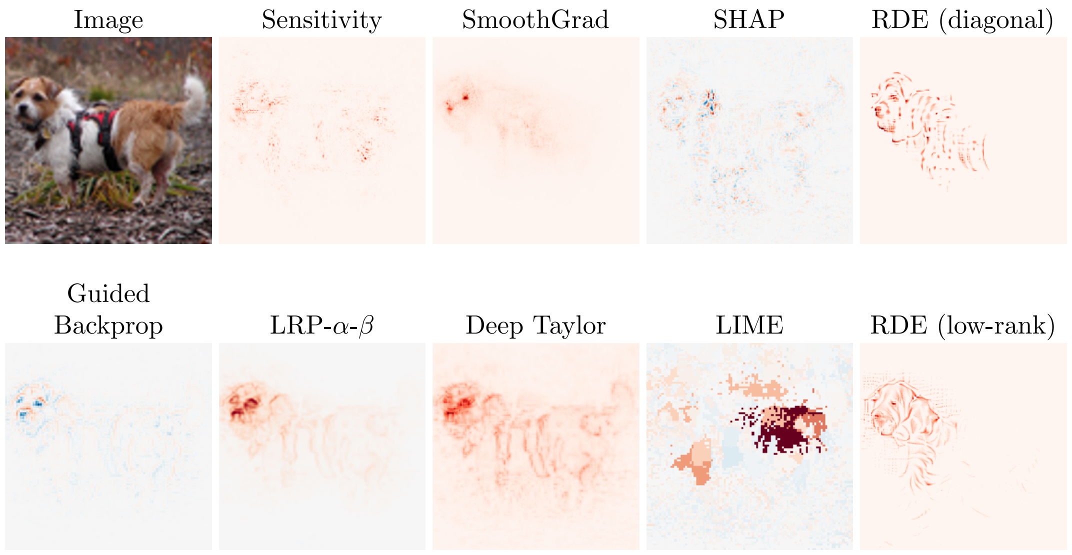


Figure 7.4: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *dog* by our network. The colourmap indicates positive relevances as red and negative relevances as blue.

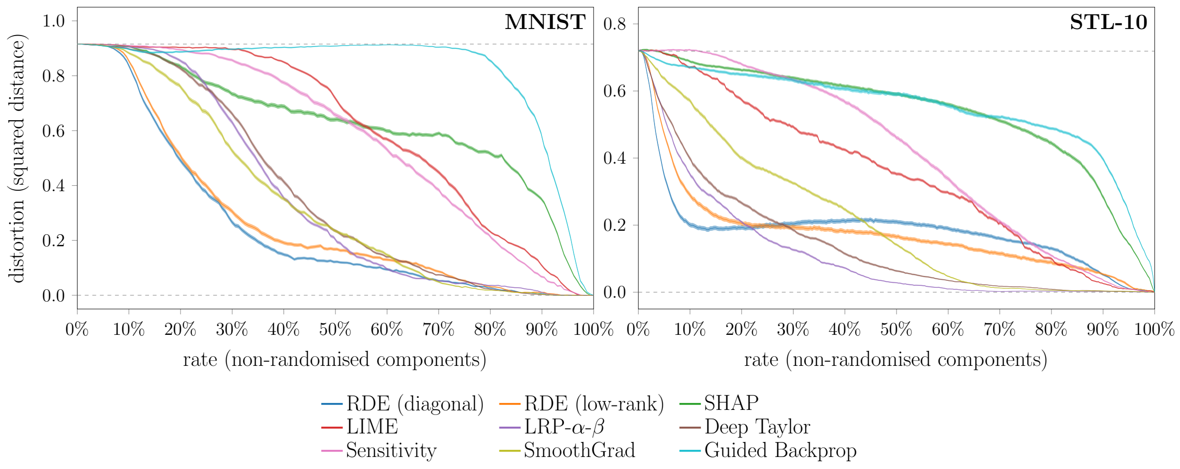


Figure 7.5: Relevance ordering test results (approximate rate-distortion function) of several methods for MNIST and STL-10. An average result over 50 images from the respective test set (5 images per class) and 512 (MNIST) and 64 (STL-10) noise input samples per image is shown (shaded regions mark ± standard deviation).

7.2.3.2 STL-10 Experiment

We use a VGG-16 network [SZ14] pre-trained on the Imagenet dataset and refined it on the STL-10 dataset to a final test accuracy of 0.935.

The relevance mappings for one example image of a dog are shown in Figure 7.4. As before the mappings are calculated for the pre-softmax score of the class with the highest activation. The difference between both variants of our proposed method is more pronounced here. The low-rank variant captures finer details, for example in the dog's face. The relevance-ordering test results are shown in Figure 7.5 (right). We observe that although our method does not obtain the smallest expected distortions across all rates it has the fastest dropping distortion for low rates indicating that the most relevant components were correctly identified. This is to be expected as our method generates sparse relevance scores highlighting only few of the most relevant components and is not designed for high rates.

7.3 Discussion

In the last chapters we showed that even for the very basic questions *Q1* and *Q2* that we posed in chapter 4, no efficient algorithm can be proven to generally answer them. The implication of these results was that we can only rely on informed heuristics and thorough numerical evaluation—both of which we provide for the Rate-Distortion Explanations.

We have shown that RDE compares very favourably to the most important competing methods. It fulfills the invariance requirements proposed by Kindermans et al [Kin+19]. It also succeeds on the binary strings experiment along with only SHAP and SmoothGrad. The significance for this test is that we have explicitely designed the neural network to find consecutive blocks of ones, so we know a priori what the relevant part of the input is.

The rate-distortion plots for the MNIST and STL10 data sets reveal that our method finds the most relevant features, especially for small rates. For the more complicated STL10 dataset the method selects features for large rates that give worse distortion than its competetitors, the

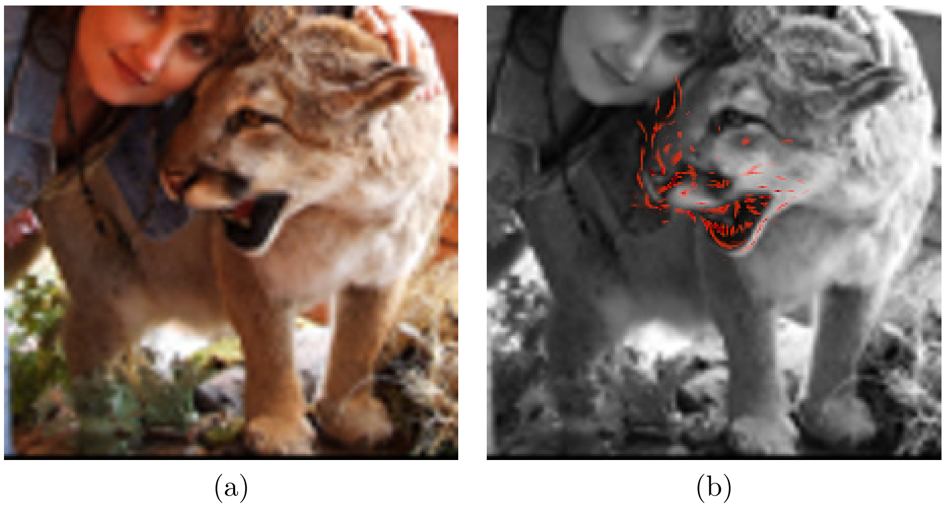


Figure 7.6: We overlay an image from the STL10 dataset showing a cougar from the class "cat" (a) with the relevance map produced by RDE (diagonal) (b). The relevance map overemphasises a tooth structure as well as a stripe pattern that isn't visible in the original image.

"diagonal" RDE are even slightly worse than for smaller rates. Transmitting more information should intuitively always lead to less distortion. What is going on there?

Nonmonotonic Rate-Distortion-Functions Intuitively, it makes no sense that transmitting more information should lead to a greater distortion. Taking our intuition from lossy compression, optimal or even sensible compression schemes always have monotonically decreasing rate-distortion curves [BM19].

One explanation for this behaviour is the presence of counterfactual evidence in the STL10 images, e.g. certain background features that are more correlated to a different class. Transmitting more pixel values and thus including these features can make the classifier less sure of the true class. We believe that this can only be a minor effect and we have found strong evidence of another phenomenon being the reason. Upon careful examination of the individual relevance maps our method produced, we noticed that occasionally the map would form features that are not apparent in the original image. This effect only appeared only STL-10 and not in MNIST. See fig. 7.6 as an example. These "superstimulous" maps, optimised to force a classification from the network with few pixels shown will perform worse for higher rates since the artificial features will disappear again.

The same effect was demonstrated in [MBP21]. See especially figure 13, where a large black bird wing gets masked to show a birds face instead, and figure 7, where a new horse head appears in the mask.

We will discuss in the next chapter how this is the result of our selected distribution $\mathcal{V}_s$ (see eq. (5.7) in chapter 5) that uses uniform noise to complete the image. We devoted the next chapter to find a solution that avoids these "superstimulous" maps, deriving a formalism that under mild assumptions guarantees that the produced features indeed contain the class information.

8 Arthur-Merlin Regularisation

Our numerical investigation in the previous chapter left us with a puzzle. How can it be that for the relevance orderings we derived with RDE more information about the image lead to worse classification accuracy? In this chapter, we want to reexamine the use of distributions that are independent from the masked part of the image. We will explain how the nonmonotonicity of the rate-distortion curve arises from "superstimulus" masks that operate similar to adversarial examples. This effect arises when we evaluate the network on input that lies off the data distribution that the network had been trained on.

8.1 The True Data Distribution

As we explained in the introduction, partial-input methods rely on so called characteristic functions $\nu\colon \mathcal{P}([d]) \to \mathbb{R}$, i.e. a function that can assign a value to any subset of the parameters. In the case of Shapley values the resulting relevances for an input parameter x_i is expressed as

$$r_i = \frac{1}{d} \sum_{S \subseteq [d] \setminus \{i\}} \binom{d-1}{|S|}^{-1} (\nu(S \cup \{i\}) - \nu(S))$$

where one sums over all possible coalitions. In case of our rate-distortion approach, we instead search for a small coalition that returns a value close to the original one:

$$S^* = \underset{|S| \leq k}{\operatorname{argmin}} \operatorname{dist}(\nu([d]), \nu(S)),$$

where $k \in [d]$ is a cap on the set size and dist is an appropriate distance measure. Practical methods proposed so far [Mac+19b; FV17] rely on a continuous relaxation of set membership expressed as a vector $\mathbf{s} \in [0,1]^d$ and an extension of the characteristic function to partial set membership, so $\nu\colon [0,1]^d \to \mathbb{R}$. They interpret the resulting optimum as the relevance scores for each feature, i.e.

$$r_i = s_i^*.$$

The problem is: These definitions rely on the notion of a value function that can be evaluated for partial input–which doesn't exist! Classifiers are only defined and trained on full data points—not partial input.

A reasonable solution, put forth by [LL17a], is to regard the missing parameters as random variables and take an expectation value conditioned on the given parameters. Let $f : D \to \{-1, 1\}$ be a classifier function for a two-class data set $D \subseteq \mathbb{R}^d$ with distribution $\mathcal{D}$. Then we can naturally define a characteristic function $\nu_{f,\mathbf{x}} : \mathcal{P}([d]) \to [-1, 1]$ as

$$\nu_{f,\mathbf{x}}(S) = \mathbb{E}_{\mathbf{y}\sim\mathcal{D}}[f(\mathbf{y}) \,|\, \mathbf{y}_S = \mathbf{x}_S] = \int f(\mathbf{x})p(\mathbf{x}_{S^c} \,|\, \mathbf{x}_S)\, \mathrm{d}\mathbf{x}_{S^c},$$

which is the expectation value of f conditioned on $\mathbf{x}_S$. The problem is that we do not know the true conditional data distribution $p(\mathbf{x}_{S^c} \,|\, \mathbf{x}_S)$ which would be equivalent to knowing the true data distribution $p(X = \mathbf{x})$.

A number of recent investigations showed that two functions that evaluate identically on $\mathrm{supp}(\mathcal{D})$ can nevertheless generate vastly different explanations when the explanation method cannot model the distribution correctly. The authors of [And+20] demonstrate this effect for sensitivity analysis, LRP, and Guided Backprop. They are able to manipulate relevance scores at will and demonstrate how this can be used to obfuscate discrimination inside of a model. LIME and SHAP can be manipulated as well [Sla+20] by using a classifier that behaves differently outside off-distribution if the wrong distribution for the explanation.

For Shapley values specifically, [Fry+20] demonstrates with multiple examples that for correlated features the Shapley values that assume an independent data distribution can give wrong explanations. They propose two solutions: Use a generative model for the data distribution or directly train a characteristic function to handle partial information. The authors of [AJL19] instead propose a weakly data informed distribution (normally distributed with mean and variance from the data set) as well as a a non-parametric Nadaraya-Watson estimator with a Kernel on partial data points from the test and training set. Both papers demonstrate improved Shapley values compared with approaches that assume independence.

We will discuss the existing approaches and explain how they still run into problems. Figure 8.1 gives a quick overview over the two main issues.

Taking an independent distribution:

$$p(X_{S^c} = x_{S^c} \mid X_S = x_S) = \prod_{i \in S^c} p(x_i)$$

This approach has been used in our investigation and similarly in [FV17] and [MBP21]. As we have seen, employing an optimisation method with this distribution can result in "superstimulus" masks– mask that generate new features that weren't there. This is especially prominent in [MBP21] in figure 13, where a bird head appears in the masking of a large monocrome bird wing.

Cutting a specific shape out of a monochrome background will with high likelihood result in an image where this shape is visible, see illustration in fig. 8.1. If the distribution was true, a monochrome shape would likely lead to a completion that is also monochrome in the same colour, destroying the artificial feature. But an independent distribution underrepresents

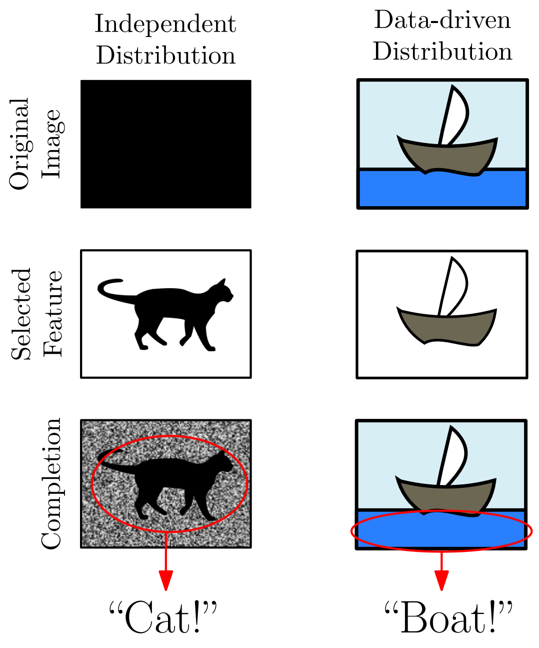


Figure 8.1: Different failure modes of unrepresentative distributions. *Left:* From a black image the shape of a cat is selected and the rest is filled with uniform noise. The shape of a cat is detected by a classifier. *Right:* An image of a ship is given. The ship-feature is selected. The data driven distribution inpaints the water back into the image, since in the dataset ships are always on water. The faulty classifier that relies on the water feature is undetected as the ship-feature indirectly leads to the correct classification.

these reasonable correlations. This phenomenon will make a "superstimulus" masks not only possible, but even the most likely optimiser for the problem.

Taking a data-determined distribution via generative model

$$p(X_{S^c} = x_{S^c} \mid X_S = x_S) = G(x_{S^c}; x_S)$$

Generative models as a means to approximate the data distribution in the context of explainability have been proposed in a series of publications [AN20; Cha+18; Liu+19; Mer+20].

This has two problems. First, for optimisation-based explanations an adversarial mask might be less likely, but is still possible. The hidden space of the generator is smaller than the image space, but still large. Second, we cannot be smarter than the data distribution. The cases where the network contradicts human intuition and does not generalise are likely introduced by biases in our dataset. But these biases will be learned by the generator as well (see fig. 8.1 for an illustration)! The important cases will be exactly the kind of cases that we will not be able to detect. If the generator has learned that horses and image source tags are highly correlated, it will inpaint an image source tag when a horse is present. This allows the network to classify correctly, even when the network only looks for the tag and has no idea about horses. The faulty distribution overrepresents correlations that aren't there in real live.

Inpainting and Blurring Alternative approaches rely on general knowledge of the data space without being specific to the test and training data. E.g. for images this is high local correlations or sparse representation in a suitable dictionary like wavelets or shearlets. This allows for inpainting techniques such as k-nearest pixels or sparsity regularisation in the shearlet domain.

The authors of [FV17] use blurring to obscure masked features and employ smoothness priors for the optimisation of their masks. Masking by blurring relies on the implicit knowledge that informative features have at least some higher frequency components.

However this is still not a solution to the problem for the following reasons: The approach is only applicable to data with a kind of locality structure like images and sounds. It is relatively time-consuming as an operation and both Shapley-based as well as Optimisation based approaches require many evaluation of $\hat{f}$ for different masks. But finally it still pushes the resulting picture far from the training set and can still be exploitable by an optimiser.

8.2 From Feature Spaces to Average Precision

Let us start our investigation with some basic definitions that we need for this chapter. The first thing we want to define is the notion of a *feature space*. What reasonably constitutes a feature depends on what kind of data we are considering. In images, it might make sense to regard translations or rotation of a subset of pixels as essentially the same feature. One could also only consider connected subsets of pixels or only up to a certain size. Thus we leave the definition of a feature abstract and regard it simply as a set of data points.

Definition 8.1 (Feature Space) Given a set D, we call a set $\Sigma \subseteq 2^D$ a feature space on D if

- $\cup\Sigma = D$, i.e. all data points have at least one feature
- $\varnothing \in \Sigma$, i.e. the empty set is contained as a default feature that appears in no data point
- $\max_{\mathbf{x}\in D}|\{\phi \in \Sigma \,|\, \mathbf{x} \in \phi\}| < \infty$, i.e. there are only finitely many features per data point.

We say a data point $\mathbf{x}$ has or contains a feature ϕ if $\mathbf{x} \in \phi$. We also say that a set of features $F \subset \Sigma$ *covers* the data points in $\bigcup_{\phi\in F}\phi$ which we abbreviate as $\cup F$.

Example 8.2 (Features as partial vectors of size up to k) For $k \in [d]$ we define a feature space corresponding to partial vectors with support up to size k subsets as

$$\Sigma = \bigcup_{\mathbf{x}\in D} \bigcup_{\substack{S\subset[d]\\ |S|\le k}} \{\{\mathbf{y} \in D \,|\, \mathbf{y}_S = \mathbf{x}_S\}\} \cup \{\varnothing\}.$$

One important practical example of a feature space is given in example 8.2, which is related to the parameter subsets we have considered in chapters 4, 5 and 7. This feature space corresponds to collections on pixels as part of an image, and every data point $\mathbf{y}$ that has the same values $\mathbf{y}_S$ on a certain subset of indices S is part of the same feature. Though this feature space for images lacks any invariance, such as translations, and changing a single pixel changes the feature, we will nevertheless use this example when imagining practical feature spaces and explicitly use it in our application section 8.8.

Definition 8.3 (Two-class Data Space) We call the tuple $((D, \sigma, \mathcal{D}), c, \Sigma)$ a two-class data space when

- $D \subseteq [0,1]^d$ is the set of possible data points,
- σ is a σ-Algebra on D,
- $\mathcal{D} : \sigma \to [0,1]$ is a probability measure, on D representing the data distribution,
- $c : D \to \{-1, 1\}$ is the true classification, function
- Σ is a feature space on D.

The *class imbalance* B is defined as

$$B := \max_{l \in \{-1,1\}} \frac{\mathbb{P}_{\mathbf{x}\sim\mu}[c(\mathbf{x}) = l]}{\mathbb{P}_{\mathbf{x}\sim\mu}[c(\mathbf{x}) = -l]}$$

Remark 8.4 We will oftentimes make use of restrictions of the set D and measure $\mathcal{D}$ to a certain class, e.g. $D_l = \{\mathbf{x} \in D \,|\, c(\mathbf{x}) = l\}$ and $\mathcal{D}_l = \mathcal{D}|_{D_l}$.

When can we say that a feature truly carries the class information? Surely, a relevant feature ϕ should reduce the entropy of the class to a minimum. Let $X \sim \mathcal{D}$ and $C = c(X)$ be random variables representing a data point and its class. For a single feature $\phi \in \Sigma$ the conditional class entropy is

$$H_{\mathcal{D}}(C \,|\, X \in \phi) = \sum_{l \in \{-1,1\}} \mathbb{P}(C = l \,|\, X \in \phi) \log(\mathbb{P}(C = l \,|\, X \in \phi)).$$

The quality of a feature is thus measured by how small it makes the class entropy. Let us translate this measure to a given explanation method. For this we introduce the notion of a *feature selector*.

Definition 8.5 (Feature Selector) For a given data set D, we call M a feature selector, if

$$M : D \to \Sigma \quad \text{s.t.} \quad \forall \mathbf{x} \in D : \mathbf{x} \in M(\mathbf{x}) \vee M(\mathbf{x}) = \varnothing$$

which means that for every data point M selects a feature that is present in it or returns the empty set as a default. We call $\mathcal{M}(D)$ the space of all feature selectors for a dataset D.

For a given a feature selector M the feature $M(X)$ becomes a random variable as well. Thus $C = c(X)$ and $M(X)$ share a non-trivial common probability measure and we can define the mutual information between the two as

$$I_{\mathcal{D}}(C, M(X)) = H_{\mathcal{D}}(C) - H_{\mathcal{D}}(C \mid M(X)).$$

The mutual information becomes maximal, when the conditional entropy be $H(C \mid M(X))$ becomes minimal. The pure class entropy $H(C)$ measures how uncertain we are about the class a priori. If the class imbalance B is high, we know the class with high probability in advance, thus the class entropy will be low and the measure of the mutual information becomes less useful. From now on we focus on the conditional

Per definition of the conditional entropy, see eq. (2.1) in chapter 2, we can expand

$$\begin{aligned} H_{\mathcal{D}}(C \mid M(X)) &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[H(C \mid M(\mathbf{x}))] \\ &= -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\sum_{l\in\{-1,1\}} P(C = l \mid M(\mathbf{x})) \log(P(C = l \mid M(\mathbf{x})))\right] \\ &= -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\sum_{l\in\{c(\mathbf{x}),-c(\mathbf{x})\}} P(C = l \mid M(\mathbf{x})) \log(P(C = l \mid M(\mathbf{x})))\right] \\ &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[H_b(P(C = c(\mathbf{x}) \mid M(\mathbf{x}))] \end{aligned}$$

where $H_b(p) = -p\log(p) - (1-p)\log(1-p)$ is the binary entropy function. Using that $H_b(p)$ is a concave function together with Jensen's inequality, we can pull in the expectation and arrive at the bound

$$H_{\mathcal{D}}(C \mid M(X)) \leq H_b(\mathrm{ap}_{\mathcal{D}}(M)),$$

where ap, the *average precision* is defined as follows:

Definition 8.6 (Average Precision) For a given two-class data space $((D, \sigma, \mathcal{D}), c, \Sigma)$ and a feature selector $M \in \mathcal{M}(D)$ we define the average precision ap of M with respect to $\mathcal{D}$ as

$$\mathrm{ap}_{\mathcal{D}}(M) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbb{P}_{\mathbf{y}\sim\mathcal{D}}[c(\mathbf{y}) = c(\mathbf{x}) \mid \mathbf{y} \in M(\mathbf{x})]]$$

A large $\mathrm{ap}_{\mathcal{D}}(M)$ tells us that with high probability M will produce a feature that determines the class with high probability. We will later derive bounds in terms of $\mathrm{ap}_{\mathcal{D}}(M)$, and since $H_b(p) \to 0$ as $p \to 1$, our bounds on g will translate into bounds of the conditional entropy.

Lemma 8.7 Given a two-class data space $((D, \sigma, \mathcal{D}), c, \Sigma)$, a feature selector $M \in \mathcal{M}(D)$ and $\delta \in [0, 1]$, we can state that: With probability $1 - \delta^{-1}(1 - \mathrm{ap}_{\mathcal{D}}(M))$ we select a feature $M(\mathbf{x})$ where $\mathbf{x} \sim \mathcal{D}$, such that

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{D}}[c(\mathbf{y}) = c(\mathbf{x}) \mid \mathbf{y} \in M(\mathbf{x})] \geq 1 - \delta.$$

Proof. The proof follows directly from Markov's inequality which states that for a nonnegative random variable Z and $\delta > 0$

$$\mathbb{P}[Z \geq \delta] \leq \frac{\mathbb{E}[Z]}{\delta}.$$

Choosing $Z = 1 - \mathbb{P}_{\mathbf{y}\sim\mathcal{D}}[c(\mathbf{y}) = c(\mathbf{x}) \,|\, \mathbf{y} \in M(\mathbf{x})]$ with $\mathbf{x} \sim \mathcal{D}$ leads to the result. □

We will now introduce a new framework that will allow us to prove bounds on $\mathrm{ap}_{\mathcal{D}}(M)$ and thus assure feature quality.

8.3 Arthur-Merlin Regularisation

The intuition for Arthur-Merlin regularisation comes from interactive proof systems, namely the Arthur-Merlin protocol. There, a verifier with poly-time resources, named Arthur, can exchange messages with an all-powerful but untrustworthy prover, named Merlin, to solve a decision problem.

8.3.1 The Arthur-Merlin Protocol

Whenever Arthur is confronted with a 'Yes'-instance of the problem, a cooperative Merlin should be able to convince him almost surely to accept, a property called *completeness*. Whenever a 'No'-instance is presented, then Arthur will reject the proofs of an antagonistic Merlin with high probability, a property called *soundness*.

This problem class is powerful enough to contain NP-complete problems like SAT. If the instance is satisfiable, Merlin can convince Arthur by presenting a satisfying assignment that Arthur can check in poly-time. Otherwise, he is unable to produce an assignment that could convince Arthur.

We can use the same setup to provide guarantees for our explanations, see fig. 8.2 for an illustration. Let us call our classifier network A for Arthur and our features selector M for Merlin. First, we look at a purely collaborative scenario. Merlin has to select a feature from an image, set the rest of the image to black and give it to Arthur who has to classify it. Let's assume that Merlin always produces features of maximum size k and that Arthur always answers the correct class of the image. Does that mean that the feature Merlin selected contains the class information? Surprisingly no. When Merlin and Arthur cooperate, they can come up with a strategy to cheat. Say, whenever the image shows a dog, Merlin selects a non-black pixel of the left half as feature and whenever a cat, he selects a pixel on the right half. Arthur can then classify correctly with $k = 1$. However, the class entropy conditioned on the single pixel will not be small. This can indeed happen, when Merlin is an optimisation method and Arthur is a very high-dimensional classifier.

A Min-Max principle Now, we extend the scenario to include an adversarial version of Merlin, $\widehat{M}$ (we call her *Morgana* to extend the metaphor). Morgana will also give masked images to Arthur for classification, but with the intention of making him state the wrong class. Arthur does not know whether the image comes from Merlin or Morgan. He is allowed to say "I don't know" if he cannot discern the class, but should never give a wrong answer, the *soundness criterion*. When the image comes from Merlin, he must answer the correct class, which is the *completeness criterion*.

It turns out, as we will later rigorously prove, that the only non-exploitable strategy that Arthur and Merlin can pursue is to exchange features that are indeed highly indicative of the true class label with respect to the data distribution. The criteria on completeness and

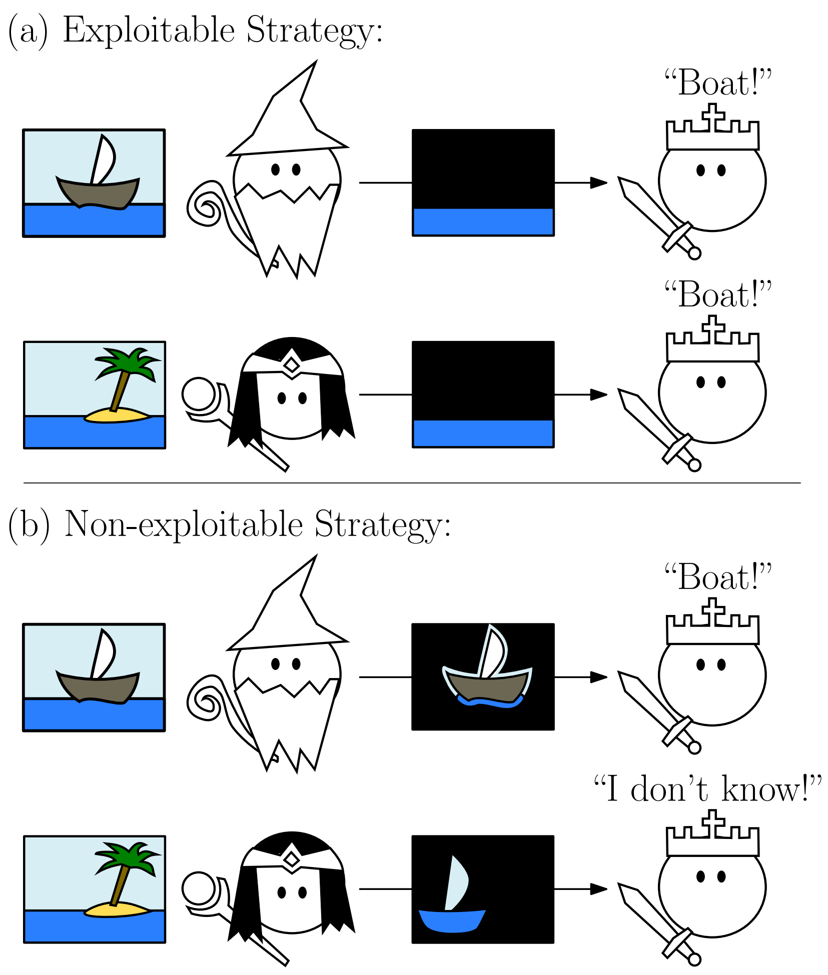


Figure 8.2: *(a):* To prove to Arthur that the image is of the "boat" class, Merlin sends the water-feature to Arthur. This can happen, when the training data set almost always shows water and boats together. This strategy can be exploited by Morgana by selecting the water from an image of a different class, here "island". *(b):* In this case, Merlin sends a feature containing the boat to Arthur. Even though Morgana cuts out the shape of a boat in her image, Arthur has learned to require the actual boundaries of the shapes to be part of the feature.

soundness result in guarantees for the explanation, without intricate schemes to fill up the masked image.

The intuition for this is: Whenever Merlin and Arthur agree to use a feature ϕ for class l albeit not being indicative of l then ϕ must also be present in images of class $-l$, which Morgana can then select from an image $\mathbf{x}$ with $c(\mathbf{x}) = -l$ to fool Arthur to give the wrong label.

Definition 8.8 (Partial Classifier) A partial classifier for a two-class data manifold is a function $A : \Sigma \to \{-1, 0, 1\}$. The option of outputting 0 corresponds to the option of answering "I don't know". We call the space of all partial classifiers $\mathcal{A}$.

For a partial classifier A and two feature selectors $M, \widehat{M}$ we define

$$E_{M,\widehat{M},A} = \left\{ x \in D \,\middle|\, A(M(\mathbf{x})) \neq c(\mathbf{x}) \vee A\Big(\widehat{M}(\mathbf{x})\Big) = -c(\mathbf{x}) \right\}, \tag{8.1}$$

the set of data points for which Merlin fails to convince Arthur of the correct class or Morgana is able to trick him into giving the wrong class, in short the set of points where Arthur and Merlin fail. We can state the following theorem that connects a competitive game to an information theoretic quantity.

Theorem 8.9 (Min-Max) *Let $M \in \mathcal{M}(D)$ be a feature selector and let*

$$\epsilon_M = \min_{A\in\mathcal{A}} \max_{\widehat{M}\in\mathcal{M}} \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}\Big[\mathbf{x} \in E_{M,\widehat{M},A}\Big].$$

Then there exists a set D' with $\mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x} \in D'] \geq 1 - \epsilon_M$ such that for $\mathcal{D}' = \mathcal{D}|_{D'}$ we have

$$ap_{\mathcal{D}'}(M) = 1 \quad \textit{and thus} \quad H_{\mathcal{D}'}(C, M(X)) = 0.$$

Proof. From the definition of ϵ_M it follows that there exists a not necessarily unique $A^\sharp \in \mathcal{A}$ such that

$$\max_{\widehat{M}\in\mathcal{M}} \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}\Big[\mathbf{x} \in E_{M,\widehat{M},A^\sharp}\Big] = \epsilon_M. \tag{8.2}$$

Given $A^\sharp$, the optimal strategy by Morgana is certainly

$$\widehat{M}^\sharp(\mathbf{x}) = \begin{cases} \phi \text{ s.t. } A(\phi) = -c(\mathbf{x}) & \text{if possible,} \\ \varnothing & \text{otherwise.} \end{cases}$$

and every optimal strategy differs only on a set of measure zero. Thus we can define

$$D' = D \setminus E_{M,\widehat{M}^\sharp,A^\sharp}$$

and have $\mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x} \in D'] \geq 1 - \epsilon_M$. We know that $A(M(\mathbf{x})) \neq 0$ when $\mathbf{x} \in D'$ and thus can finally assert that

$$\forall \mathbf{x}, \mathbf{y} \in D' : \mathbf{y} \in M(\mathbf{x}) \Rightarrow c(\mathbf{y}) = c(\mathbf{x}),$$

since otherwise there would be at least one $\mathbf{y} \in D'$ that would also be in $E_{M,\widehat{M}^\sharp,A^\sharp}$. Thus $g_{\mathcal{D}'}(M) = 1$ and from

$$0 \leq H_{\mathcal{D}'}(C, M(X)) \leq H_b(g_{\mathcal{D}'}(M)) = 0$$

it follows that $H_{\mathcal{D}'}(C, M(X)) = 0$. □

This theorem states that if Merlin uses a strategy that allows Arthur to classify almost always correctly, thus small ϵ_M, then there exists a data set that covers almost all of the original data set and on which the class entropy conditioned on the features selected by Merlin is zero.

This statement with a new set D' appears convoluted at first, and we would prefer a simple bound such as

$$\text{ap}_{\mathcal{D}}(M) \geq 1 - \epsilon_M,$$

where we take the average precision over the whole data set. This is, however, not straightforwardly possible due to a principle we call *asymmetric feature correlation* and which we introduce in the next section.

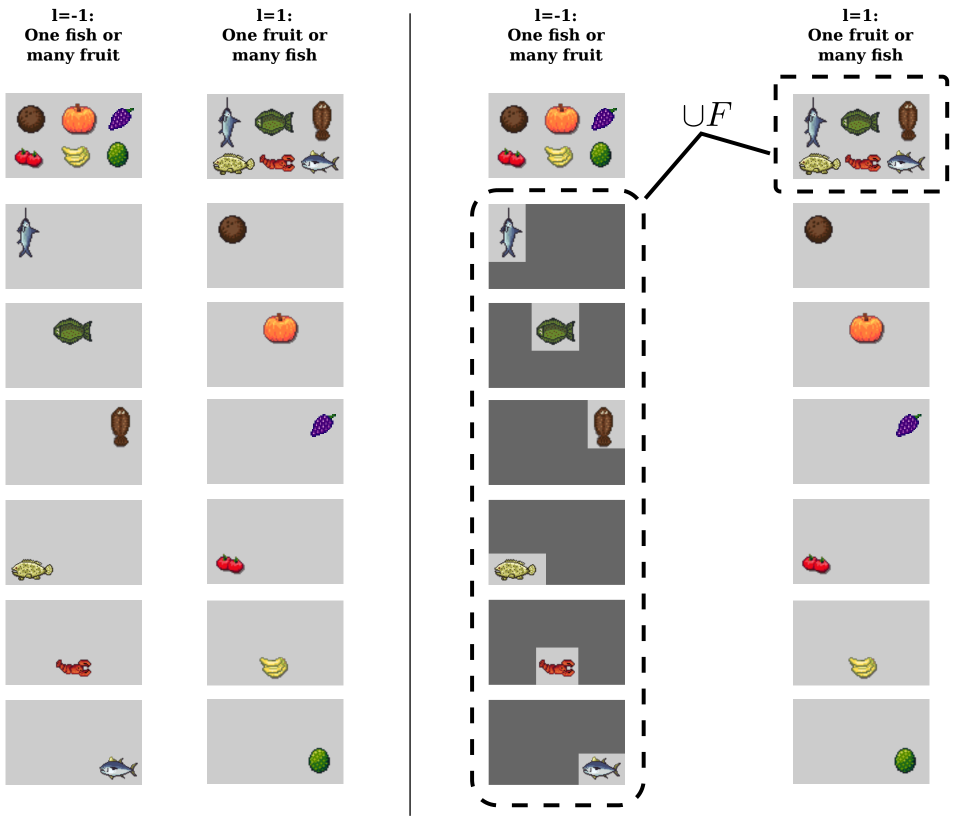

l=-1: One fish or many fruit
l=1: One fruit or many fish
l=-1: One fish or many fruit
l=1: One fruit or many fish
$\cup F$

Figure 8.3: Illustration of a data space with strong asymmetric feature concentration. *Left:* A data set with fish and fruit features. The features are asymmetrically correlated, because all the fruit features are maximally correlated in class -1 (they are all in the same image) and maximally uncorrelated in 1 (no two fruits share the same image). The reverse is true for the fruits. See fig. 8.4 for a strategy for Merlin that ensures strong completeness and soundness with uninformative features.
Right: Asymmetric feature correlation for a specific feature selection. For the class -1 we select the set F of all "fish" features. Each individual fish feature in F covers a fraction of $\frac{1}{6}$ of $(\cup F) \cap D_{-1}$ and all images (one) in $(\cup F) \cap D_1$. The expectation value in eq. (8.3) thus evaluates to $k = 6$. This is also the maximum AFC for the whole data set as no different feature selection gives a higher value.

8.4 Asymmetric feature Correlation

Asymmetric feature correlation (AFC) is a concept that will allow us to state our main result. It measures if there is a set of features that are much stronger correlated on data points of one class than of the other. This represents a possible quirk of data sets that can result in a scenario where Arthur and Merlin cooperate successfully with high probability, Morgana is unable to fool Arthur except with low probability–and yet the features exchanged by Arthur and Merlin are uninformative for the class. An illustration of such an unusual data set is given in fig. 8.3.

For an illustration of the asymmetric feature correlation consider two-class data space $\{(D, \sigma, \mathcal{D}), c, \Sigma\}$, e.g. the "fish and fruit" data depicted in fig. 8.3. Let us choose $F \subset \Sigma$ to be all the "fish" features. We see that these features are strongly anti-correlated in class $l = -1$ (none of them share an image) and strongly correlated in class $l = 1$ (all of them are in the same image).

For now, let us assume F is finite and let $\mathcal{F} = \mathcal{U}(F)$, the uniform measure over F.

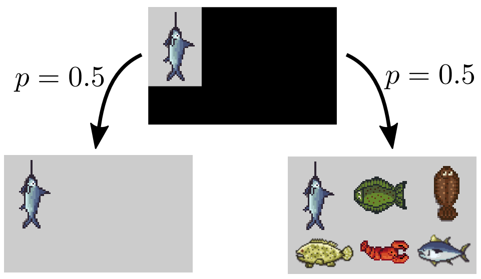


Figure 8.4: In the data set presented in fig. 8.3, Merlin can use the strategy to always select the fish features for class $l = -1$ and the fruit features for class $l = 1$ if they exist and choose something arbitrary otherwise. Arthur can then guess $l = 1$ if he gets a fish and $l = -1$ for a fruit. This strategy fails only for the images containing all fruits or fish and can only be exploited by Morgana for those same two images out of 14. The completeness and soundness constants in this case are both $\frac{1}{7}$. But as illustrated here, each "fish" feature is completely uninformative of the class. Conditioned on the selected fish it could either be the image from class $l = -1$ or from $l = 1$.

We have strong AFC if the class-wise ratio of what each feature covers individually is much larger than what the features cover as a whole:

$$\mathbb{E}_{\phi\sim\mathcal{F}}\left[\frac{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[\mathbf{x}\in\phi]}{\mathbb{P}_{\mathbf{x}\sim\hat{\mathcal{D}}_{l}}[\mathbf{x}\in\phi]}\right] \gg \frac{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[\mathbf{x}\in\cup F]}{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{l}}[\mathbf{x}\in\cup F]}.$$

In our example, every individual "fish" feature covers one image in each class, so the left side is equal to 1. As a feature set, they cover 6 images in class -1 and one in class 1, so the right side is $\frac{1}{6}$. Using

$$\frac{\mathbb{P}[\mathbf{x}\in\phi]}{\mathbb{P}[\mathbf{x}\in\cup F]} = \mathbb{P}[\mathbf{x}\in\phi\,|\,\mathbf{x}\in\cup F],$$

we can restate this expression as

$$\mathbb{E}_{\phi\sim\mathcal{F}}[k_l(\phi, F)] \gg 1, \tag{8.3}$$

where

$$k_l(\phi, F) = \frac{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[\mathbf{x}\in\phi\,|\,\mathbf{x}\in\cup F]}{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{l}}[\mathbf{x}\in\phi\,|\,\mathbf{x}\in\cup F]}.$$

For our "fish" features we have $k(\phi, F) = 6$ for every feature $\phi \in F$. Considering infinite set F, we need a way to get a reasonable measure $\mathcal{F}$, where we don't want to "overemphasise" features that are appear only in very few data points. We thus define a feature selector $f_F \in \mathcal{M}(\cup F)$ as

$$f_F(\mathbf{x}) = \underset{\substack{\phi\in F\\ \text{s.t. } \mathbf{x}\in\phi}}{\operatorname{argmax}}\, k(\phi, F) \tag{8.4}$$

and we can define $\mathcal{F} = f_{F*}\mathcal{D}_l|_{\cup F}$. For our fish and fruit example, where F is the set of all fish features, f_{F*} would select a fish feature for every image in class -1 that is in $\cup F$.

Putting everything together we get the following definition.

Definition 8.10 (Asymmetric Feature Correlation) Let $((D, \sigma, \mathcal{D}), c, \Sigma)$ be a two-class data space, then the asymmetric feature correlation $k > 0$ is defined as

$$k = \max_{l \in \{-1,1\}} \max_{F \subset \Sigma} \mathbb{E}_{\mathbf{y} \sim \mathcal{D}_l|_{\cup F}} \left[\max_{\substack{\phi \in F \\ \text{s.t. } \mathbf{y} \in \phi}} \frac{\mathbb{P}_{\mathbf{x} \sim \mathcal{D}_{-l}}[\mathbf{x} \in \phi \,|\, \mathbf{x} \in \cup F]}{\mathbb{P}_{\mathbf{x} \sim \mathcal{D}_l}[\mathbf{x} \in \phi \,|\, \mathbf{x} \in \cup F]} \right].$$

As we have seen in the "fish and fruit“ example, the AFC can be made arbitrarily large, as long as one can fit many individual features into a single image. We can prove that the maximum amount of features per data point indeed also gives an upper bound on the AFC constant.

Definition 8.11 Let $((D, \sigma, \mathcal{D}), c, \Sigma)$ be a two-class data space. The maximum amount of features in a data point $K > 0$ is defined as

$$K = \max_{\mathbf{x} \in D} |\{\phi \in \Sigma \,|\, \mathbf{x} \in \phi\}|$$

Lemma 8.12 Let $((D, \sigma, \mathcal{D}), c, \Sigma)$ be a two-class data space with AFC constant k. Let K be the maximal number of features per data point. Then

$$k \leq K.$$

Proof. Let $l \in \{-1, 1\}$ and let $F \subset \Sigma$. We define $f_F \in \mathcal{M}(\cup F)$ as in eq. (8.4) as well as $\mathcal{F} = f_{F*}\mathcal{D}_l|_{\cup F}$. We can assert that

$$\mathbb{P}_{\mathbf{x} \sim \mathcal{D}_l}[\mathbf{x} \in \phi \,|\, \mathbf{x} \in \cup F] \geq \mathbb{P}_{\mathbf{x} \sim \mathcal{D}_l}[f(\mathbf{x}) = \phi \,|\, \mathbf{x} \in \cup F] = \mathcal{F}(\phi). \tag{8.5}$$

We then switch the order of taking the expectation value in the definition of the AFC constant:

$$\begin{aligned} k_l(f, \hat{D}) &= \mathbb{E}_{\phi \sim \mathcal{F}} \left[\frac{\mathbb{P}_{\mathbf{x} \sim \mathcal{D}_{-l}}[\mathbf{x} \in F \,|\, \mathbf{x} \in \cup F]}{\mathbb{P}_{\mathbf{y} \sim \hat{\mathcal{D}}_l}[\mathbf{y} \in \phi \,|\, \mathbf{y} \in \cup F]} \right] \\ &= \mathbb{E}_{\phi \sim \mathcal{F}} \left[\frac{\mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{-l}}[\mathbb{1}(\mathbf{x} \in \phi) \,|\, \mathbf{x} \in \cup F]}{\mathbb{P}_{\mathbf{y} \sim \hat{\mathcal{D}}_l}[\mathbf{y} \in \phi \,|\, \mathbf{y} \in \cup F]} \right] \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{-l}} \left[\mathbb{E}_{\phi \sim \mathcal{F}} \left[\frac{\mathbb{1}(\mathbf{x} \in \phi)}{\mathbb{P}_{\mathbf{y} \sim \hat{\mathcal{D}}_l}[\mathbf{y} \in \phi \,|\, \mathbf{y} \in \cup F]} \right] \middle| \, \mathbf{x} \in \cup F \right] \end{aligned}$$

Since there are only finitely many features in a data point we can express the expectation value over a countable sum weighted by the probability of each feature:

$$\begin{aligned} k_l(f,\hat{D}) &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{-l}}\left[\sum_{\phi\in F:\,\mathbf{x}\in F}\left[\frac{\mathcal{F}(\phi)}{\mathbb{P}_{\mathbf{y}\sim\hat{\mathcal{D}}_l}[\mathbf{y}\in\phi\,|\,\mathbf{y}\in\cup F]}\right]\middle|\,\mathbf{x}\in\cup F\right] \\ &\leq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{-l}}\left[\sum_{\phi\in F:\,\mathbf{x}\in F} 1\,\middle|\,\mathbf{x}\in\cup F\right] \\ &\leq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{-l}}[K\,|\,\mathbf{x}\in\cup F] \\ &= K, \end{aligned}$$

where in the first step we used eq. (8.5) and the definition of K in the second. □

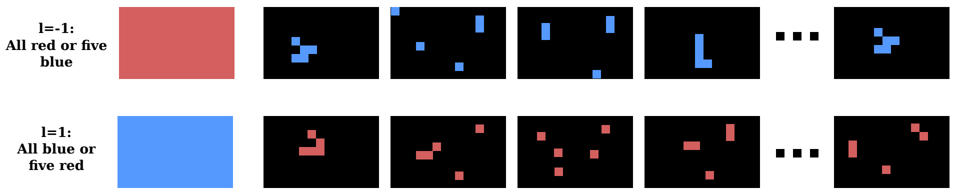


Figure 8.5: An example of a data set with very high asymmetric feature correlation. The completely red image shares a feature with each of the m-red-pixel images (here $m=5$), of which there are $\binom{d}{m}$ many. In the worst case $m=\frac{d}{2}$, resulting in $k=\binom{d}{d/2}$ thus exponential growth in d.

The number of features per data point is dependent on which kinds of features we consider. Without limitations this number can be 2^d, i.e. exponentially high. See fig. 8.5 for an example of exponentially large AFC parameter. If we consider, for example for image data, only features of a fixed size and shape the number of features per data point drops to $\approx d$.

8.5 Main Theorem

We are now able to state our main theorem.

Theorem 8.13 *Let $((D,\sigma,\mathcal{D}),c,\Sigma)$ be a two-class data space with asymmetric feature concentration of k and class imbalance B. Let $A:[0,1]^d\to\{-1,0,1\}$ be a partial classifier and $M\in\mathcal{M}(D)$ a feature selector for D. From*

1. *Completeness:*

$$\min_{l\in\{-1,1\}}\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[A(M(\mathbf{x}))=l]\geq 1-\epsilon_c,$$

2. *Soundness:*

$$\max_{\widehat{M}\in\mathcal{M}(D)}\max_{l\in\{-1,1\}}\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}\left[A\left(\widehat{M}(\mathbf{x})\right)=-l\right]\leq\epsilon_s,$$

follows

$$ap_{\mathcal{D}}(M)\geq 1-\epsilon_c-\frac{k\epsilon_s}{1-\epsilon_c+k\epsilon_s B^{-1}}.$$

The theorem gives a bound on the probability that data points with features selected by Merlin will also belong to the same class. This probability is large as long as we have a bound on the AFC of the data set.

Proof. The proof of our lemma is fairly intuitive, although the notation can appear cumbersome. Here we give a quick overview over the proof steps.

1. In the definition of the AFC, we maximise over all possible features sets. We will choose as a special case (for each class $l \in \{-1, 1\}$) the features that Merlin selects for data points that Arthur classifies correctly.

2. These feature sets cover the origin class at least up to $1 - \epsilon_c$, and the other class at most up to ϵ_s, which is required by the completeness and soundness criteria respectively. This gives us a high precision for the whole feature set.

3. The AFC constant upper bounds the quotient of the precision of the whole feature set and expected precision of the individual features, which finally allows us to state our result.

Let us define a partition of D according to the true class and the class assigned by Arthur. From now on let $l \in \{-1, 1\}$ and $m \in \{-1, 0, 1\}$. We introduce the data sets

$$D_{l,m} = \{\mathbf{x} \in D_l \mid A(M(\mathbf{x})) = m\},$$

which means that $D_{l,l}$ are the datapoints that Arthur classifies correctly, and furthermore

$$F_l = M(D_{l,l}).$$

To use the the AFC bound we need a feature selector $f : D_l|_{\cup F} \to F$. Merlin itself maps to features outside of F when applied to data points in $D_l|_{\cup F_l} \setminus D_{l,l}$. Let us thus define $\sigma : D_F \setminus D_{l,l} \to F$ which selects an arbitrary feature from F (in case one is concerned whether such a representative can always be chosen, consider a discretised version of the data space which allows for an ordering). Then we can define

$$f(\mathbf{x}) = \begin{cases} M(\mathbf{x}) & \mathbf{x} \in D_{l,l}, \\ \sigma(\mathbf{x}) & \mathbf{x} \in D_l|_{\cup F_l} \setminus D_{l,l}, \end{cases} \quad \text{and} \quad \mathcal{F}_l = f_* D_l|_{\cup F_l}.$$

This feature selector will allow us to use the AFC bound. We now abbreviate

$$p(\mathbf{x}, f) = \mathbb{P}_{\mathbf{y} \sim \mathcal{D}}[c(\mathbf{y}) \neq c(\mathbf{x}) \mid \mathbf{y} \in f(\mathbf{x})] \quad \text{and} \quad P_l = \mathbb{P}_{\mathbf{x} \sim \mathcal{D}}[\mathbf{x} \in D_l].$$

Then

$$1 - \mathrm{ap}_{\mathcal{D}}(M) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[p(\mathbf{x}, M)] = \sum_{l \in \{-1,1\}} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_l}[p(\mathbf{x}, M)] P_l. \tag{8.6}$$

Using the completeness criterion and the fact that $p(\mathbf{x}, M) \leq 1$ we can bound

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, M)] &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, M)\mathbb{1}(\mathbf{x} \in D_{l,l})] + \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, M)\mathbb{1}(\mathbf{x} \notin D_{l,l})] \\
&\leq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, M)\mathbb{1}(\mathbf{x} \in D_{l,l})] + \epsilon_c \\
&\leq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, M)\mathbb{1}(\mathbf{x} \in D_{l,l})] + \epsilon_c + \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, \sigma)\mathbb{1}(\mathbf{x} \in D_l|_{\cup F_l} \setminus D_{l,l})] \\
&\leq \frac{\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, M)\mathbb{1}(\mathbf{x} \in D_{l,l}) + p(\mathbf{x}, \sigma)\mathbb{1}(\mathbf{x} \in D_l|_{\cup F_l} \setminus D_{l,l})]}{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[\mathbf{x} \in D_l|_{\cup F_l}]} + \epsilon_c \\
&= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l|_{\cup F_l}}[p(\mathbf{x}, f)] + \epsilon_c \\
&= \mathbb{E}_{\phi\sim\mathcal{F}_l}[\mathbb{P}_{\mathbf{y}\sim\mathcal{D}}[c(\mathbf{y}) = -l \mid \mathbf{y} \in \phi]] + \epsilon_c.
\end{aligned}$$

We can expand the expression in the expectation using Bayes' Theorem:

$$\begin{aligned}
\mathbb{P}_{\mathbf{y}\sim\mathcal{D}}[c(\mathbf{y}) = -l \mid \mathbf{y} \in \phi] &= \frac{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y} \in \phi]\mathbb{P}_{-l}}{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y} \in \phi]P_{-l} + \mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y} \in \phi]P_l} \\
&= h\left(\frac{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y} \in \phi]\mathbb{P}_{-l}}{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y} \in \phi]P_l}\right),
\end{aligned}$$

were $h(t) = (1 + t^{-1})^{-1}$. Since $h(t)$ is a concave function for $t \geq 0$, we know that for any random variable R have $\mathbb{E}[h(R)] \leq h(\mathbb{E}[R])$, so

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, M)] \leq h\left(\mathbb{E}_{\phi\sim\mathcal{F}_l}\left[\frac{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y} \in \phi]}{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y} \in \phi]}\right]\frac{P_{-l}}{P_l}\right) + \epsilon_c. \tag{8.7}$$

From the definition of the AFC constant k we know that

$$\mathbb{E}_{\phi\sim\mathcal{F}_l}\left[\frac{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y} \in \phi]}{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y} \in \phi]}\right] \leq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l|_{\cup F_l}}\left[\max_{\substack{\phi\in F \\ \text{s.t. } \mathbf{x}\in\phi}} \frac{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y} \in \phi]}{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y} \in \phi]}\right] \leq k\frac{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y} \in \cup F]}{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y} \in \cup F]}. \tag{8.8}$$

Now we make use of the fact that we can lower bound $\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y} \in F]$ completeness property

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y} \in \cup F] \geq 1 - \epsilon_c,$$

and upper bound $\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y} \in \cup F]$ with the soundness property

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y} \in \cup F] \leq \epsilon_s,$$

since $\mathbf{y} \in \cup F$ implies that there are features Morgana can use to convince Arthur of class l whereas $\mathbf{y} \sim \mathcal{D}_{-l}$. Together with Equation (8.7) and Equation (8.8) we arrive at

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, M)] \leq h\left(k\frac{\epsilon_s}{1 - \epsilon_c}\frac{P_{-l}}{P_l}\right) + \epsilon_c = \frac{k\epsilon_s\frac{P_{-l}}{P_l}}{1 - \epsilon_c + k\epsilon_s\frac{P_{-l}}{P_l}} + \epsilon_c.$$

Using $\frac{P_l}{P_{-l}} \leq B$ thus $\frac{P_{-l}}{P_l} \geq B^{-1}$, we can express

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_l}[p(\mathbf{x}, M)] \leq \frac{k\epsilon_s\frac{P_{-l}}{P_l}}{1 - \epsilon_c + k\epsilon_s B^{-1}} + \epsilon_c.$$

Inserted back into equation Equation (8.6) leads us to

$$1 - \mathrm{ap}_{\mathcal{D}}(M) \leq \frac{k\epsilon_s}{1 - \epsilon_c + k\epsilon_s B^{-1}} + \epsilon_c.$$

□

8.6 Context Dependence and Realistic Algorithms

One reasonable critique could be that we only shifted the problem of approximating the real data distribution to solving a hard optimisation task, namely finding a perfectly playing Morgana. This is correct, however, we want to make following two points:

1. The data manifold is something we have no idea about how well we are approximating it if our data set is biased. For Morgana we can at least theoretically search over all possible features sets, even though that is computationally expensive.

2. Morgana only has to find features that have been successfully found by Merlin, albeit in a different class. The question is thus how important this class context (the rest of the data point outside of the feature) could be to find a feature that is not strongly correlated to the class.

Instead of presupposing Morgana to be an optimal algorithm, we consider the same algorithm as for Merlin but with the opposite goal. Given $\mathbf{x} \in D$ and a partial classifier A we define

$$M^{\mathrm{opt}}(\mathbf{x}, l) = \underset{\phi \in \Sigma}{\operatorname{argmax}}\ lA(\phi) \tag{8.9}$$

This means M^{opt} returns a mask that convinces Arthur of class l if at all possible. $M^{\mathrm{opt}}(\mathbf{x}, c(\mathbf{x}))$ and $M^{\mathrm{opt}}(\mathbf{x}, -c(\mathbf{x}))$ represent the best strategy for Merlin and Morgana respectively given a fixed Arthur. We will henceforth consider Merlin and Morgana to be approximation algorithms for M^{opt}. The question is whether for non-optimal (time and space constrained) players an equilibrium will emerge that is close to the one for optimal players.

Our new assumption becomes that finding a small feature ϕ given that it exists should not be more difficult in one class than the other. This assumption might be reasonable for images, but one can easily construct examples where the rest of the image outside of the feature matters, see fig. 8.6.

Let us make this notion more formal. We want to compare the success rates of Merlin and Morgana conditioned on a convincing feature ϕ with $A(\phi) = l$ being present in the data point:

$$\frac{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[M(\mathbf{x}, l) \text{ succeeds} \,|\, \mathbf{x} \in F]}{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[M(\mathbf{x}, l) \text{ succeeds} \,|\, \mathbf{x} \in F]}.$$

The question is how to define the "success". If we measure success by returning ϕ, it might be that Morgana can actually find a smaller feature that convinces Arthur which in that case would not count as a success. If we instead just require that $A(M(\mathbf{x}, l)) = l$ then we have to consider that ϕ might be a feature that is generally hard to find. Merlin could instead rely on a different, easily findable feature that is only present in data points of the correct class, and

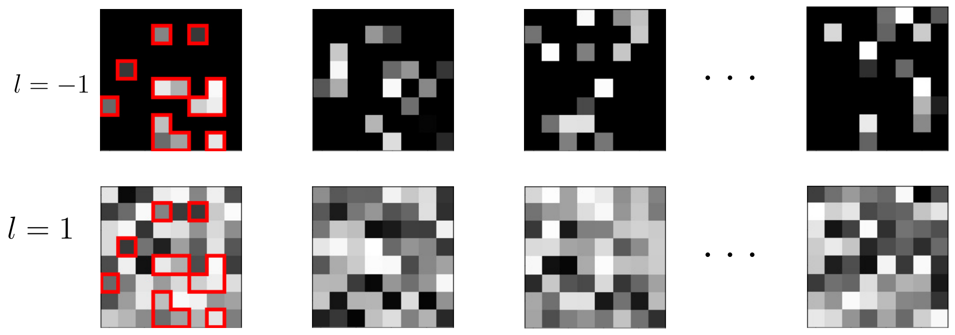


Figure 8.6: Illustration of a data set with strong context dependence. Class -1 consists of k-sparse images whose pixel values sum to some number S. For each of these images there is a non-sparse image in class 1 that shares all non-zero values (marked in red for the first image). Merlin can use the strategy to show all k non-zero pixels for an image from class -1 and $k + 1$ arbitrary non-zero pixels for class 1. Arthur checks if the sum is equal to S or if the number of pixels equal to $k + 1$, otherwise he says "I don't know!". He will then classify with 100% accuracy. Nevertheless, the features Merlin uses for class -1 are completely uncorrelated with the class label. To exploit this, however, Morgana would have to solve the NP-hard [KT06] subset sum problem to find the pixels for images in class 1 that sum to S. The question is not in which class we can find the features, but in which class we can find the features *efficiently*.

Morgana would need to be a again a perfect optimiser to find the feature ϕ. We reason that both approaches would make the fraction unreasonably large even for practical data sets.

The solution is to make the fraction asymmetric and require Merlin to actually find the feature ϕ and Morgana to convince Arthur on the data point. So we consider the expression

$$\frac{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[M(\mathbf{x},l) = \phi \,|\, \mathbf{x} \in F]}{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[A(M(\mathbf{x},l)) = l \,|\, \mathbf{x} \in F]}$$

and try to assume an upper bound. In the following definition we apply this idea to whole sets of features on which Arthur is constant. That means we maximise over sets $F \subseteq A^{-1}(l)$ where A^{-1} is the preimage of A. If Merlin can find features from this set in one class, then Morgana should not have a much harder time finding them in the other class, given that they are present in the data point.

Definition 8.14 (α-context dependence) Let $((D, \sigma, \mathcal{D}), c, \Sigma)$ be a two-class data space. Let $A \in \mathcal{A}$ and $M\colon D \times \{-1, 1\} \to \Sigma^d$ an approximation for M^{opt} as defined in eq. (8.9). Then the context dependence of M with respect to A is defined as

$$\max_{l\in\{-1,1\}} \max_{F\subseteq A^{-1}(l)} \frac{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[M(\mathbf{x},l) \in F \,|\, \mathbf{x} \in \cup F]}{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[A(M(\mathbf{x},l)) = l \,|\, \mathbf{x} \in \cup F]} \leq \alpha.$$

It might make sense for the definition to require that $\mathcal{D}(\cup F)$ to be large, so that no insignificant feature sets blow up the context dependence, but since we cannot evaluate this quantity numerically anyway we leave this unrestricted for now.

Lemma 8.15 Let $((D,\sigma,\mathcal{D}),c,\Sigma)$ be a two-class data space with AFC constant k and class imbalance B. Let $A \in \mathcal{A}$ and $M\colon D\times\{-1,1\} \to \Sigma^d$ be an approximation for M^{opt} as defined in eq. (8.9) with context dependence α. From

1. Completeness:
$$\min_{l\in\{-1,1\}} \mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[A(M(\mathbf{x},l)) = c(\mathbf{x})] \geq 1-\epsilon_c,$$
2. Soundness:
$$\max_{l\in\{-1,1\}} \mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[A(M(\mathbf{x},-l)) = -c(\mathbf{x})] \leq \epsilon_s,$$

follows
$$\mathrm{ap}_{\mathcal{D}}(M) \geq 1-\epsilon_c - \frac{\alpha k\epsilon_s}{1-\epsilon_c+\alpha k\epsilon_s B^{-1}}.$$

Proof. We follows the same proof steps and definitions as in the proof of theorem 8.13 up to eq. (8.8). Then we consider the following equation

$$\alpha^{-1}\frac{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y}\in F]}{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y}\in F]} \leq \frac{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y}\in F]}{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y}\in F]}\frac{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[A(M(\mathbf{x},l)) = l\,|\,\mathbf{x}\in F]}{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[M(\mathbf{x},l)\in F\,|\,\mathbf{x}\in F]} \tag{8.10}$$
$$= \frac{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[A(M(\mathbf{x},l)) = l,\, \mathbf{x}\in F]}{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[M(\mathbf{x},l)\in F,\, \mathbf{x}\in F]} \tag{8.11}$$
$$= \frac{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[A(M(\mathbf{x},l)) = l]}{\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[M(\mathbf{x},l)\in F]}, \tag{8.12}$$

where we used in the last step that $M(\mathbf{x},l)\in F \Rightarrow \mathbf{x}\in F$ and that when $\mathbf{x}\in D_l$ we have $A(M(\mathbf{x},l)) = l \Rightarrow \mathbf{x}\in F$ both by definition of F. We know by the soundness criterion

$$\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_{-l}}[A(M(\mathbf{x},l)) = l] \leq \epsilon_s.$$

From the definition of F and the completeness criterion we get $\mathbb{P}_{\mathbf{x}\sim\mathcal{D}_l}[M(\mathbf{x},l)\in F] \geq 1-\epsilon_c$. Putting everything together we arrive at

$$\frac{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_{-l}}[\mathbf{y}\in F]}{\mathbb{P}_{\mathbf{y}\sim\mathcal{D}_l}[\mathbf{y}\in F]} \leq \frac{\alpha\epsilon_s}{1-\epsilon_c},$$

which allows us to continue the proof analogously to theorem 8.13. □

8.7 Finite and biased Data Sets

After considering realistic algorithms we have to consider realistic, finite data sets on which we evaluate the completeness and soundness criteria. Let us introduce two data distributions $\mathcal{T}$ and $\mathcal{D}$ on the same data set D, where $\mathcal{T}$ is considered the “true” distribution and $\mathcal{D}$ a different potentially biased distribution. We define this bias as the distance of the two distributions with respect to a specific feature $\phi \in \Sigma$ as

$$d^l_\phi(\mathcal{D}, \mathcal{T}) = |\mathbb{P}_{\mathbf{x}\in\mathcal{D}}[c(\mathbf{x}) = l \,|\, \mathbf{x} \in \phi] - \mathbb{P}_{\mathbf{x}\in\mathcal{T}}[c(\mathbf{x}) = l \,|\, \mathbf{x} \in \phi]|.$$

Note that $d^1_\phi(\mathcal{D}, \mathcal{T}) = d^{-1}_\phi(\mathcal{D}, \mathcal{T}) =: d_\phi(\mathcal{D}, \mathcal{T})$ and $0 \leq d_\phi(\mathcal{D}, \mathcal{T}) \leq 1$. This distance measures if a feature ϕ is differently distributed to the two classes for between the two distributions.

Consider as ϕ for example the water in the boat images of the PASCAL VOC data set, see chapter 3. The feature is a strong predictor for the "boat" class in the test data distribution $\mathcal{D}$ but should not be indicative for the real world distribution which includes images of water without boats. We now want to prove that a feature selected by M is either an informative feature or is misrepresented in the test data set. This means that if we encounter an unintuitive feature in the test data set we can pinpoint to where the dataset might be biased.

Lemma 8.16 Let $D, \sigma, \mathcal{D}, c, \Sigma, k, B, A, M$ and α be defined as in lemma 8.15. Let $\mathcal{T}$ be the true data distribution on D. Then for every $\delta \in [0, 1]$ with probability

$$1 - \frac{1}{\delta}\left(\frac{k\alpha\epsilon_s}{1 + k\alpha\epsilon_s B^{-1} - \epsilon_c} + \epsilon_c\right)$$

for $\mathbf{x} \sim \mathcal{D}$ we have

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{T}}[c(\mathbf{y}) = c(\mathbf{x}) \,|\, \mathbf{y} \in M(\mathbf{x})] \geq 1 - \delta - d_{M(\mathbf{x})}(\mathcal{D}, \mathcal{T}).$$

The proof that follows is straightforward from the definition of the feature-wise distance.

Proof. From lemma 8.15 we get that $\mathrm{ap}_\mathcal{D}(M) \geq 1 - \epsilon_c - \frac{k\epsilon_s}{1-\epsilon_c+k\epsilon_s B^{-1}}$ and together with lemma 8.7 we take that with probability $1 - \delta^{-1}(1 - \mathrm{ap}_\mathcal{D}(M))$ we have

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{D}}[c(\mathbf{x}) = c(\mathbf{x}) \,|\, \mathbf{y} \in M(\mathbf{x})] \geq 1 - \delta.$$

with $\mathbf{x} \sim \mathcal{D}$. Then by definition

$$\begin{aligned}\mathbb{P}_{\mathbf{y}\sim\mathcal{T}}[c(\mathbf{x}) = c(\mathbf{x}) \,|\, \mathbf{y} \in M(\mathbf{x})] &\geq \mathbb{P}_{\mathbf{y}\sim\mathcal{D}}[c(\mathbf{x}) = c(\mathbf{x}) \,|\, \mathbf{y} \in M(\mathbf{x})] - d_{M(\mathbf{x})}(\mathcal{D}, \mathcal{T}) \\ &\geq 1 - \delta - d_{M(\mathbf{x})}(\mathcal{D}, \mathcal{T}).\end{aligned}$$

□

The next iteration considers the fact that we only sample a finite amount of data from the possibly biased test data distribution. This will only give us an approximate idea of the soundness and completeness constants.

Lemma 8.17 Let $D, \sigma, \mathcal{D}, cA, M$ and $\mathcal{T}$ be defined as in lemma 8.16. Let $D^{\text{test}} = (\mathbf{x}_i)_{i=1}^N \sim \mathcal{D}^N$ be N random samples from $\mathcal{D}$. Let

$$\epsilon_c^{\text{test}} = \max_{l \in \{-1,1\}} \frac{1}{N} \sum_{\mathbf{x} \in D_l^{\text{test}}} \mathbb{1}(A(M(\mathbf{x}, c(\mathbf{x}))) \neq c(\mathbf{x})),$$

and

$$\epsilon_s^{\text{test}} = \max_{l \in \{-1,1\}} \frac{1}{N} \sum_{\mathbf{x} \in D_l^{\text{test}}} \mathbb{1}(A(M(\mathbf{x}, -c(\mathbf{x}))) = -c(\mathbf{x})).$$

Then it holds with probability $1 - \eta$ where $\eta \in [0, 1]$ that on the true data distribution $\mathcal{T}$ A and M obey completeness and soundness criteria with

$$\begin{aligned} \epsilon_c &\leq \epsilon_c^{\text{test}} + \epsilon_{\text{dist}} + \epsilon_{\text{sample}} \quad \text{and} \\ \epsilon_s &\leq \epsilon_s^{\text{test}} + \epsilon_{\text{dist}} + \epsilon_{\text{sample}} \end{aligned}$$

respectively, where $\epsilon_{\text{dist}} = \max_{l \in \{-1,1\}} \|\mathcal{D}_l - \mathcal{T}_l\|_{TV}$ and $\epsilon_{\text{sample}} = \sqrt{\frac{1}{2N} \log\left(\frac{4}{\eta}\right)}$.

The proof follows trivially from Hoeffding's inequality and the definition of the total variation norm.

Proof. We define $E_{c,l} = \{\mathbf{x} \in D_l \,|\, A(M(\mathbf{x}, c(\mathbf{x}))) \neq c(\mathbf{x})\}$ for $l \in \{-1, 1\}$ and let $E_{c,l}^{\mathcal{D}}$ be the Bernoulli random variable for the event that $X \in E_{c,l}$ where $X \sim \mathcal{D}$. Then

$$\mathbb{P}_{\mathbf{x} \sim \mathcal{D}_l}[A(M(\mathbf{x}, c(\mathbf{x}))) \neq c(\mathbf{x})] = \mathbb{E}\left[E_{c,l}^{\mathcal{D}}\right]$$

Using Hoeffding's inequality we can bound for any $t > 0$

$$\mathbb{P}\left[\left|\left(\frac{1}{N} \sum_{\mathbf{x} \in D_l^{\text{test}}} \mathbb{1}(\mathbf{x} \in E_{c,l})\right) - \mathbb{E}\left[E_{c,l}^{\mathcal{D}}\right]\right| > t\right] \leq e^{-2nt^2}.$$

We choose t such that $e^{-2t^2} = \frac{\eta}{4}$. We use a union bound for the maximum over $l \in \{-1, 1\}$ which results in a probability of $2\frac{\eta}{4} = \frac{\eta}{2}$ we have

$$\max_{l \in \{-1,1\}} \mathbb{E}\left[E_{c,l}^{\mathcal{D}}\right] > \epsilon_c^{\text{test}} + \sqrt{\frac{1}{2N} \log\left(\frac{4}{\eta}\right)},$$

and thus with $1 - \frac{\eta}{2}$ we have $\max_{l \in \{-1,1\}} \mathbb{E}\left[E_{c,l}^{\mathcal{D}}\right] \leq \epsilon_c^{\text{test}} + \epsilon^{\text{sample}}$. Using the definition of the total variation norm

$$\|\mathcal{T} - \mathcal{D}\|_{TV} = \sup_{J \subset D} |\mathcal{T}(J) - \mathcal{D}(J)|,$$

with $J = E_{c,l}$ we can derive $\mathbb{E}\left[E_{c,l}^{\mathcal{T}}\right] \leq \mathbb{E}\left[E_{c,l}^{\mathcal{T}}\right] + \|\mathcal{T} - \mathcal{D}\|_{TV}$ and thus

$$\epsilon_c \leq \epsilon_c^{\text{test}} + \epsilon^{\text{sample}} + \epsilon^{\text{dist}}.$$

We can treat ϵ_s analogously and take a union bound over both the completeness and soundness bounds holding which results in the probability of $1-\eta$. □

Using lemma 8.17 allows us to bound the soundness and completeness constants that A and M obey on the true data distribution $\mathcal{T}$ and thus to get an idea about the quality of the features selected by M.

8.8 Implementation of Arthur-Merlin Regularisation

In the last sections we introduced how we can derive guarantees on the quality of a feature selector from empirical quantities on test data sets. The question that remains is how to actually get a classifier (Arthur) and feature selector (Merlin) to ensure small completeness and soundness bounds.

To stay on topic, we want to model the classifier Arthur as a neural network. For Merlin and Morgana we propose two different approaches: first an optimisation procedure and secondly a UNet or autoencoder network. Both approaches lead to different training setups which we want to compactly explain in this section. We now explicitly model our feature space as in example 8.2, considering parts of images up to size $k \in \mathbb{N}$ as features.

Generally, Arthur now becomes a continuous function which returns not a single label $l \in \{-1, 0, 1\}$ as before, but a normalised probability vector for all labels, so $A : [0,1]^d \to [0,1]^3$ with $\sum_{l\in\{-1,0,1\}} A_l(\mathbf{x}_S) = 1$. Regarding the loss function we consider the correct label for a feature selected by Merlin the label of the data point it was selected from. For a feature selected by Morgana both the label of the data point and the "I don't know" options become valid labels. The loss for Arthur in terms of the cross entropy for example, would be

$$L_A = -\mathbb{E}_{\mathbf{x}\sim D}\Big[\log(A(M(\mathbf{x}))_{c(\mathbf{x})}) + \lambda \log(1 - A(\widehat{M}(\mathbf{x}))_{-c(\mathbf{x})})\Big],$$

where $\lambda \in \mathbb{R}_{\geq 0}$. The parameter λ can be used to tune the trade-off between soundness and completeness. To derive the loss for Merlin and Morgana we have to be more specific in our realisation.

Realisation with Optimisation Procedure For this approach Arthur will be pretrained on the data set, considering all data points as if generated by Merlin with the whole data point as feature. We then select a number $k < d$ and define Merlin and Morgana as

$$M(\mathbf{x}) = \mathbf{x}_{S^*} \text{ where } S^* = \underset{|S|\leq k}{\operatorname{argmax}} \log(A(\mathbf{x}_S)_{c(\mathbf{x})})$$

and

$$\widehat{M}(\mathbf{x}) = \mathbf{x}_{S^*} \text{ where } S^* = \underset{|S|\leq k}{\operatorname{argmax}} \log(A(\mathbf{x}_S)_{-c(\mathbf{x})}).$$

Since A is differentiable, we propose again a convex relaxation as in chapter 7 in the form of

$$\mathbf{s}^*(\mathbf{x}, l) = \underset{\mathbf{s}\in[0,1]^d \; \|\mathbf{s}\|\leq k}{\operatorname{argmax}} \log(A(\mathbf{s}\cdot\mathbf{x})_l)$$

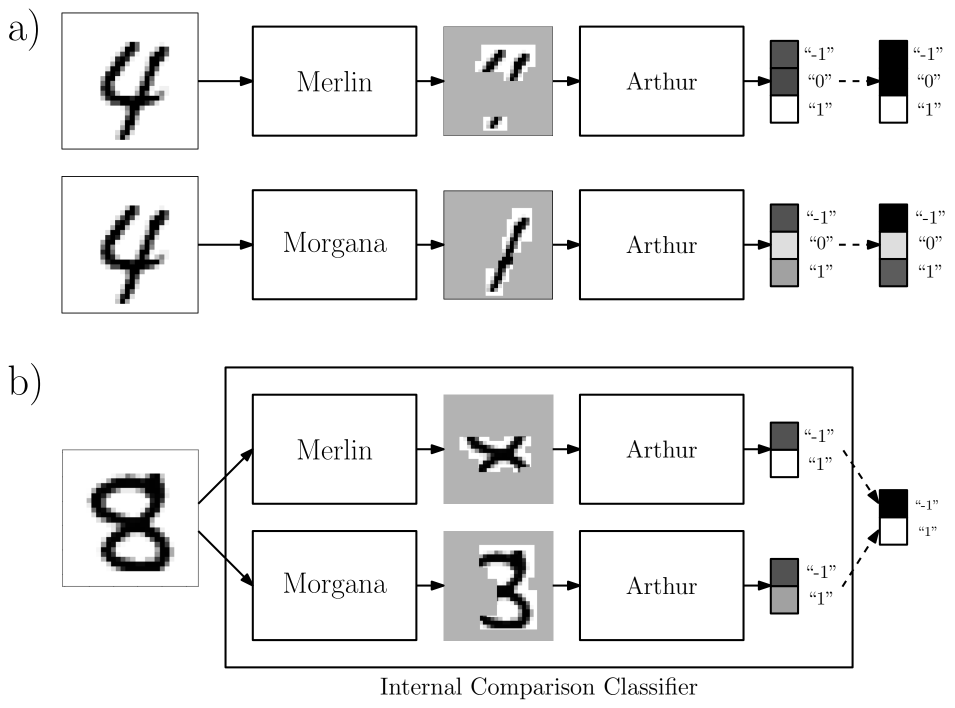


Figure 8.7: Two possible training setups where Arthur, Merlin and Morgana are all realised as networks. The dashed lines in both correspond to softmax operations. a) Examlple of classifying "4" against "1". Arthur puts out class probabilities for masked images of Merlin and Morgana separately. b) Examplel classifying "8" against "3". Arthur generates logit values for both masks separately but aggregates them to give class probability only for the class labels, without having the option of saying "I don't know.".

and solve this problem by projected gradient descent or Frank-Wolfe. We set the k largest components of $\mathbf{s}$ to 1 and the rest to 0, let us call this vector $\mathbf{s}(\mathbf{x}, l)^k$. Then

$$M(\mathbf{x}) = \mathbf{s}(\mathbf{x}, c(\mathbf{x}))^k \cdot \mathbf{x} \quad \text{and} \quad \widehat{M}(\mathbf{x}) = \mathbf{s}(\mathbf{x}, -c(\mathbf{x}))^k \cdot \mathbf{x}.$$

Running these optimisations over the data set D now generates a new data set of masked data that can be used to retrain Arthur. This process can be iterated until a satisfying sparsity of Merlin has been reached or until the prediction accuracy of Arthur cannot be recovered with the smaller features. The whole setup can be summarised as follows.

1. Arthur trains on the original data set D.
2. Merlin and Morgana generate their respective masks and the resulting features are saved in D_M, resp. $D_{\widehat{M}}$ for some $k \leq d$.
3. Arthur trains on D_M and $D_{\widehat{M}}$.
4. Repeat from step 2 with $k \leftarrow k' < k$ as long as Arthur can reliably predict the label from Merlin's features.

To speed up the process, masks from earlier iterations can be used as initialisation for the next round.

Realisation with Masking network Another possible approach is making Merlin and Morgana networks themselves. These network would for every data point output a mask to mask the data point with, which in turn get's classified by Arthur. If we allow for continuous masks, so $M : D \to [0,1]^d$, the function $A(M(\mathbf{x}) \cdot \mathbf{x})$ becomes differentiable and we can directly optimise A, M and $\widehat{M}$ in parallel. See fig. 8.7 a) as an illustration. The loss functions for Merlin and Morgana then become

$$L_M = -\mathbb{E}_{\mathbf{x}\sim D}\Big[\log(A(M(\mathbf{x}))_{c(\mathbf{x})}) + \lambda_1 \mathrm{Reg}(M(\mathbf{x}))\Big],$$
$$L_{\widehat{M}} = -\mathbb{E}_{\mathbf{x}\sim D}\Big[\log(A(M(\mathbf{x}))_{-c(\mathbf{x})}) + \lambda_2 \mathrm{Reg}(\widehat{M}(\mathbf{x}))\Big],$$

where $\lambda_1, \lambda_2 \in \mathbb{R}_{\geq 0}$ and where Reg is a regularisation term that enforces sparseness and potentially contains an entropy term for every mask element driving them toward 0 or 1, e.g.

$$\mathrm{Reg}(\mathbf{s}) = \|\mathbf{s}\|_1 + \gamma \sum_{i\in[d]} H_b(s_i).$$

The parameters λ_1, λ_2 and $\gamma \in \mathbb{R}_{\geq 0}$ have to be fine-tuned during training.

An alternative approach is to let Arthur aggregate the logits he derived from Merlin's and Morgana's mask to give a final decision over the class, see fig. 8.7 b). In this case $A : \Sigma^d \to \mathbb{R}^2$. The loss function for Arthur can then be defined as

$$L_A = -\mathbb{E}_{\mathbf{x}\sim D}\Big[\log\Big(\mathrm{softmax}(A(M(\mathbf{x})) + A(\widehat{M}(\mathbf{x}))_{c(\mathbf{x})}\Big)\Big],$$

which is the cross-entropy loss for the softmax of the added logits. The advantage of that approach is that one does not need a third "I don't know" output. All three networks can then be seen as a single classifier with an internal comparison of the two masked data points.

Compared with the optimisation approach the network approach has the advantage of being computationally more efficient, at least in the deployment phase. Explanations for classifications can directly be generated by the network and don't have to be optimised by a potentially lengthy optimisation procedure.

The downside is that we can expect from our experiences in training other adversarial setups that training might be unstable and success heavily dependent on the parameters. The optimisation based approach is basically just a method for data set augmentation and should thus be expected to converge in a comparably stable manner.

8.9 Discussion

As we have seen, without sufficient knowledge of the true underlying data distribution there cannot be reliable definitions of relevance for a classifier function. The authors of [Rud19] raise the important point that black box models are often not necessary, and models with an *intrinsic* explanation should be preferred. One needs strong performance advantages to justify using black box models that have to be explained 'post hoc'. However, a lot of the most impressive breakthroughs (competitive games, protein folding, language models) were possible by using black box models like neural networks.

In this chapter we have formulated a novel approach for classification setup where meaningful features have to be exchanged between different black-box agents. This is interesting because it constitutes neither a post-hoc explanation of an existing classifier but neither does it reduce the expressibility of the considered function classes.

Thus Arthur-Merlin regularisation serves three functions:

1. They augment the data set since Arthur has effectively more examples to train on.
2. They push Arthur to use the right features.
3. They allow us evaluate the quality of the features selected by Merlin.

Crucially, our results hinge on the assumption that realistically trained agents (e.g. neural networks trained by gradient descent) operate at a similar equilibrium as optimal agents. This of course is difficult to prove and we leave it open to further research how to approach this question. Promising in this regard is the sound performance of generative adversarial networks [KLA19] that operate at a similar equilibrium, though admittedly with only two actors instead of three.

What we need now is a thorough numerical investigation on real-world datasets. In our upcoming research we plan to show that we can reach strong completeness and soundness bounds and that we can do so with stable convergence.

9 Conclusion

This chapter concludes our investigation. We take this opportunity to review what we have learned and to discuss the open questions that still remain, as well as how we plan to direct our upcoming research.

9.1 Summary

We started our investigation with a series of case studies that demonstrate the usefulness of relevance maps: They can be applied to detect artefacts in data sets as well as quantitatively track the focus of neural network agents playing simple computer games. This prompted our quest to find formal definitions of relevance maps that allow us to prove guarantees for the "usefulness" of these maps. Let us revisit the central questions of this endeavour and discuss how far we have come in answering them.

1. How can we formally define relevance maps?

2. How hard is it to calculate maps with theoretical guarantees?

3. How can we determine them practically?

Let us start with the first question. So far, except for the Shapley values, most relevance methods rely on heuristic motivations. We aimed to complement these heuristics by defining relevance maps in a formal way using the preexisting concept of prime implicants. To make them suitable for high-dimensional non-linear functions, we generalised prime implicant explanations in two aspects: First, from Boolean to continuous classifier functions on a continuous domain, and secondly from a deterministic to a probabilistic setting.

This resulted in our rate-distortion formalism with a trade-off between the size of the implicant and the deviation of the classifier output. We were then able to give meaning to relevance maps by interpreting them as greedy strategies to minimise the rate-distortion functional. The idea is simple: For a fixed rate k select the k most relevant input parameters as implicant to minimise the distortion. Different maps can then be compared by how well they minimise the rate-distortion functional for varying rates.

Regarding the second question, we showed in our theoretical analysis that evaluating the optimal rate-distortion function generally constitutes a hard problem, even if we only want to approximate it with any non-trivial approximation factor. This is due to the highly flexible nature of neural networks that allowed us to encode arbitrary logical formulas. This same flexibility also facilitated our impossibility result for propagating distributions through neural networks.

Thus for efficient methods we will not be able to prove that they produce any non-trivial results. This unfortunate fact holds as long as we consider neural networks general enough to represent arbitrary Boolean circuits. This lies in contrast to the fact that relevance methods indeed provide useful information in a lot of practical scenarios. A way forward would be to introduce subtle restrictions on the network and the input space that reflect the real-world data structures they model. As of yet, however, these are not well enough understood. As long as this is the case, we have to rely on heuristic solutions and evaluate them numerically in realistic scenarios.

Let us finally address the third question. To facilitate our numerical investigation we relaxed the problem into an optimisation of a non-convex functional over a convex set—fit for projected gradient descend. The resulting maps were either the best or among the best relevance maps for the challenges we considered. However, we noticed that even in a probabilistic setting the prime implicants may not fully be fitting for high-dimensional data that is distributed in an intricate manner: Without faithfully representing the true data distribution most explanation methods will have scenarios in which they fail, including ours. To remedy this fact, we presented in our last chapter a new framework, called Arthur-Merlin-Regularisation, that provides guarantees for the selected features without explicitly modelling the data distribution. We posit this as a way forward to find prime implicants that provably have high information about the class of the input.

9.2 Outlook

There still remains work to be done until relevance methods can be reliably used to interpret the decisions of neural network classifiers. Intuitively there is always a trade-off between how expressive your models are and how easy it is to understand them. The field of XAI still has to prove that they are more than an afterthought in the design of the most performant models. In how far expressiveness and explainability can be successfully combined remains an open question.

Certain relevance methods such as RDE and SHAP will give reasonable results when the underlying data manifold is modelled correctly. There exists a number of approaches such as variational autoencoders, normalisation flow networks and generative adversarial networks that have achieved practical success in modelling challenging data distributions. What we need now are practical guarantees that measure how well the data distribution gets modelled, which we can then translate into guarantees for our explanation methods.

For interacting black-box models, such as our Arthur-Merlin approach, we presented theoretical evidence that they lead to trustworthy explanations without having to model the data distribution. Of course, our analysis hinges on the idea that the equilibrium found by efficiently realisable agents is close the the one at which optimal agents operate—an equally difficult to confirm assumption. Additionally, we need to follow up these results with numerical experiments that confirm that these systems can be trained in an efficient and stable manner. We hope to continue our research in this direction.

References

[AC16] Dario Amodei and Jack Clark. *Faulty Reward Functions in the Wild.* 2016.

[AJL19] Kjersti Aas, Martin Jullum, and Anders Løland. “Explaining individual predictions when features are dependent: More accurate approximations to Shapley values”. In: *arXiv preprint arXiv:1903.10464* (2019).

[Alb+18a] Maximilian Alber et al. “iNNvestigate neural networks!” In: *Preprint at https://arxiv.org/abs/1808.04260* (2018).

[Alb+18b] Maximilian Alber et al. “iNNvestigate neural networks!” In: *CoRR* abs/1808.04260 (2018).

[All+12] Jeffrey D. Allen et al. “Comparing Statistical Methods for Constructing Large Scale Gene Networks”. In: *PLoS ONE* 7.1 (Jan. 2012), pp. 1–9.

[AN20] Chirag Agarwal and Anh Nguyen. “Explaining image classifiers by removing input features using generative models”. In: *Proceedings of the Asian Conference on Computer Vision.* 2020.

[And+20] Christopher Anders et al. “Fairwashing explanations with off-manifold detergent”. In: *International Conference on Machine Learning.* PMLR. 2020, pp. 314–323.

[Arr+17a] Leila Arras et al. ““What is Relevant in a Text Document?”: An Interpretable Machine Learning Approach”. In: *PLoS ONE* 12.8 (2017), e0181142.

[Arr+17b] Leila Arras et al. “Explaining Recurrent Neural Network Predictions in Sentiment Analysis”. In: *Proc. EMNLP’17 Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis (WASSA).* 2017, pp. 159–168.

[Bac+15a] Sebastian Bach et al. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PLoS ONE* 10.7 (2015), e0130140.

[Bac+15b] Sebastian Bach et al. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PLOS ONE* 10.7 (July 2015), pp. 1–46.

[Bac+16] Sebastian Bach et al. “Controlling Explanatory Heatmap Resolution and Semantics via Decomposition Depth”. In: *Proc. IEEE International Conference on Image Processing.* 2016, pp. 2271–2275.

[Bae+10] David Baehrens et al. “How to Explain Individual Classification Decisions”. In: *J. Mach. Learn. Res.* 11 (2010), pp. 1803–1831.

[Bar10] Deirdre Barrett. *Supernormal stimuli: How primal urges overran their evolutionary purpose.* WW Norton & Company, 2010.

[Bel+13] M. G. Bellemare et al. “The Arcade Learning Environment: An Evaluation Platform for General Agents”. In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 253–279.

[Bel52] Richard Bellman. “On the Theory of Dynamic Programming”. In: *Proceedings of the National Academy of Sciences* 38.8 (1952), pp. 716–719.

[Ber03] Toby Berger. “Rate-Distortion Theory”. In: *Wiley Encyclopedia of Telecommunications.* Ed. by J. G. Proakis. American Cancer Society, 2003.

[BHN99] Richard H. Byrd, Mary E. Hribar, and Jorge. Nocedal. “An Interior Point Algorithm for Large-Scale Nonlinear Programming”. In: *SIAM Journal on Optimization* 9.4 (1999), pp. 877–900.

[Bin+16] Alexander Binder et al. “Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers”. In: *Proc. International Conference on Artificial Neural Networks.* Lecture Notes in Computer Science 9887 (2016), pp. 63–71.

[BK98] Xavier Boyen and Daphne Koller. “Tractable Inference for Complex Stochastic Processes”. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence.* UAI’98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., 1998, pp. 33–42.

[BM19] Yochai Blau and Tomer Michaeli. “Rethinking lossy compression: The rate-distortion-perception tradeoff”. In: *International Conference on Machine Learning.* PMLR. 2019, pp. 675–685.

[Bos12] Nick Bostrom. “The superintelligent will: Motivation and instrumental rationality in advanced artificial agents”. In: *Minds and Machines* 22.2 (2012), pp. 71–85.

[Bro+17] Tom B. Brown et al. “Adversarial Patch”. In: *CoRR* abs/1712.09665 (2017).

[Bro+20] Tom B Brown et al. “Language models are few-shot learners”. In: *arXiv preprint arXiv:2005.14165* (2020).

[Bry86] Randal E Bryant. “Graph-based algorithms for boolean function manipulation”. In: *Computers, IEEE Transactions on* 100.8 (1986), pp. 677–691.

[Byr+95] Richard H. Byrd et al. “A Limited Memory Algorithm for Bound Constrained Optimization”. In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208.

[Cam+20] Nick Cammarata et al. “Thread: circuits”. In: *Distill* 5.3 (2020), e24.

[Cha+11] Ken Chatfield et al. “The devil is in the details: an evaluation of recent feature encoding methods”. In: *Proc. British Machine Vision Conference.* 2011, p. 8.

[Cha+18] Chun-Hao Chang et al. “Explaining image classifiers by counterfactual generation”. In: *arXiv preprint arXiv:1807.08024* (2018).

[CJ08] Cassio P. de Campos and Qiang Ji. "Strategy Selection in Influence Diagrams Using Imprecise Probabilities". In: *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*. UAI'08. Helsinki, Finland: AUAI Press, 2008, pp. 121–128.

[CM92] Olivier Coudert and Jean Christophe Madre. "Implicit and Incremental Computation of Primes and Essential Primes of Boolean Functions." In: *DAC*. Vol. 92. 1992, pp. 36–39.

[CNL11] Adam Coates, Andrew Ng, and Honglak Lee. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 215–223.

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[Dar00] Adnan Darwiche. "On the tractable counting of theory models and its application to belief revision and truth maintenance". In: *CoRR* cs.AI/0003044 (2000).

[Dar11] Adnan Darwiche. "SDD: A new canonical representation of propositional knowledge bases". In: *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.

[Dev08] Karthik Devarajan. "Nonnegative Matrix Factorization: An Analytical and Interpretive Tool in Computational Biology". In: *PLoS Comput. Biol.* 4.7 (2008).

[DP94] Xiaotie Deng and Christos H Papadimitriou. "On the complexity of cooperative solution concepts". In: *Mathematics of Operations Research* 19.2 (1994), pp. 257–266.

[Duc+08] John Duchi et al. "Efficient projections onto the ℓ_1-ball for learning in high dimensions". In: *Proceedings of the 25 th International Conference on Machine Learning*. 2008.

[Dug66] James Dugundji. *Topology*. Ed. by I. Kaplansky. Allyn and Bacon, Inc, 1966.

[Edg08] Gerald Edgar. *Measure, Topology, and Fractal Geometry*. Ed. by S. Axler and K. A. Ribet. 2nd. Spinger-Verlag New York, 2008.

[EG95] Thomas Eiter and Georg Gottlob. "The complexity of logic-based abduction". In: *Journal of the ACM (JACM)* 42.1 (1995), pp. 3–42.

[Eve+10] Mark Everingham et al. "The Pascal visual object classes (VOC) challenge". In: *Int. J. Comput. Vision* 88.2 (2010), pp. 303–338.

[FA14] Hatem Fayed and Amir Atiya. "An evaluation of the integral of the product of the error function and the normal probability density with application to the bivariate normal integral". In: *Mathematics of Computation* 83.285 (2014), pp. 235–250.

[Fis36] Ronald A Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of eugenics* 7.2 (1936), pp. 179–188.

[Fry+20] Christopher Frye et al. “Shapley explainability on the data manifold”. In: *arXiv preprint arXiv:2006.01272* (2020).

[Fuk80] Kunihiko Fukushima. “A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biol. Cybern.* 36 (1980), pp. 193–202.

[FV17] Ruth C Fong and Andrea Vedaldi. “Interpretable explanations of black boxes by meaningful perturbation”. In: *Proceedings of the IEEE International Conference on Computer Vision.* 2017, pp. 3429–3437.

[FWJ08] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. “A linear approximation method for the Shapley value”. In: *Artificial Intelligence* 172.14 (2008), pp. 1673–1699.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.* Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 315–323.

[GBC16a] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. “Back-propagation and other differentiation algorithms”. In: *Deep Learning* (2016), pp. 200–220.

[GBC16b] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT Press, 2016.

[GE03] Isabelle Guyon and André Elisseeff. “An Introduction to Variable and Feature Selection”. In: *J. Mach. Learn. Res.* 3 (2003), pp. 1157–1182.

[GL10] Karol Gregor and Yann LeCun. “Learning Fast Approximations of Sparse Coding”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10).* Haifa, Israel: Omnipress, June 2010, pp. 399–406.

[GLS88] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization.* English. Vol. 2. Algorithms and Combinatorics. Springer, 1988.

[GR18] Jochen Gast and Stefan Roth. “Lightweight Probabilistic Deep Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.* June 2018, pp. 3369–3378.

[Gra+18] Katja Grace et al. “When will AI exceed human performance? Evidence from AI experts”. In: *Journal of Artificial Intelligence Research* 62 (2018), pp. 729–754.

[Gre+18] Samuel Greydanus et al. “Visualizing and Understanding Atari Agents”. In: *Proc. International Conference on Machine Learning.* 2018, pp. 1787–1796.

[Gup+21] Abhishek Gupta et al. “Deep Learning for Object Detection and Scene Perception in Self-Driving Cars: Survey, Challenges, and Open Issues”. In: *Array* (2021), p. 100057.

[Han+98] Robin Hanson et al. *A Critical Discussion of Vinge’s Singularity Concept.* 1998.

[Hau+14] Stefan Haufe et al. “On the interpretation of weight vectors of linear models in multivariate neuroimaging”. In: *NeuroImage* 87 (2014), pp. 96–110.

[He+16] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 770–778.

[Hoe94] Wassily Hoeffding. “Probability inequalities for sums of bounded random variables”. In: *The Collected Works of Wassily Hoeffding.* Springer, 1994, pp. 409–426.

[Hor+19] Fabian Horst et al. “Explaining the Unique Nature of Individual Gait Patterns with Deep Learning”. In: *Sci. Rep.* 9 (2019), p. 2391.

[Hoy02] P.O. Hoyer. “Non-negative sparse coding”. In: *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing.* 2002, pp. 557–565.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural Comput.* 9.8 (1997), pp. 1735–1780.

[IIM20] Yacine Izza, Alexey Ignatiev, and Joao Marques-Silva. “On explaining decision trees”. In: *arXiv preprint arXiv:2010.11034* (2020).

[INM19] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. “Abduction-based explanations for machine learning models”. In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 33. 2019, pp. 1511–1519.

[IS15] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning.* PMLR. 2015, pp. 448–456.

[Izz+21] Yacine Izza et al. “Efficient Explanations With Relevant Sets”. In: *arXiv preprint arXiv:2106.00546* (2021).

[Jia+14] Yangqing Jia et al. “Caffe: Convolutional architecture for fast feature embedding”. In: *Proc. ACM International Conference on Multimedia.* 2014, pp. 675–678.

[JNV14] Pasi Jylänki, Aapo Nummenmaa, and Aki Vehtari. “Expectation propagation for neural networks with sparsity-promoting priors”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1849–1901.

[Jum+21] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (2021), pp. 583–589.

[Kho+19] Pasha Khosravi et al. “What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19.* International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 2716–2724.

[Kin+19] Pieter-Jan Kindermans et al. “The (un) reliability of saliency methods”. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning.* Springer, 2019, pp. 267–280.

[KLA19] 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.* 2019, pp. 4401–4410.

[KLM96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. “Reinforcement learning: A survey”. In: *Journal of Artif. Intel. Res.* 4 (1996), pp. 237–285.

[Kob+17] Erich Kobler et al. “Variational Networks: Connecting Variational Methods and Deep Learning”. In: *Pattern Recognition.* Ed. by Volker Roth and Thomas Vetter. Cham: Springer International Publishing, 2017, pp. 281–293.

[Kon+10] Igor Kononenko et al. “An efficient explanation of individual classifications using game theory”. In: *Journal of Machine Learning Research* 11.Jan (2010), pp. 1–18.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Proc. Advances in Neural Information Processing Systems.* 2012, pp. 1097–1105.

[KT06] Jon Kleinberg and Eva Tardos. *Algorithm design.* Pearson Education India, 2006.

[Lap+16a] S. Lapuschkin et al. “Analyzing classifiers: Fisher vectors and deep neural networks”. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 2912–2920.

[Lap+16b] Sebastian Lapuschkin et al. “Analyzing classifiers: Fisher vectors and deep neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 2912–2920.

[Lap+16c] Sebastian Lapuschkin et al. “The Layer-wise Relevance Propagation Toolbox for Artificial Neural Networks”. In: *J. Mach. Learn. Res.* 17.114 (2016), pp. 1–5.

[Lap+19] Sebastian Lapuschkin et al. “Unmasking Clever Hans predictors and assessing what machines really learn”. In: *Nature communications* 10.1 (2019), p. 1096.

[LB+95] Yann LeCun, Yoshua Bengio, et al. “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[LeC+98] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324.

[Lee+17] June-Goo Lee et al. “Deep learning in medical imaging: general overview”. In: *Korean journal of radiology* 18.4 (2017), pp. 570–584.

[Lei+17] Jan Leike et al. “AI Safety Gridworlds”. In: *Preprint at https://arxiv.org/abs/1711.09883* (2017).

[LGM98] Michael L Littman, Judy Goldsmith, and Martin Mundhenk. “The computational complexity of probabilistic planning”. In: *Journal of Artificial Intelligence Research* 9 (1998), pp. 1–36.

[LH18] Boyu Lyu and Anamul Haque. “Deep learning based tumor type classification using gene expression data”. In: *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics.* ACM. 2018, pp. 89–96.

[Liu+18] Xin Liu et al. “DPatch: Attacking Object Detectors with Adversarial Patches”. In: *CoRR* abs/1806.02299 (2018).

[Liu+19] Shusen Liu et al. “Generative counterfactual introspection for explainable deep learning”. In: *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2019, pp. 1–5.

[LL17a] Scott M Lundberg and Su-In Lee. “A unified approach to interpreting model predictions”. In: *Proceedings of the 31st international conference on neural information processing systems*. 2017, pp. 4768–4777.

[LL17b] Scott M. Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774.

[LMP01] Michael L Littman, Stephen M Majercik, and Toniann Pitassi. “Stochastic boolean satisfiability”. In: *Journal of Automated Reasoning* 27.3 (2001), pp. 251–296.

[Low99] David G Lowe. “Object recognition from local scale-invariant features”. In: *Proc. IEEE International Conference on Computer Vision*. 1999, pp. 1150–1157.

[Mac+19a] Jan Macdonald et al. “A Rate-Distortion Framework for Explaining Neural Network Decisions”. In: *CoRR* abs/1905.11092 (2019).

[Mac+19b] Jan Macdonald et al. “A rate-distortion framework for explaining neural network decisions”. In: *arXiv preprint arXiv:1905.11092* (2019).

[Mac+20] Jan Macdonald et al. “Explaining neural network decisions is hard”. In: *XXAI Workshop, 37th ICML*. 2020.

[Mar00] Pierre Marquis. “Consequence Finding Algorithms”. In: *Handbook of Defeasible Reasoning and Uncertainty Management Systems: Algorithms for Uncertainty and Defeasible Reasoning*. Ed. by Jürg Kohlas and Serafín Moral. Dordrecht: Springer Netherlands, 2000, pp. 41–145.

[Mar91] Pierre Marquis. “Extending abduction from propositional to first-order logic”. In: *International Workshop on Fundamentals of Artificial Intelligence Research*. Springer. 1991, pp. 141–155.

[MB17] Anirbit Mukherjee and Amitabh Basu. “Lower bounds over Boolean inputs for deep neural networks with ReLU gates”. In: *CoRR* abs/1711.03073 (2017).

[MBP21] Jan Macdonald, Mathieu Besançon, and Sebastian Pokutta. “Interpretable Neural Networks with Frank-Wolfe: Sparse Relevance Maps and Relevance Orderings”. In: *arXiv preprint arXiv:2110.08105* (2021).

[Mer+20] Silvan Mertes et al. “This is not the texture you are looking for! Introducing novel counterfactual explanations for non-experts using generative adversarial learning”. In: *arXiv preprint arXiv:2012.11905* (2020).

[Min01] Thomas P. Minka. “A Family of Algorithms for Approximate Bayesian Inference”. AAI0803033. PhD thesis. Cambridge, MA, USA: Massachusetts Institute of Technology, 2001.

[Mni+13] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *Preprint at https://arxiv.org/abs/1312.5602* (2013).

[Mni+15] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), pp. 529–533.

[MOM12] Grégoire Montavon, Geneviève Orr, and Klaus-Robert Müller. *Neural networks: tricks of the trade.* Vol. 7700. springer, 2012.

[Mon+17] Grégoire Montavon et al. “Explaining nonlinear classification decisions with deep Taylor decomposition”. In: *Pattern Recognit.* 65 (2017), pp. 211–222.

[Mon+19] Grégoire Montavon et al. “Layer-wise relevance propagation: an overview”. In: *Explainable AI: interpreting, explaining and visualizing deep learning* (2019), pp. 193–209.

[Mor+17] Matej Moravčík et al. “DeepStack: Expert-level artificial intelligence in heads-up no-limit poker”. In: *Science* 356.6337 (2017), pp. 508–513.

[MSH07] Shuangge Ma, Xiao Song, and Jian Huang. “Supervised group Lasso with applications to microarray data analysis”. In: *BMC Bioinform.* 8 (2007).

[MSM18] Gregoire Montavon, Wojciech Samek, and Klaus-Robert Müller. “Methods for interpreting and understanding deep neural networks”. In: *Digital Signal Processing* 73 (2018), pp. 1–15.

[MW21] Jan Macdonald and Stephan Wäldchen. *A Complete Characterisation of ReLU-Invariant Distributions.* 2021.

[Nar+19] Nina Narodytska et al. “Assessing heuristic machine learning explanations with model counting”. In: *International Conference on Theory and Applications of Satisfiability Testing.* Springer. 2019, pp. 267–278.

[Ngu+16] Anh Nguyen et al. “Synthesizing the preferred inputs for neurons in neural networks via deep generator networks”. In: *Proc. Advances in Neural Information Processing Systems.* 2016, pp. 3387–3395.

[NH10] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning.* ICML’10. Haifa, Israel: Omnipress, 2010, pp. 807–814.

[Nie15] Michael A Nielsen. *Neural networks and deep learning.* Vol. 25. Determination press San Francisco, CA, 2015.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* second. New York, NY, USA: Springer, 2006.

[NYC16] 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: *Preprint at https://arxiv.org/abs/1602.03616* (2016).

[Ord20] Toby Ord. *The precipice: existential risk and the future of humanity.* Hachette Books, 2020.

[OWT19] Jose Oramas, Kaili Wang, and Tinne Tuytelaars. “Visual Explanation by Interpretation: Improving Visual Feedback Capabilities of Deep Neural Networks”. In: *arXiv e-prints* abs/1712.06302 (2019).

[Par02] James D Park. “MAP complexity results and approximation methods”. In: *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc. 2002, pp. 388–396.

[Par96] Ian Parberry. *Circuit complexity and feedforward neural networks.* Hillsdale, NJ: Lawrence Erlbaum, 1996.

[Pos20] Luca M Possati. “Algorithmic unconscious: Why psychoanalysis helps in understanding AI”. In: *Palgrave Communications* 6.1 (2020), pp. 1–13.

[Pot07] Steve M Potter. “What can AI get from neuroscience?” In: *50 years of artificial intelligence.* Springer, 2007, pp. 174–185.

[Pro56] Yu. V. Prokhorov. “Convergence of Random Processes and Limit Theorems in Probability Theory”. In: *Theory of Probability & Its Applications* 1.2 (1956), pp. 157–214.

[PRV18] Philipp Petersen, Mones Raslan, and Felix Voigtlaender. “Topological properties of the set of functions generated by neural networks of fixed size”. In: *arXiv e-prints*, arXiv:1806.08459 (June 2018), arXiv:1806.08459.

[PSM10] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. “Improving the Fisher kernel for large-scale image classification”. In: *Proc. European Conference on Computer Vision.* 2010, pp. 143–156.

[PV18] Philipp Petersen and Felix Voigtlaender. “Optimal approximation of piecewise smooth functions using deep ReLU neural networks”. In: *Neural Networks* 108 (2018), pp. 296–330.

[RSG16a] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.* 2016, pp. 1135–1144.

[RSG16b] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ““Why Should I Trust You?”: Explaining the Predictions of Any Classifier”. In: *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 2016, pp. 1135–1144.

[RSG18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Anchors: High-precision model-agnostic explanations”. In: *Thirty-Second AAAI Conference on Artificial Intelligence.* 2018.

[Rud19] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[Rus+15] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *Int. J. Comput. Vision* 115.3 (2015), pp. 211–252.

[RZL17] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. *Searching for Activation Functions.* 2017.

[Sag94] Hans Sagan. *Space-Filling Curves*. Ed. by J. H. Ewing, F. W. Gehring, and P. R. Halmos. Spinger-Verlag New York, 1994.

[Sam+17] Wojciech Samek et al. “Evaluating the Visualization of What a Deep Neural Network Has Learned”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.11 (Nov. 2017), pp. 2660–2673.

[SCD18] Andy Shih, Arthur Choi, and Adnan Darwiche. “A Symbolic Approach to Explaining Bayesian Network Classifiers”. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI’18. Stockholm, Sweden: AAAI Press, 2018, pp. 5103–5111.

[Sha53] Lloyd S. Shapley. “A Value for n-Person Games”. In: *Contributions to the Theory of Games II*. Ed. by Harold W. Kuhn and Albert W. Tucker. Princeton: Princeton University Press, 1953, pp. 307–317.

[SHM14] Daniel Soudry, Itay Hubara, and Ron Meir. “Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights”. In: *NIPS*. Vol. 1. 2014, p. 2.

[Sil+16] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (2016), pp. 484–489.

[Sil+17] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359.

[Sil+18] David Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. In: *Science* 362.6419 (2018), pp. 1140–1144.

[Sla+20] Dylan Slack et al. “Fooling lime and shap: Adversarial attacks on post hoc explanation methods”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 180–186.

[Smi+17] Daniel Smilkov et al. “SmoothGrad: removing noise by adding noise”. In: *CoRR* abs/1706.03825 (2017).

[Spr+15] Jost Tobias Springenberg et al. “Striving for Simplicity: The All Convolutional Net”. In: *ICLR (Workshop)*. 2015.

[Sri+14] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[Stu+16] Irene Sturm et al. “Interpretable Deep Neural Networks for Single-Trial EEG Classification”. In: *J. Neurosci. Methods* 274 (2016), pp. 141–145.

[SVZ13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *Preprint at https://arxiv.org/abs/1312.6034* (2013).

[SVZ14] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[SWM17] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. “Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models”. In: *CoRR* abs/1708.08296 (2017).

[SZ14] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014).

[TBR19] David Tuckey, Krysia Broda, and Alessandra Russo. “Saliency Maps Generation for Automatic Text Summarization”. In: *arXiv preprint arXiv:1907.05664* (2019).

[Tho+18] Armin W. Thomas et al. “Interpretable LSTMs For Whole-Brain Neuroimaging Analyses”. In: *Preprint at https://arxiv.org/abs/1810.09945* (2018).

[TLS21] Jacopo Teneggi, Alexandre Luster, and Jeremias Sulam. “Fast Hierarchical Games for Image Explanations”. In: *arXiv preprint arXiv:2104.06164* (2021).

[Vap91] Vladimir Vapnik. “Principles of risk minimization for learning theory”. In: *NIPS*. Vol. 91. 1991, pp. 831–840.

[Vid+15] Marina M-C Vidovic et al. “Opening the black box: Revealing interpretable sequence motifs in kernel-based learning algorithms”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2015, pp. 137–153.

[Vin+19] Oriol Vinyals et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. In: *Nature* 575.7782 (2019), pp. 350–354.

[Wäl+21] Stephan Wäldchen et al. “The Computational Complexity of Understanding Binary Classifier Decisions”. In: *Journal of Artificial Intelligence Research* 70 (Jan. 2021), pp. 351–387.

[Wat89] Christopher John Cornish Hellaby Watkins. “Learning from Delayed Rewards”. PhD thesis. Cambridge, UK: King’s College, 1989.

[Wer74] Paul Werbos. “Beyond regression:" new tools for prediction and analysis in the behavioral sciences”. In: *Ph. D. dissertation, Harvard University* (1974).

[Woo85] Steve Woolgar. “Why not a sociology of machines? The case of sociology and artificial intelligence”. In: *Sociology* 19.4 (1985), pp. 557–572.

[Yan+18] Yinchong Yang et al. “Explaining Therapy Predictions with Layer-Wise Relevance Propagation in Neural Networks”. In: *Proc. IEEE International Conference on Healthcare Informatics*. 2018, pp. 152–162.

[Yos+15] Jason Yosinski et al. “Understanding neural networks through deep visualization”. In: *Preprint at https://arxiv.org/abs/1506.06579* (2015).

[ZBM16] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. “Graying the black box: Understanding DQNs”. In: *Proc. International Conference on Machine Learning.* 2016, pp. 1899–1908.

[ZF14a] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *Proc. European Conference on Computer Vision.* 2014, pp. 818–833.

[ZF14b] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 818–833.

[Zho+16] Bolei Zhou et al. “Learning Deep Features for Discriminative Localization”. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 2921–2929.

A

Use cases of Explanation Methods

A.1 Layer-Wise Relevance Propagation

We present here the Layer-wise Relevance Propagation (LRP) approach proposed by [Bac+15a] for obtaining a pixel-wise decomposition of classifier decisions. We used this technique the experiments in chapter 3. LRP is a backpropagation-based approach and is applicable to most of the state-of-the-art architectures for image classification or neural reinforcement learning, including in particular AlexNet [KSH12], or Atari-based neural networks [Mni+15], and can also be applied to other types of models such as Fisher vector with SVM classifiers or LSTMs [HS97; Arr+17b].

We restate here the definition from chapter 1. Let $\Phi\colon \mathbb{R}^d \to \mathbb{R}$ with $d \in \mathbb{N}$ be a neural network of depth L, whose output for a specific input $\mathbf{x} \in \mathbb{R}^d$ is recursively defined as

$$\begin{aligned} \mathbf{x}^0 &= \mathbf{x}, \\ \mathbf{x}^l &= \rho\Big(\mathbf{W}\mathbf{x}^{l-1}\Big), \quad l = 1, \dots, L, \\ \Phi(\mathbf{x}) &= \mathbf{x}^L. \end{aligned}$$

The propagation rule of LRP is defined as a ‘mirror image‘ of network output as

$$\begin{aligned} \mathbf{r}^L &= \Phi(\mathbf{x}), \\ \mathbf{r}^{l-1} &= \mathbf{V}^l \mathbf{r}^l, \quad l = 1, \dots, L, && \text{(A.1)} \\ \mathbf{r} &= \mathbf{r}^0. && \text{(A.2)} \end{aligned}$$

Many versions of LRP have been proposed [Lap+16a; Bac+16] that are specified over the definition of $\mathbf{V}^l$ matrices. We give an overview over the most important ones and explain where they are used both in the literature as well as in our investigation. We restate the definitions as described in [Lap+16c; Alb+18a] in the following:

- The $\alpha\beta$-rule:
$$v_{ij} = \left(\alpha \frac{(w_{ij}x_j)^+}{\sum_k (w_{kj}x_j)^+} + \beta \frac{(w_{ij}x_j)^-}{\sum_k (w_{kj}x_j)^-} \right),$$
where $(x)^+ = \max(0, x)$, resp. $(x)^- = \min(0, -x)$, select for positive, resp. negative values. The values for $\alpha, \beta \in \mathbb{R}_{\geq 0}$ are tunable parameters of the method. This rule is used for the Atari agents in section A.3, as well as in [Bac+15a; Lap+16a; Bac+16].

- The w^2-rule:
$$v_{ij} = \frac{w_{ij}^2}{\sum_i w_{kj}^2}. \tag{A.3}$$
which is used in the bottom layer of the Atari agents.

- The ϵ-rule:
$$v_{ij} = \frac{w_{ij}x_j}{\sigma_\epsilon(\sum_k w_{kj}x_j)},$$
where $\sigma_\epsilon(t) = t + \epsilon\,\mathrm{sign}(t)$. The σ is used to keep the denominator from reaching zero, which causes numerical instability. The value of $\epsilon \in \mathbb{R}_{\geq 0}$ is a tunable parameter of the method. This method was used for the Fisher Vector classifiers in section 3.1.

- The flat rule:
$$v_{ij} = \frac{\mathbb{1}(w_{ij} \neq 0)}{\sum_k \mathbb{1}(w_{kj} \neq 0)},$$
This rule is used for the lower convolutional layers in the Atari agents for coarse graining, as well as in [Lap+16a; Bac+15a].

Additional Rules: The max-pooling layers are treated in this paper by redirecting all relevance to the neuron in the pool that has the highest activation. The Caffe reference model [Jia+14], as used for image categorization in Supplementary Note A.2, employs local renormalization layers. These layers are treated with an approach based on Taylor expansion [Bin+16]. Finally, pixel-wise relevance is obtained by pooling relevance over the RGB components of each pixel.

A.2 Task II: Image Categorization

Here we give an overview over the used classifiers as well as the preprocessing. The accuracy for each classifier can be found in table A.1 for each of the classes in the PASCAL VOC data set.

A.2.1 The Fisher Vector Classifier

The FV model evaluated in this section follows the configuration and implementation from [Cha+11; PSM10] and can easily be replicated by using the Encoding Evaluation Toolkit [Cha+11]. Twenty one-vs-all object detection models are trained on the `train_val` partition of the Pascal VOC 2007 dataset — one for each object class. The FV model pipeline consists of a local feature extraction step, computing dense SIFT descriptors [Low99] from an input image, which are then mapped to Fisher Vector representations via a Gaussian

	aer	**bic**	**bir**	**boa**	**bot**	**bus**	**car**
FV	79.08	66.44	45.90	70.88	27.64	69.67	80.96
DNN	88.08	79.69	80.77	77.20	35.48	72.71	86.30
	cat	**cha**	**cow**	**din**	**dog**	**hor**	**mot**
FV	59.92	51.92	47.60	58.06	42.28	80.45	69.34
DNN	81.10	51.04	61.10	64.62	76.17	81.60	79.33
	per	**pot**	**she**	**sof**	**tra**	**tvm**	**mAP**
FV	85.10	28.62	49.58	49.31	82.71	54.33	59.99
DNN	92.43	49.99	74.04	49.48	87.07	67.08	72.12

Table A.1: Prediction performances per model and class on the Pascal VOC 2007 test set in percent average precision.

mixture model (GMM) fit to the set of local descriptors of the whole training set. The FV representations of each image are then used for training a linear SVM [CV95] model. The parameters used in each step of the pipeline are as follows:

The images are normalized in size to a side length of at most 480 pixels, while maintaining the original aspect ratio. From a gray-scale version of the image, SIFT features are computed with spatial bin sizes of $\{4, 6, 8, 10\}$ pixels with fixed feature mask orientation at spatial strides of $\{4, 6, 8, 10\}$ pixels respectively. Using PCA, the descriptors are then projected from the originally 128-dimensional feature space to an 80-dimensional subspace, onto which a GMM with $k = 256$ Gaussian mixture components is fit. The trained GMM is then used as a soft visual vocabulary for mapping the projected local descriptors to a FV. The mapping process involves a spatial pyramid mapping scheme, subdividing an input image into 1×1, 3×1 and 2×2 grid areas. For each of the in total 8 image areas, a FV is computed based on the corresponding descriptor set, which are then concatenated to form one FV descriptor per image. After the application of power- and ℓ_2-normalization steps, reducing the sparsity of the vector and benefitting the model quality [Cha+11] this *Improved Fisher Vector* [PSM10] representation is used for training and prediction together with a linear SVM [CV95] classifier.

Evaluating the model in the `test` partition of the Pascal VOC 2007 dataset results in a 61.69% mAP score in [Cha+11] and 59.99% mAP in [Lap+16a].

A.2.2 The Multilabel Deep Neural Network

In order to contrast a Deep Neural Network to the previously introduced FV classifier, a pre-trained model has been fine-tuned on the Pascal VOC data and classes to achieve a comparable evaluation setting. For fine-tuning, we use the Caffe Deep Learning Framework [Jia+14] and the pre-trained BVLC Caffe reference model available from `http://dl.caffe.berkeleyvision.org` as a starting point. The model has been adjusted to match the number of Pascal VOC classes by setting the number of output neurons of the topmost fully connected layer to 20. The softmax layer has been replaced with a standard hinge loss layer to accommodate for the appearance of multiple object classes per input sample as it is the case with the Pascal VOC data.

The training dataset consists of the `train_val` partition of the Pascal VOC 2012 dataset, augmented by first resizing each image to 256 pixels on its longest side and padding the image by repeating the border pixels to a size of 256×256, then cropping 227×227 sized subimages at the corners and the image center. Together with a horizontally mirrored version for each of the subimages, the tenfold augmented training data then counts $115,300$ training samples. Fine-tuning is performed for $15,000$ iterations with training patches holding 256 samples per iteration and a fixed learning rate of 0.001. The resulting model achieves an mAP score of 72.12% on the Pascal VOC 2007 dataset. We have made this fine-tuned multi-label model available as part of the Caffe Model Zoo[1].

A.2.3 Comparing Fisher Vector and Deep Network Prediction Strategies

We compare the reasoning of two different classifiers — an Improved Fisher Vector SVM classifier from [Cha+11; PSM10] as a state-of-the-art Bag of Words-type model to the much deeper convolutional neural network architecture described before — by analyzing the relevance feedback obtained via LRP on pixel level. Both models are trained / fine-tuned to predict the 20 object classes defined in the Pascal VOC datasets. In order to compute pixel-level relevance scores for the FV model, we employ the ϵ-rule to obtain a numerically stable decomposition of the FV mapping step and the "flat" rule to project local descriptor relevances to pixels. The relevance decompositions for the SVM predictor layer are computed using the "simple" ($\epsilon = 0$) rule (cf. [Lap+16a] and [Bac+16] for further technical details). Since the fine-tuned DNN model structure is composed of a repeating sequence of convolutional or fully connected layers interleaved with pooling and ReLU activation layers, we uniformly apply the $\alpha\beta$-rule with $\alpha = 2, \beta = -1$ throughout the network, which complements the ReLU-activated inputs fed into hidden layers especially well for explanation. Since the output of ReLUs is strictly ≥ 0, positive weights in the succeeding convolutional or linear layer do carry the input activation to further layers, whereas negatively weighted connections serve as inhibitors to a layer's output signals. Using the $\alpha\beta$ decomposition rule allows for a numerically stable relevance decomposition, where the parameter α controls the balance of neuron exciting and neuron inhibiting patterns for the explanation. We presented our results for the "boat" and "horse" classes in chapter 3. We add a more detailed analysis for the "boat" class in figs. A.1 and A.2.

[1]`https://github.com/BVLC/caffe/wiki/Model-Zoo#pascal-voc-2012-multilabel-classification-model`

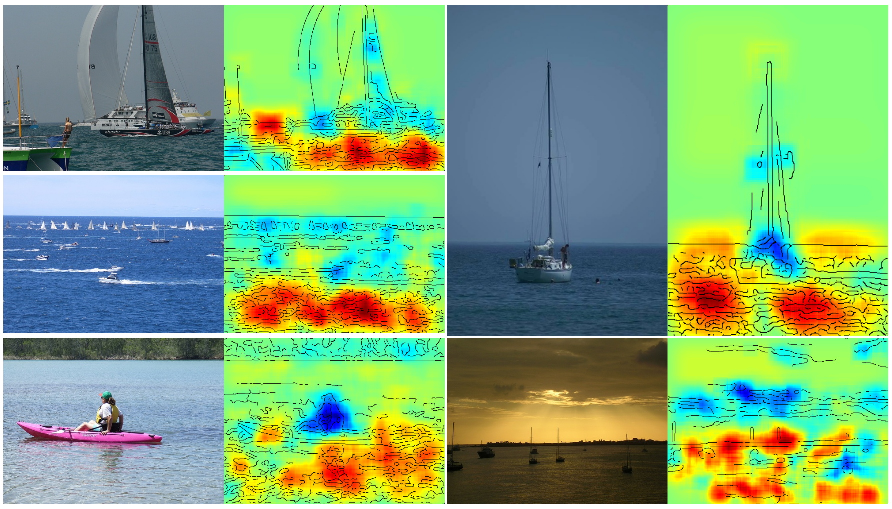


Figure A.1: Several images showing boats below the horizon line, with predictions explained for the "boat"-class model. In all images, the water itself receives strong positive relevance scores. *Top Left* and *Top Right*: Both sail boats are located in front of the horizon, receiving negative relevance scores in the respective image areas. Unobstructed horizon lines in both images yield information indicating the presence of class "boat". *Middle Left*: Distant sails and motor boats receive negative relevance scores, while the water surface in the bottom part of the image strongly indicates the target class. *Bottom Left*: The features extracted for the human visually blocking the water receive strong negative relevance values, even though class "person" frequently appears together with class "boat". The visual information obtained from the image area showing the person strongly contradicts the learned concept of class "boat". *Bottom Right:* In this scene, masts crossing the horizon and the water itself count towards the concept of "boat", while some vessels below the horizon — both boats to the middle left aligned with the camera taking the image and both boats to the right — are yielding disruptive visual features. See also Supplementary Figure A.2 for complementary heatmaps computed for class "boat" based on the DNN model.

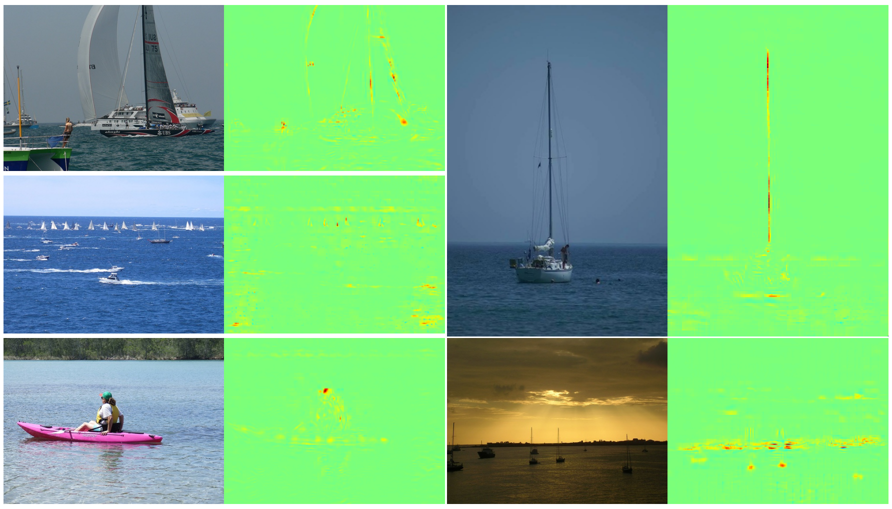


Figure A.2: Supplementary Figure as a counterpart for Supplementary Figure A.1, showing the additional relevance responses for class "boat" for the fine-tuned DNN model. Due to the filigree character of the relevance response no edge maps have been drawn on top of the heatmaps. The relevance scores follow the boat-like features very closely, rendering the heatmaps well-readable and the relevance scores easy to localize. In contrast as shown by the heatmaps for the FV model in Supplementary Figure A.1, the DNN does not regard boats below the horizon line as disrupting objects. Frequently, even people sitting or working (sail) on boats are rates as contributing factors to class "boat", which can be attributed to the label "person" appearing in almost every third "boat" image.

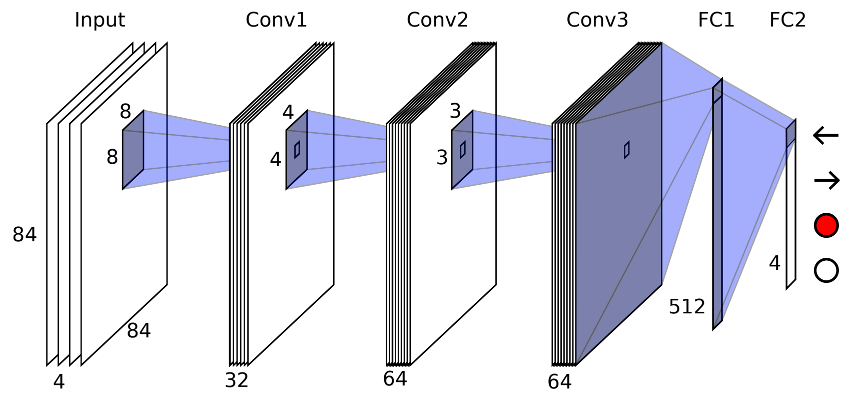


Figure A.3: The Atari Network architecture. The network consists of three convolutional layers (Conv) and two fully connected layers (FC). All layers comprise a ReLU-nonlinearity except for the last FC layer. The network output corresponds to a vector that predicts the Q-function for every possible action, in this example "go left", "go right", "fire" and "do nothing".

A.3 Playing Atari Games

This section explains the details of how the neural network agents for the Atari games Pong and Breakout have been trained, as well as how exactly the relevance maps have been produced.

A.3.1 Training

The first intelligence task inspected studies neural network agents playing simple Atari video games. The training of these models has been performed using a Python- and Theano-based implementation, which is publicly available from `https://github.com/spragunr/deep_q_rl` and implements the system described in [Mni+15]. The method uses a modification to Equation 3.2 to calculate the long-term reward of the next action and considers both the current set of model parameters θ, as well as an older, temporarily fixed version thereof θ^*, where after every k steps, θ^* will be updated to the values of θ. Having a different set of parameters for the target and for the prediction stabilizes the training process. The resulting update step for the network parameters is

$$\theta_{n+1} = \theta_n + \alpha \sum_{(s,a,r,s')\in B} \nabla_{\boldsymbol{\theta}} Q(s,a;\boldsymbol{\theta}) D\Big(Q(s,a;\boldsymbol{\theta}) - \Big[r + \gamma \max_{a'} Q(s',a';\boldsymbol{\theta}^*)\Big]\Big), \qquad \text{(A.4)}$$

where α is the learning rate of the training process. Following the approach of [Mni+15], the function D clips the difference term at the end of Equation A.4 between $[-1\ 1]$ to curb oscillations in updates where the network is far from satisfying the Bellman equation. This is equivalent to employing a quadratic error term until the value 1 and a constant error term beyond.

The network consists of three convolutional layers and two inner product layers. The exact architecture from [Mni+15] is described in Supplementary Figure A.3 and in Supplementary Table A.2. An input state corresponds to the last four frames of game visuals as seen by a human player, transformed into gray-scale brightness values and scaled to 84×84 pixels in size, is fed to the network as a $4 \times 84 \times 84$-sized tensor with pixel values rescaled between 0 (black) and 1 (white). The network prediction is a vector of the expected long-term reward for each possible action, where the highest rated action is then passed as input to the game.

Every action is repeated for four time steps (i.e. every four frames the model receives a new input spanning four frames and makes a decision which is used as input for the next four frames) which corresponds to the typical amount of time it takes for a human player to press a button. With a probability of 10%, the trained agent will choose a random action instead of using the action predicted to be the most valuable option; as we will see in our analysis this randomness is essential. Upon training a completely fresh model, the probability of picking actions at random is initiated at 100% and linearly over time lowered to 10% with ongoing training. As suggested in [Mni+15], we use stochastic gradient descent with a batch size of 32 and a learning rate of $\alpha = 2.5 \cdot 10^{-4}$. Every fourth update step, the parameters from θ are copied to θ^*. During training, a replay memory is maintained as an active training set. The memory is updated as a FIFO-style buffer structure, as soon as its capacity of 10^6 time steps or observations is exhausted. Model weight updates are performed as soon as $5 \cdot 10^4$ observations have been collected.

A.3.2 Explanations

We use LRP to explain how the network decides which action to take. Specifically, we use the $\alpha\beta$-rule, for LRP (see Supplementary Table A.1) with parameters $\alpha = 1$ and $\beta = 0$. It distributes relevance proportionally to the positive forward contribution in every layer. In the input layer, the black background (black) of the game corresponds to a gray-scale value of zero and would receive no relevance. To be able to distribute relevance onto pixels in the lowest layer, which can have zero values , we resort to the w^2-rule described in [Mon+17], see A.3, which was designed to address this situation.

LRP is used to back-propagate the dominant action (the action with the highest activation value) through the network onto the input which results in a $4 \times 84 \times 84$-sized tensor of relevance values. To create a heatmap, we sum over the first axis and normalise the values to a value range of $[-1\ 1]$ per pixel — $R_i \leftarrow R_i/(\max_j |R_j|)$ — and rescale the image via linear interpolation to the original size of 210×160 pixels. To give an intuition for interpreting the heatmaps, we demonstrate how they correspond to different situations and strategies for two examples, the games *Breakout* and *Video Pinball*, both of which show above-human performance for the trained learning machines [Mni+15].

A.3.2.1 Varying Depth of Architecture

In this section we compare the relevance responses of neural network agents with different model architectures, namely the architecture from [Mni+15] (denoted as Nature architecture), the architecture of its predecessor in [Mni+13], (denoted as NIPS architecture) and a third network with an architecture similar to [Mni+15] (Small architecture). The NIPS architecture has one convolutional layer less than its successor, whereas the Small architecture has the same number of convolutional layers as the Nature network, but is limited in its decision making capacity by only one fully connected layer instead of two. An overview in the different network architectures is given in Supplementary Table A.2.

For the NIPS architecture, the focus on ball and paddle is delayed, emerging approximately 5 training epochs later when compared to the Nature network (see Supplementary Figure A.4). The relevance on the tunnel region shifts in two stages and finally rises to an amount higher

NIPS architecture	Nature architecture	Small architecture
C1 (4×8×8)→(16), [4×4]	C1 (4×8×8)→(32), [4×4]	C1 (4×8×8)→(32), [4×4]
C2 (16×4×4)→(32), [2×2]	C2 (32×4×4)→(64), [2×2]	C2 (32×4×4)→(64), [2×2]
	C3 (64×3×3)→(64), [1×1]	C3 (64×3×3)→(64), [1×1]
F1 (2592)→(256)	F1 (3136)→(512)	F1 (3136)→(4)
F2 (256)→(4)	F2 (512)→(4)	

Table A.2: A comparison of the investigated network architectures. To describe a layer, we use the notation $(I) \to (O), [S]$ where I is the shape of the (convolutional) weights, O are the output responses and S is the stride for a convolutional layer. For the Nature network, C1 (4×8×8)→(32), [4×4] describes the first convolutional layer with a learned filter bank counting 32 convolutional 8×8 filter tensors of 4 input channels each, applied at a 4×4 stride across the layer input. Similarly, F2 (512)→(4) describes the second and in this case last fully connected layer with 512 input nodes and 4 outputs.

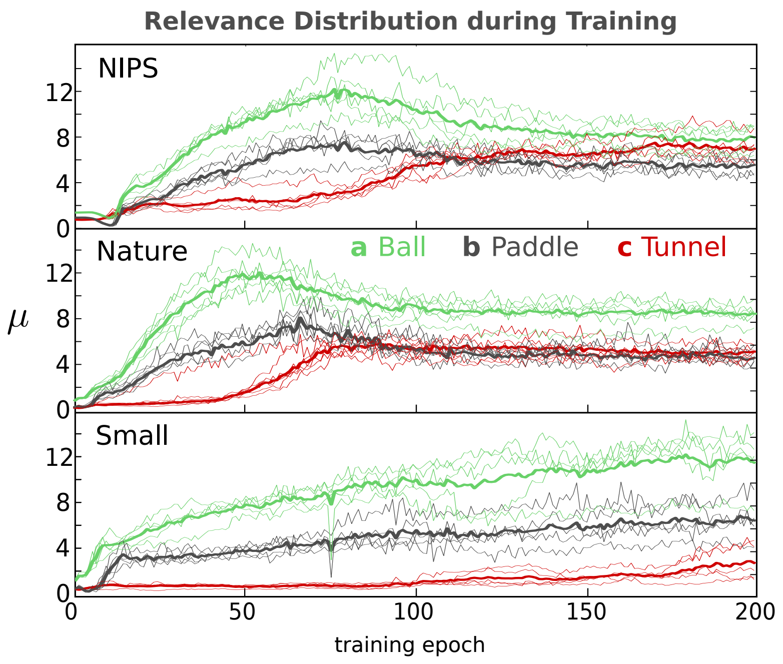


Figure A.4: Comparison of the network attention in terms of relevance responses of three different model architectures to important game elements (ball, paddle and tunnel) as a function of training time. An overview of the compared model architectures is given in Supplementary Table A.2. All three architectures show a shift in strategy towards tunnel building. For the NIPS architecture the shift appears in two stages resulting in a larger shift in total, whereas for the Small architecture the tunnel does not become a relevant objective during the training period.

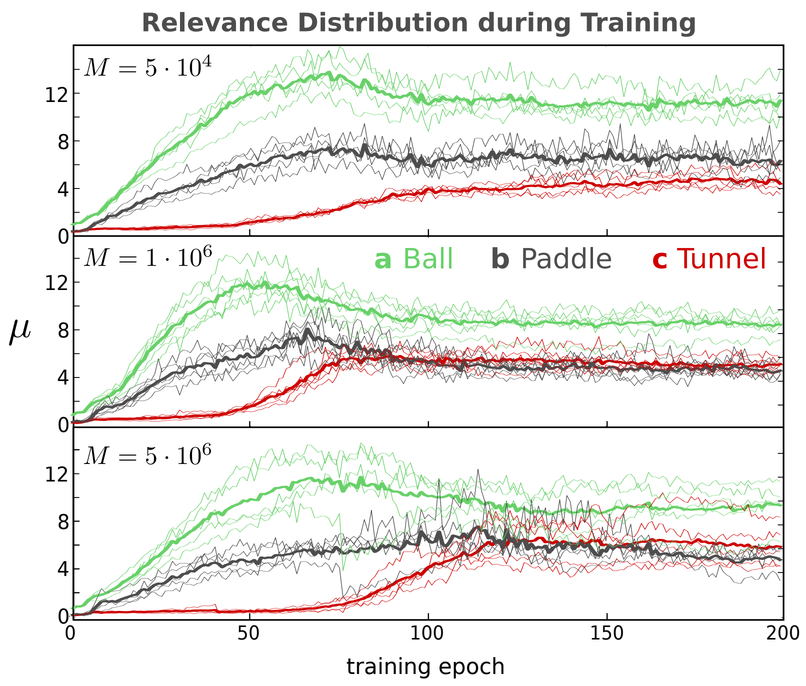


Figure A.5: Evolution of network attention on the game elements ball, paddle and tunnel for varying replay memory sizes M, with $M = 5 \cdot 10^4$, $M = 10^6$ and $M = 5 \cdot 10^6$. All three memory settings allow for a shift of strategy towards tunnel building.

than that recorded for the Nature architecture. A possible interpretation is that the shallower convolutional architecture slows down the recognition of the important game elements, but once recognized the development of higher level strategies is possible.

For the Small architecture, the focus switches very early to ball and paddle. The three convolutional layers allow for a quick recognition of the important game objects. However, the game play performance remains at a sub-human level and more advanced strategies like tunnel building remain out of focus. The agent learns to follow the ball with the paddle, but is only able to successfully reflect the ball in three out of four interactions.

A.3.2.2 Varying Size of Replay Memory

We investigate the influence of replay memory size on the development of a neural network agent. The replay memory is a dataset of previous game situations described as the tuple `(state, action, reward, next state)`. After each interaction with the Atari emulator one new game situation is added to the memory. Once the memory capacity is reached, the earliest entries are replaced with newer ones. For training, a random batch is drawn from the memory.

In Supplementary Figure A.5 we present the development of the relevance of the ball, paddle and tunnel to neural network agents, for memories of capacities $5 \cdot 10^4$, 10^6 and $5 \cdot 10^6$. For all memory size settings, the corresponding models shift their attention to the tunnel area with ongoing training. For the smallest memory size of $5 \cdot 10^4$ game states we note a more gradual shift in strategy that starts earlier in training and leads to less concentrated model attention registered on the tunnel area. The largest memory of $5 \cdot 10^6$ game states shifts to the advanced strategy at a later time, but more abruptly so. Relevance scores concentrate more on the tunnel area when compared to the other models. A later onset of strategically recognizing the tunnel during training as an effect of replay memory size can be attributed to the vast amount of early game states still populating the memory. Many more training epochs are needed to fully eliminate the effect of those game states that have resulted from many short games with uncontrolled game play in early training epochs. A smaller replay memory might replace those game state recordings faster, but does not allow for enough memorized observations of later phases of the game (or full games played more expertly) to learn the tunnel building strategy successfully.

A.3.2.3 Changes of Model Attention during Game Play

Over the course of a running game of *Atari Breakout*, we investigate how and where relevance allocation changes over time. To visualise heatmaps over time, the relevance map computed for each frame is summed over the horizontal axis, resulting in a relevance vector with an entry for each vertical pixel location. Then, the compacted frames from all time steps are concatenated column-wise with the time now being the new horizontal axis. Relevance values attributed to the ball, the paddle and the tunnel can still be distinguished since they have different vertical positions for most of the time. We complement this analysis with information about at which time steps or events, certain actions are predicted significantly stronger than the other options. For a trained network, the Q-values of all possible actions generally only differ by relatively small quantities. If the ball is far from the paddle, for example, the next action does not influence the predicted score and game play much. At points in time when single actions do have a large impact, the predicted Q-values diverge. We measure this relative deviation of Q-values as

$$\Delta q = \frac{q_{max} - q_{min}}{q_{max}}$$

where q_{max} and q_{min} are the maximum and minimum predicted Q-value for a given input, respectively. Over the course of training, the networks become more decisive. Supplementary Figure A.6 shows Δq during a game sequence of 500 frames for three different stages in training: After two, twenty and two hundred epochs of training, with 100.000 training steps per epoch. We observe that the fully trained network's prediction output focuses on single output neurons at times when it has to interact with the ball, while earlier versions of the same network are not yet fully able to decide with a comparable amount of certainty.

The apparent *zig-zag* structure in the heatmap visualizations in cupplementary A.6 is the up-and-down movement of the ball, as tracked by the neural network agent over time. Applying LRP allows us to track the ball without having to design a specific tracker. Combining this with the temporal focus of the network we can discern at which height of the ball the decision for an action is made. Supplementary fig. A.6 illustrates these more important action predictions for a fully trained network as red circles and visualize the change in relevance allocation over the time of a game. Note that the network itself becomes more focused over training time (cf. the discussion in Supplementary Note 3.2).

Supplementary Figure 3.8 presents the results obtained when using Sensitivity Analysis instead using LRP for the setup of Supplementary fig. 3.7. Observing the gradient maps during game play reveals that the saliency follows the general area where the ball is located, but is *not* focused on the ball itself. The response obtained with Sensitivity Analysis, when

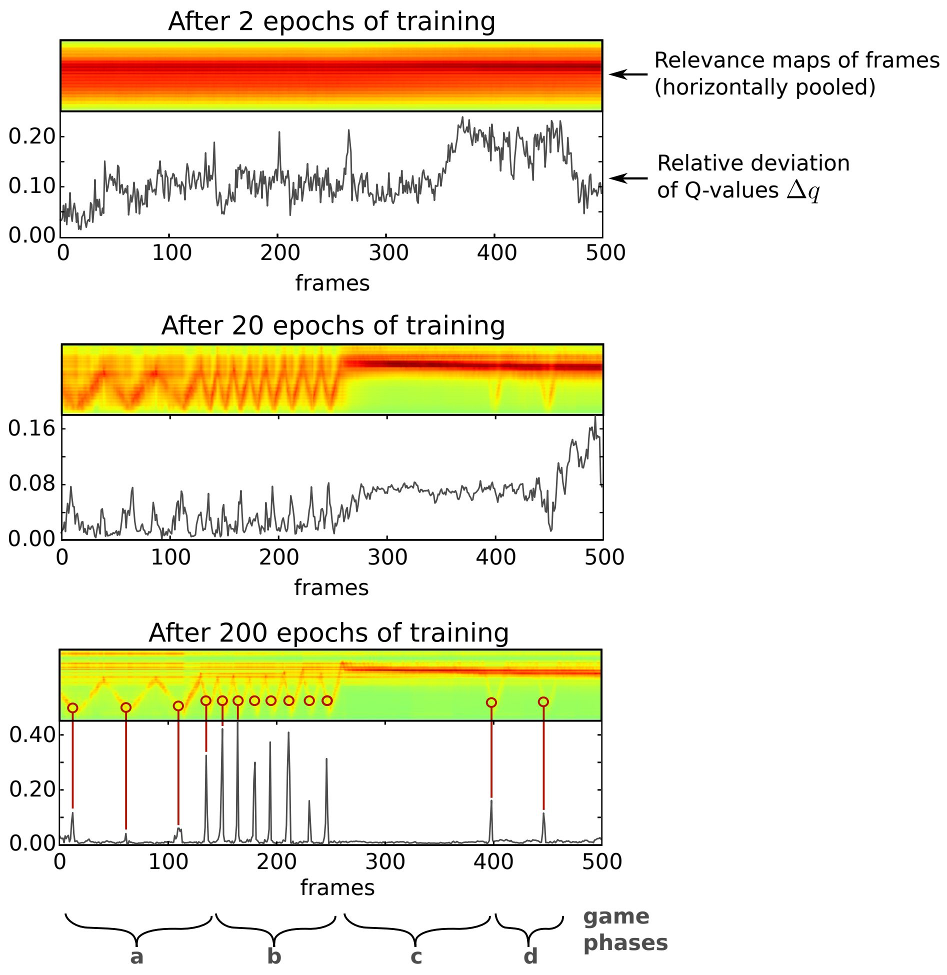


Figure A.6: Divergence of Q-values over game time, contrasted with horizontally pooled heatmaps. a) The ball moves slowly in this early phase of the game, only small spikes in Δq are recorded. b) The ball increases in speed after hitting a brick in a layer beyond the lowest two. Spikes in Δq increase as recovery time after a mispredicted action is decreased. c) The ball breaks through and stays above the wall. Temporarily no interaction from the agent is necessary. d) The ball moves back down twice and is reflected by the paddle. To increase visual readability of the network attention, gamma correction with $\gamma = 0.5$ has been applied after normalizing the heatmaps, increasing the contrast of the color coded heatmaps responses. We compare the Δq and the horizontally pooled heatmaps between three selected training epochs, i.e. after 2, 20 and 200 epochs of training. At certain frames in the game, Δq becomes much larger than zero, signifying the expected reward of one action dominating the others. When it makes no difference which action is taken, since the ball is far from the paddle, the correct long-term reward of all actions will be close and Δq small. Late-stage networks appear to incorporate this fact – they become more focused on certain frames. After two epochs of training, Δq fluctuates strongly with no apparent focus. After 20 epochs, peaks of diverging q-values emerge. After 200 epoch, Δq is generally close to zero except for frames in which a falling ball has reached a height were action is necessary. Complementing the temporal focusing is a spatial focusing of the heatmaps on important game objects, see fig. 3.7. We illustrate this showing the horizontally pooled heatmaps over the game time. Over training, a narrower portion of the game frame gets an increasing portion of the relevance, making the focus on the ball apparent.

given a model and an input data point, measures how much a change in the input would affect the model prediction. Here, peaking gradient values in close proximity to the ball essentially indicate where the ball should be to cause the steepest change in predicted Q-values, compared to where it actually is. Furthermore, for each of the neural network agents trained, gradient maps did not attribute weights on the pixels covering the tunnel element. The changes in game play strategy are not observable through Sensitivity Analysis.

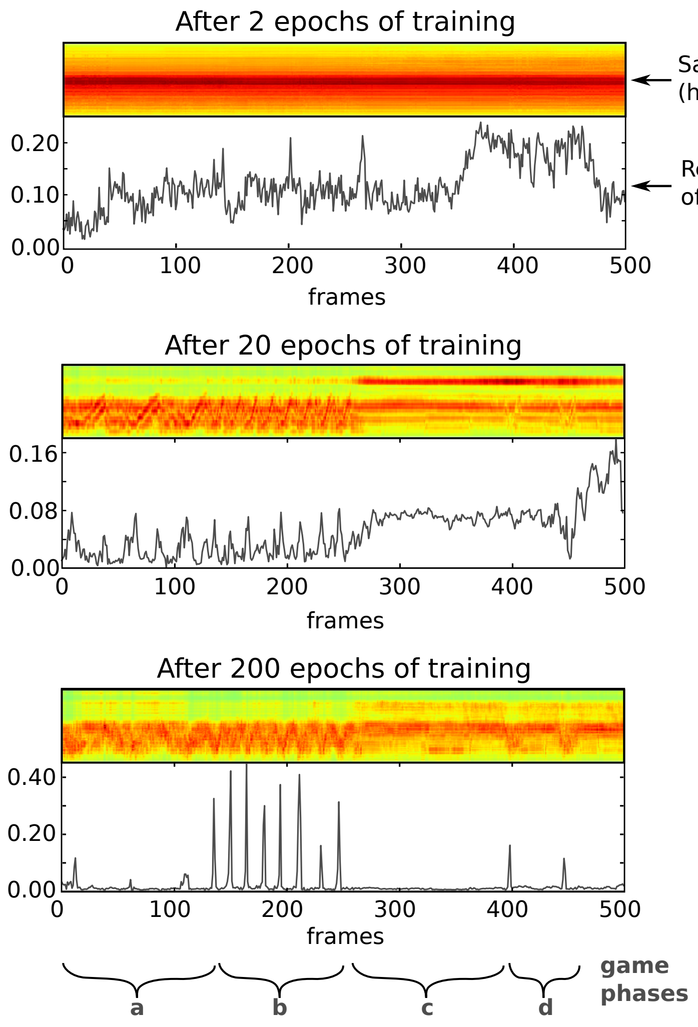


Figure A.7: a)–d): Compare to supplementary fig. A.6. The gradient map obtained by Sensitivity Analysis gives a less clear picture than LRP.

B

Probabilistic Prime Implicants

B.1 Raising the Probability

We give constructive proofs of lemmas 4.20 and 4.24, starting with the first.

Proof of lemma 4.20: Let $\Psi\colon \{0,1\}^d \to \{0,1\}$ be arbitrary and $0 \leq \delta_1 < \delta_2 < 1$. We will construct a monotone function $\Pi\colon \{0,1\}^n \to \{0,1\}$ such that

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) > \delta_1 \quad \iff \quad P_{(\mathbf{y},\mathbf{r})}(\Psi(\mathbf{y}) \vee \Pi(\mathbf{r})) \geq \delta_2, \tag{B.1}$$

with

$$n \in \mathcal{O}\bigg(\Big(d + \log_2\Big(\frac{1-\delta_1}{1-\delta_2}\Big)\Big)^2\bigg).$$

In our context δ_1 and δ_2 are constant and therefore $n \in \mathcal{O}(d^2)$.

Denote $\Psi'\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}\colon (\mathbf{y},\mathbf{r}) \mapsto \Psi(\mathbf{y}) \vee \Pi(\mathbf{r})$, then

$$P(\Psi') = P(\Psi) + (1 - P(\Psi))P(\Pi), \tag{B.2}$$

which is monotonically increasing in both $P(\Psi)$ and $P(\Pi)$. Thus, it suffices to consider the edge case when $P(\Psi)$ is close to δ_1. Since $P(\Psi)$ can only take values in $\left\{\frac{0}{2^d}, \frac{1}{2^d}, \ldots, \frac{2^d}{2^d}\right\}$ we see that (B.1) is equivalent to the two conditions

$$\begin{aligned} P(\Psi) &= \frac{\lfloor \delta_1 2^d \rfloor}{2^d} &&\implies && P(\Psi') < \delta_2, \\ P(\Psi) &= \frac{\lfloor \delta_1 2^d \rfloor + 1}{2^d} &&\implies && P(\Psi') \geq \delta_2, \end{aligned}$$

which together with (B.2) is equivalent to

$$\frac{\lfloor \delta_1 2^d \rfloor}{2^d} + \frac{2^d - \lfloor \delta_1 2^d \rfloor}{2^d} P(\Pi) < \delta_2 \tag{B.3}$$

$$\frac{\lfloor \delta_1 2^d \rfloor + 1}{2^d} + \frac{2^d - \lfloor \delta_1 2^d \rfloor - 1}{2^d} P(\Pi) \geq \delta_2. \tag{B.4}$$

In the case $\delta_1 < \delta_2 \leq \frac{\lfloor \delta_1 2^d \rfloor + 1}{2^d}$ these conditions are already fulfilled if we simply set $\Pi \equiv 0$. Otherwise, if $\delta_2 > \frac{\lfloor \delta_1 2^d \rfloor + 1}{2^d}$, rearranging (B.3) and (B.4) yields the bounds

$$a \leq P(\Pi) < b$$

on $P(\Pi)$, where

$$a = \frac{\delta_2 2^d - \lfloor \delta_1 2^d \rfloor - 1}{2^d - \lfloor \delta_1 2^d \rfloor - 1},$$
$$b = \frac{\delta_2 2^d - \lfloor \delta_1 2^d \rfloor}{2^d - \lfloor \delta_1 2^d \rfloor}.$$

It is not hard to check that indeed we have $0 \leq a < b \leq 1$.

In appendix B.3 we show for $\eta \in [0,1]$ and $\ell \in \mathbb{N}$ the existence of a monotone DNF-function $\Pi_{\eta,\ell} \colon \{0,1\}^n \to \{0,1\}$ such that $\Pi_{\eta,\ell}(\mathbf{0}_n) = 0$, $\Pi_{\eta,\ell}(\mathbf{1}_n) = 1$, and

$$|P(\Pi_{\eta,\ell}) - \eta| \leq 2^{-\ell}$$

with $n \leq \frac{\ell(\ell+3)}{2} \in \mathcal{O}(\ell^2)$. We conclude by choosing

$$\eta = \frac{b+a}{2},$$
$$\ell = \left\lfloor -\log_2\left(\frac{b-a}{2}\right) \right\rfloor + 1 \in \mathcal{O}\left(d + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right),$$

and setting $\Pi = \Pi_{\eta,\ell}$. We get $n \in \mathcal{O}(\ell^2) = \mathcal{O}\left(\left(d + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right)^2\right)$, which finishes the proof of lemma 4.20.

Proof of lemma 4.24: We proceed analogously to before. For $0 \leq \delta_1 \leq \delta_2 < 1$, we construct a monotone function $\Pi \colon \{0,1\}^n \to \{0,1\}$ such that

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) \geq \delta_1 \quad \Longleftrightarrow \quad P_{(\mathbf{y},\mathbf{r})}(\Psi(\mathbf{y}) \vee \Pi(\mathbf{r})) > \delta_2,$$

with

$$n \in \mathcal{O}\left(\left(d + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right)^2\right).$$

Again, δ_1 and δ_2 are constant in our setting and therefore $n \in \mathcal{O}(d^2)$.

Similar to before, in the case that $\delta_1 \leq \delta_2 < \frac{\lceil \delta_1 2^d \rceil}{2^d}$, we can simply set $\Pi \equiv 0$. Otherwise, we get the bounds

$$a < P(\Pi) \leq b$$

with

$$a = \frac{\delta_2 2^d - \left\lceil \delta_1 2^d \right\rceil}{2^d - \lceil \delta_1 2^d \rceil},$$
$$b = \frac{\delta_2 2^d - \left\lceil \delta_1 2^d \right\rceil + 1}{2^d - \lceil \delta_1 2^d \rceil + 1}.$$

Again, we can check that $0 \leq a < b \leq 1$, and set

$$\eta = \frac{b+a}{2},$$
$$\ell = \left\lfloor -\log_2\left(\frac{b-a}{2}\right) \right\rfloor + 1 \in \mathcal{O}\left(d + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right),$$

and $\Pi = \Pi_{\eta,\ell}$ with $n \in \mathcal{O}(\ell^2) = \mathcal{O}\left(\left(d + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right)^2\right)$, which concludes the proof of lemma 4.24.

B.2 Lowering the Probability

We give constructive proofs of lemmas 4.21 and 4.25, starting with the first.

Proof of lemma 4.21: Let $\Psi\colon \{0,1\}^d \to \{0,1\}$ be arbitrary and $0 < \delta_1 \leq \delta_2 \leq 1$. We will construct a monotone function $\Pi\colon \{0,1\}^n \to \{0,1\}$ such that

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) > \delta_2 \quad \iff \quad P_{(\mathbf{y},\mathbf{r})}(\Psi(\mathbf{y}) \wedge \Pi(\mathbf{r})) \geq \delta_1, \tag{B.5}$$

with

$$n \in \mathcal{O}\left(\left(d + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right)^2\right).$$

In our context δ_1 and δ_2 are constant and therefore $n \in \mathcal{O}(d^2)$.

Denote $\Psi'\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}\colon (\mathbf{y},\mathbf{r}) \mapsto \Psi(\mathbf{y}) \wedge \Pi(\mathbf{r})$, then

$$P(\Psi') = P(\Psi)P(\Pi), \tag{B.6}$$

which is monotonically increasing in both $P(\Psi)$ and $P(\Pi)$. Thus, it suffices to consider the edge case when $P(\Psi)$ is close to δ_2. Since $P(\Psi)$ can only take values in $\left\{\frac{0}{2^d}, \frac{1}{2^d}, \ldots, \frac{2^d}{2^d}\right\}$ we see that (B.5) is equivalent to the two conditions

$$P(\Psi) = \frac{\lfloor \delta_2 2^d \rfloor}{2^d} \quad \implies \quad P(\Psi') < \delta_1,$$
$$P(\Psi) = \frac{\lfloor \delta_2 2^d \rfloor + 1}{2^d} \quad \implies \quad P(\Psi') \geq \delta_1,$$

which together with (B.6) is equivalent to

$$\frac{\lfloor \delta_2 2^d \rfloor}{2^d} P(\Pi) < \delta_1 \tag{B.7}$$

$$\frac{\lfloor \delta_2 2^d \rfloor + 1}{2^d} P(\Pi) \geq \delta_1. \tag{B.8}$$

In the case $\frac{\lfloor \delta_2 2^d \rfloor}{2^d} < \delta_1 \leq \delta_2$ these conditions are already fulfilled if we simply set $\Pi \equiv 1$. Otherwise, if $\delta_1 \leq \frac{\lfloor \delta_2 2^d \rfloor}{2^d}$, rearranging (B.7) and (B.8) yields the bounds

$$a < P(\Pi) \leq b$$

on $P(\Pi)$, where

$$a = \frac{\delta_1 2^d}{\lfloor \delta_2 2^d \rfloor + 1},$$
$$b = \frac{\delta_1 2^d}{\lfloor \delta_2 2^d \rfloor}.$$

It is not hard to check that indeed we have $0 \leq a < b \leq 1$.

In appendix B.3, we show for $\eta \in [0,1]$ and $\ell \in \mathbb{N}$ the existence of a monotone DNF-function $\Pi_{\eta,\ell} \colon \{0,1\}^n \to \{0,1\}$ such that $\Pi_{\eta,\ell}(\mathbf{0}_n) = 0$, $\Pi_{\eta,\ell}(\mathbf{1}_n) = 1$, and

$$|P(\Pi_{\eta,\ell}) - \eta| \leq 2^{-\ell}$$

with $n \leq \frac{\ell(\ell+3)}{2} \in \mathcal{O}(\ell^2)$. We conclude by choosing

$$\eta = \frac{b+a}{2},$$
$$\ell = \left\lfloor -\log_2\left(\frac{b-a}{2}\right) \right\rfloor + 1 \in \mathcal{O}\left(d + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right),$$

and setting $\Pi = \Pi_{\eta,\ell}$. We get $n \in \mathcal{O}(\ell^2) = \mathcal{O}\left(\left(d + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right)^2\right)$, which finishes the proof of lemma 4.21.

Proof of lemma 4.25: We proceed analogously to before. For $0 < \delta_1 < \delta_2 \leq 1$, we construct a monotone function $\Pi \colon \{0,1\}^n \to \{0,1\}$ such that

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) \geq \delta_2 \iff P_{(\mathbf{y},\mathbf{r})}(\Psi(\mathbf{y}) \wedge \Pi(\mathbf{r})) > \delta_1,$$

with

$$n \in \mathcal{O}\left(\left(d + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right)^2\right).$$

Again, δ_1 and δ_2 are constant in our setting and therefore $n \in \mathcal{O}(d^2)$.

Similar to before, in case that $\frac{\lceil \delta_2 2^d \rceil - 1}{2^d} \leq \delta_1 < \delta_2$, we can simply set $\Pi \equiv 1$. Otherwise, we get the bounds

$$a < P(\Pi) \leq b$$

with

$$a = \frac{\delta_1 2^d}{\lceil \delta_2 2^d \rceil}$$
$$b = \frac{\delta_1 2^d}{\lceil \delta_2 2^d \rceil - 1}.$$

Again, we can check that $0 \leq a < b \leq 1$, and set

$$\eta = \frac{b+a}{2},$$
$$\ell = \left\lfloor -\log_2\left(\frac{b-a}{2}\right)\right\rfloor + 1 \in \mathcal{O}\left(d + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right),$$

and $\Pi = \Pi_{\eta,\ell}$ with $n \in \mathcal{O}(\ell^2) = \mathcal{O}\left(\left(d + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right)^2\right)$, which concludes the proof of lemma 4.25.

B.3 Construction of the Functions $\Pi_{\eta,\ell}$

For $\eta \in [0,1]$ (the target probability) and $\ell \in \mathbb{N}$ (the accuracy) we construct a Boolean function $\Pi_{\eta,\ell}\colon \{0,1\}^n \to \{0,1\}$ in disjunctive normal form with $n \in \mathcal{O}(\ell^2)$, $\Pi_{\eta,\ell}(\mathbf{0}_n) = 0$, $\Pi_{\eta,\ell}(\mathbf{1}_n) = 1$, and

$$|\eta - P(\Pi_{\eta,\ell})| \leq 2^{-\ell}.$$

If $\eta \leq 2^{-\ell}$, we can simply choose $\Pi_{\eta,\ell}(x_1,\ldots,x_\ell) = \bigwedge_{k=1}^{\ell} x_k$. So from now on assume $2^{-\ell} < \eta \leq 1$. In this case we construct a sequence of functions $\Pi_i\colon \{0,1\}^{n_i} \to \{0,1\}$ such that $p_i = P(\Pi_i)$ is monotonically increasing and converges to η from below. We proceed according to the following iterative procedure: Start with the constant function $\Pi_0 \equiv 0$. Given Π_i and p_i we can stop and set $\Pi_{\eta,\ell} = \Pi_i$ if $|\eta - p_i| \leq 2^{-\ell}$. Otherwise, we set $n_{i+1} = n_i + \Delta n_i$ with

$$\Delta n_i = \operatorname{argmin}\{\, n \in \mathbb{N} : p_i + (1-p_i)2^{-n} \leq \eta \,\}, \tag{B.9}$$

and

$$\Pi_{i+1}(x_1,\ldots,x_{n_{i+1}}) = \Pi_i(x_1,\ldots,x_{n_i}) \vee \left(\bigwedge_{k=n_i+1}^{n_{i+1}} x_k\right).$$

Clearly, we obtain $p_{i+1} = p_i + (1-p_i)2^{-\Delta n_i}$. We will see below that Δn_i can not be too large and thus (B.9) can be efficiently computed by sequential search.

Lemma B.1 The sequence $(p_i)_{i\in\mathbb{N}}$ is monotonically increasing and we have

$$|\eta - p_{i+1}| \leq \frac{1}{2}|\eta - p_i|$$

for all $i \in \mathbb{N}$. In particular $|\eta - p_i| \leq 2^{-i}$ and $p_i \to \eta$ as $i \to \infty$.

Proof. Since $0 = p_0 \leq \eta$ and by choice of Δn_i, we have $p_i \leq \eta$ for all $i \in \mathbb{N}$. Also from (B.9) we know that $p_i + (1 - p_i)2^{-(\Delta n_i - 1)} > \eta$ since otherwise Δn_i would be chosen smaller. Therefore,

$$\begin{aligned}
\eta - p_{i+1} &= \eta - p_i - (1 - p_i)2^{-\Delta n_i} \\
&= \eta - \frac{1}{2}p_i - \frac{1}{2}\Big(p_i + (1 - p_i)2^{-(\Delta n_i - 1)}\Big) \\
&\leq \frac{1}{2}(\eta - p_i).
\end{aligned}$$

The second part simply follows by repeatedly applying the above recursion i times and from the fact that $\eta - p_0 = \eta \leq 1$. □

We conclude that the desired accuracy is reached after at most ℓ iterations in which case we stop and set $\Pi_{\eta,\ell} = \Pi_\ell$. It remains to determine how many variables need to be used in total. We first bound how many variables are added in each step.

Lemma B.2 For any $i \in \mathbb{N}$, we have $\Delta n_i < -\log_2(\eta - p_i) + 1$.

Proof. As before we know $p_i + (1 - p_i)2^{-(\Delta n_i - 1)} > \eta$ since otherwise Δn_i would be chosen smaller. This implies

$$2^{-(\Delta n_i - 1)} > \frac{\eta - p_i}{1 - p_i} \geq \eta - p_i,$$

and therefore $\Delta n_i < -\log_2(\eta - p_i) + 1$. □

This can finally be used to bound how many variables are used in total.

Lemma B.3 The total number of variables for $\Pi_{\eta,\ell} = \Pi_\ell$ is

$$n = n_\ell = \sum_{i=1}^{\ell} \Delta n_{i-1} \in \mathcal{O}\Big(\ell^2\Big).$$

Proof. From lemma B.1 we get $\eta - p_i \geq 2(\eta - p_{i+1})$ and thus $\eta - p_i \geq 2^{\ell-1-i}(\eta - p_{\ell-1})$. Without loss of generality we can assume $\eta - p_{\ell-1} \geq 2^{-\ell}$ since otherwise we can stop the iterative construction of $\Pi_{\eta,\ell}$ at $\ell - 1$. Using lemma B.2, this immediately results in

$$\begin{aligned}
n &= \sum_{i=1}^{\ell} \Delta n_{i-1} \\
&\leq \sum_{i=1}^{\ell} -\log_2(\eta - p_{i-1}) + 1 \\
&\leq \sum_{i=1}^{\ell} -\log_2\Big(2^{\ell-i}(\eta - p_{\ell-1})\Big) + 1 \\
&\leq \sum_{i=1}^{\ell} -\log_2\Big(2^{-i}\Big) + 1 \\
&= \frac{\ell(\ell+1)}{2} + \ell \in \mathcal{O}\Big(\ell^2\Big).
\end{aligned}$$

□

B.4 Neutral Operation

We provide a constructive proof of lemma 4.22.

Proof of lemma 4.22: Let $\Psi\colon \{0,1\}^d \to \{0,1\}$ be arbitrary and $0 < \delta < 1$. We will construct a function $\Gamma = \Gamma_{d,\delta}\colon \{0,1\}^r \to \{0,1\}$ so that for some $n_{d,\delta} \in \mathbb{N}$ we have

$$P_{\mathbf{y}}(\Psi(\mathbf{y})) \geq \delta \quad\iff\quad P_{(\mathbf{y},\mathbf{r},\mathbf{t})}\left((\Psi(\mathbf{y}) \wedge \Gamma(\mathbf{r})) \vee \left(\bigwedge_{i=1}^{n} t_i\right)\right) \geq \delta, \tag{B.10}$$

for all $n \geq n_{d,\delta}$ and

$$r + n_{d,\delta} \in \mathcal{O}\left(\log\left(\frac{1}{\delta}\right) + d^2\right).$$

We introduce $\Psi' = \Psi \wedge \Gamma$ and $\Psi'' = \Psi' \vee (\bigwedge_{i=1}^{n} t_i)$. Let us distinguish three cases

$$\begin{aligned}
&\textbf{Case I:} && \delta \leq 2^{-d}, \\
&\textbf{Case II:} && \delta > 2^{-d} \quad\text{and}\quad \left\lceil \delta 2^d \right\rceil - \delta 2^d \geq \frac{2}{3}, \\
&\textbf{Case III:} && \delta > 2^{-d} \quad\text{and}\quad \left\lceil \delta 2^d \right\rceil - \delta 2^d < \frac{2}{3}.
\end{aligned}$$

Let us begin with the construction for the first case. Here, we see that $P(\Psi) < \delta$ is equivalent to $P(\Psi) = 0$. We simply set $\Gamma \equiv 1$ and $n_{d,\delta} = \left\lceil \log\left(\frac{1}{\delta}\right)\right\rceil + 1$. It is easy to check that this satisfies eq. (B.10).

Next, for the second case, we want to construct Γ such that

$$P(\Psi) = \frac{\left\lceil \delta 2^d \right\rceil}{2^d} \quad\implies\quad P(\Psi') \geq \delta, \tag{B.11}$$

$$P(\Psi) = \frac{\left\lceil \delta 2^d \right\rceil - 1}{2^d} \quad\implies\quad P(\Psi') < \delta - \frac{1}{3}2^{-d}, \tag{B.12}$$

which results in the condition

$$a \leq P(\Gamma) < b$$

with

$$\begin{aligned}
a &= \frac{\delta 2^d}{\lceil \delta 2^d \rceil} \\
b &= \frac{\delta 2^d - \frac{1}{3}}{\lceil \delta 2^d \rceil - 1}.
\end{aligned}$$

Thus we can set $\Gamma = \Pi_{\eta,\ell}$ according to appendix B.3 with $\eta = \frac{b+a}{2}$ and $\ell = \lfloor \log\left(\frac{2}{b-a}\right) \rfloor + 1$. Using the fact that $\delta > 2^{-d}$ and hence

$$b - a = \frac{\delta 2^d - \frac{1}{3}\left\lceil \delta 2^d \right\rceil}{\lceil \delta 2^d \rceil \left(\lceil \delta 2^d \rceil - 1\right)} \geq \frac{1}{3}2^{-2d},$$

we obtain $\ell \leq 2d + \lfloor \log(6) \rfloor + 1 = 2d + 3$. In appendix B.3 we showed that $r \in \mathcal{O}(\ell^2)$ and thus $r \in \mathcal{O}(d^2)$. We continue to construct Ψ'' by choosing $n_{d,\delta}$ such that

$$\begin{aligned} P(\Psi') &\geq \delta &&\implies && P(\Psi'') \geq \delta, \\ P(\Psi') &< \delta - \frac{1}{3}2^{-d} &&\implies && P(\Psi'') < \delta, \end{aligned}$$

holds for all $n \geq n_{d,\delta}$. The first condition is automatically fulfilled. From

$$P(\Psi'') = P(\Psi') + (1 - P(\Psi'))2^{-n},$$

as well as $(1 - P(\Psi')) \leq 1$ we observe that

$$2^{-n} < \frac{1}{3}2^{-d}$$

is sufficient for the other condition. Thus, we choose $n_{d,\delta} = d + \lfloor \log(3) \rfloor + 1 = d + 2$.

Finally, for the third case, we again want to construct Γ so that (B.11) and (B.12) hold. Here, this is already satisfied by setting $\Gamma \equiv 1$. We continue analogously as in the second case and choose the same $n_{d,\delta}$.

C

Rate-Distortion Explanation

C.1 Going from Binary to Continuous Distributions

C.1.1 Proof of Lemma 5.3

Lemma. *Let* $\Psi\colon \{0,1\}^d \to \{0,1\}$ *and* $\mathbf{x} \in \{0,1\}^d$. *Then for any* $S \subseteq [d]$ *we have*

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[\Psi(\mathbf{y}) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S] = 1 - 2D^b_{\Psi,\mathbf{x}}(S).$$

Proof. The claim follows directly from the definition of $D^b_{\Psi,\mathbf{x}}$. We have

$$\begin{aligned} D^b_{\Psi,\mathbf{x}}(S) &= \frac{1}{2}\mathbb{E}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}\left[(\Psi(\mathbf{y}) - \Psi(\mathbf{x}))^2 \,\middle|\, \mathbf{y}_S = \mathbf{x}_S\right] \\ &= \frac{1}{2}\mathbb{P}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[\Psi(\mathbf{y}) \neq \Psi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S] \\ &= \frac{1}{2}\Big(1 - \mathbb{P}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[\Psi(\mathbf{y}) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S]\Big) \end{aligned}$$

and thus

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}[\Psi(\mathbf{y}) = \Psi(\mathbf{x}) \,|\, \mathbf{y}_S = \mathbf{x}_S] = 1 - 2D^b_{\Psi,\mathbf{x}}(S). \qquad \square$$

C.1.2 Proof of lemma 5.4

Lemma. *Let* $\Psi\colon \{0,1\}^d \to \{0,1\}$ *and* $\mathbf{x} \in \{0,1\}^d$. *Then for any* $\Phi_0\colon [0,1]^d \to [0,1]$ *interpolating* Ψ, $S \subseteq [d]$, *and* $0 < \eta \leq 1$ *we have for* $\Phi = \Phi_0 \circ \Phi_\eta$ *that*

$$D^b_{\Phi,\mathbf{x}}(S) = D^b_{\Phi_0,\mathbf{x}}(S) = D^b_{\Psi,\mathbf{x}}(S)$$

as well as

$$\left|D^c_{\Phi,\mathbf{x}}(S) - D^b_{\Psi,\mathbf{x}}(S)\right| \leq \frac{d\eta}{2}.$$

Proof. The first part of the claim follows directly from the fact that both Φ_0 and $\Phi = \Phi_0 \circ \Phi_\eta$ interpolate Ψ.

For the second part we consider the event

$$C_S = \left\{ \mathbf{y} \in [0,1]^d \,:\, \exists i \in S^c \text{ such that } y_i \in \left(\frac{1-\eta}{2}, \frac{1+\eta}{2} \right) \right\},$$

and observe that

$$\mathbb{P}_{\mathbf{y}\sim\mathcal{U}([0,1]^d)}[\mathbf{y} \in C_S] = 1 - (1-\eta)^{d-|S|}.$$

Using the abbreviated notation

$$\begin{aligned} A_S &= \mathbb{E}_{\mathbf{y}\sim\mathcal{U}([0,1]^d)}\left[(\Phi(\mathbf{y}) - \Phi(\mathbf{x}))^2 \,\middle|\, \mathbf{y}_S = \mathbf{x}_S, \mathbf{y} \in C_S \right] \\ B_S &= \mathbb{E}_{\mathbf{y}\sim\mathcal{U}([0,1]^d)}\left[(\Phi(\mathbf{y}) - \Phi(\mathbf{x}))^2 \,\middle|\, \mathbf{y}_S = \mathbf{x}_S, \mathbf{y} \notin C_S \right] \end{aligned}$$

we can split the expectation value in the continuous distortion term as

$$\begin{aligned} D^c_{\Phi,\mathbf{x}}(S) &= \frac{1}{2}\mathbb{E}_{\mathbf{y}\sim\mathcal{U}([0,1]^d)}\left[(\Phi(\mathbf{y}) - \Phi(\mathbf{x}))^2 \,\middle|\, \mathbf{y}_S = \mathbf{x}_S \right] \\ &= \frac{1}{2}A_S \mathbb{P}_{\mathbf{y}\sim\mathcal{U}([0,1]^d)}[\mathbf{y} \in C_S] + \frac{1}{2}B_S \mathbb{P}_{\mathbf{y}\sim\mathcal{U}([0,1]^d)}[\mathbf{y} \notin C_S] \\ &= \frac{1}{2}A_S\Big(1 - (1-\eta)^{d-|S|}\Big) + \frac{1}{2}B_S(1-\eta)^{d-|S|}. \end{aligned}$$

For the second term, we get

$$\begin{aligned} B_S &= \mathbb{E}_{\mathbf{y}\sim\mathcal{U}([0,1]^d)}\left[(\Phi_0 \circ \Phi_\eta(\mathbf{y}) - \Phi_0 \circ \Phi_\eta(\mathbf{x}))^2 \,\middle|\, \mathbf{y}_S = \mathbf{x}_S, \mathbf{y} \notin C_S \right] \\ &= \mathbb{E}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}\left[(\Phi_0(\mathbf{y}) - \Phi_0(\mathbf{x}))^2 \,\middle|\, \mathbf{y}_S = \mathbf{x}_S \right] \\ &= \mathbb{E}_{\mathbf{y}\sim\mathcal{U}(\{0,1\}^d)}\left[(\Psi(\mathbf{y}) - \Psi(\mathbf{x}))^2 \,\middle|\, \mathbf{y}_S = \mathbf{x}_S \right] \\ &= 2D^b_{\Psi,\mathbf{x}}(S), \end{aligned}$$

where the second equality follows from the choice of Φ_η and $\mathbf{y} \notin C_S$ and the third equality follows from the fact that Φ_0 interpolates Ψ. Thus, we conclude

$$\begin{aligned} D^c_{\Phi,\mathbf{x}}(S) &= \frac{1}{2}A_S\Big(1 - (1-\eta)^{d-|S|}\Big) + D^b_{\Psi,\mathbf{x}}(S)(1-\eta)^{d-|S|} \\ &= D^b_{\Psi,\mathbf{x}}(S) + \frac{1}{2}\Big(1 - (1-\eta)^{d-|S|}\Big)(A_S - B_S) \end{aligned}$$

which using Bernoulli's inequality and $0 \le A_S, B_S \le 1$ finally results in

$$\Big| D^c_{\Phi,\mathbf{x}}(S) - D^b_{\Psi,\mathbf{x}}(S) \Big| \le \frac{1}{2}\Big(1 - (1-\eta)^{d-|S|}\Big) \le \frac{(d-|S|)\eta}{2} \le \frac{d\eta}{2}. \qquad \square$$

C.2 Choice of the Reference Distribution

C.2.1 Non-Uniform Distributions

We use the uniform distribution $\mathcal{U}\big([0,1]^d\big)$ as a reference or baseline distribution $\mathcal{V}$ in our proof for the hardness result on approximating the rate distortion function. This can easily be extended to more general probability measures μ on $[0,1]^d$. We can choose μ as a product of any independent one-dimensional measures μ_i for the individual input components as long as for every $i \in [d]$ and $0 < \eta \leq 1$ there exist lower and upper thresholds $a_i, b_i \in [0,1]$ with $a_i < b_i$ such that

$$\mu_i((-\infty, a_i]) = \mu_i([b_i, \infty)) \qquad \text{and} \qquad \mu_i([a_i, b_i]) \leq \eta,$$

which is possible for all probability measures on $[0,1]$ without point masses, such as truncated Gaussian or exponential distributions. In that case we simply have to adapt the function Φ_η as $\varphi((\mathbf{x}-\mathbf{a}) \oslash (\mathbf{b}-\mathbf{a}))$ and proceed with the remaining proof as before (here $\oslash$ denotes component-wise division).

C.2.2 Conditional versus Marginal Distributions

We want to make another remark regarding the choice of the distributions $\mathcal{V}_S$ used in our rate distortion framework. We define the obfuscation $\mathbf{y}$ to be deterministically given by $\mathbf{y}_S = \mathbf{x}_S$ on S and distributed according to $\mathbf{y}_{S^c} = \mathbf{n}_{S^c}$ with $\mathbf{n} \sim \mathcal{V}$ on the complement S^c. This means that the resulting distribution $\mathcal{V}_S$ of $\mathbf{y}$ corresponds to $\mathcal{V}$ marginalised over all components in S. One might be tempted to condition on the given components $\mathbf{x}_S$ instead of marginalising. But this could actually be detrimental to uncover how the classifier operates. Let us illustrate this with an example. Consider a classifier that is trained to detect ships, but actually only learned to detect the water surrounding the ship, as in [Lap+16b]. The classifier can achieve high accuracy as long as the data set only contains ships on water and no other objects surrounded by water. Now assume we have a relevance map selecting a subset of pixels showing a ship as relevant. If we complete the rest of the image with random values from a conditional distribution, we will most likely see water in the completion, as most images with a ship will also have water surrounding it. The classifier would correctly classify the completed image with high probability. The potentially small subset of pixels containing the ship will thus give a small distortion and will be considered relevant. However, this result is not useful to uncover the underlying workings of the network. It does not tell us that the network does not recognise ships but only the surrounding water. Using a very data adapted and restricted conditional distribution compensates the shortcoming of the network. That is why we advocate for using a less data adapted marginal distribution. In fact, we believe that using maximally uninformed distributions like uniform or truncated Gaussian distributions is beneficial for uncovering the network's reasoning.

C.3 Variants of the RDE Optimisation Problem

The main idea behind the RDE framework is to utilise the rate-distortion trade-off as a way of characterising the relevance of components of the input signal $\mathbf{x}$ for a classifier prediction $\Phi(\mathbf{x})$. There are three ways to express this trade-off as a concrete optimisation problem. We can use the Lagrangian formulation with a regularisation parameter $\lambda > 0$

$$\begin{aligned} &\text{minimise} && D(\mathbf{s}) + \lambda\|\mathbf{s}\|_1 \\ &\text{subject to} && \mathbf{s} \in [0,1]^d \end{aligned} \tag{L-RDE}$$

as in (5.4) in the main paper or the rate-constrained formulation with a maximal rate $0 \leq k \leq d$

$$\begin{aligned} \text{minimise} \quad & D(\mathbf{s}) \\ \text{subject to} \quad & \|\mathbf{s}\|_1 \leq k, \\ & \mathbf{s} \in [0,1]^d \end{aligned} \tag{RC-RDE}$$

or the distortion-constrained formulation with a maximal distortion $\epsilon \geq 0$

$$\begin{aligned} \text{minimise} \quad & \|\mathbf{s}\|_1 \\ \text{subject to} \quad & D(\mathbf{s}) \leq \epsilon, \\ & \mathbf{s} \in [0,1]^d. \end{aligned} \tag{DC-RDE}$$

The three formulations are of course connected, however the relation between λ, k, and ϵ that would lead to the same solutions is far from trivial. Hence, depending on the goal one of the three formulations is usually preferable: (RC-RDE) if we want to precisely control the allowed rate, (DC-RDE) if we want to precisely control the allowed distortion, or (L-RDE) if we want to find a nice balance between rate and distortion.

Also, one should keep in mind that the three formulation behave quite differently regarding their numerical optimisation. The Lagrangian formulation (L-RDE) results in a non-convex optimisation problem with box constraints and can be solved by any gradient based algorithm capable of handling box constraints, such as projected gradient descent [NW06], or L-BFGS-B [Byr+95]. The rate-constrained formulation results in a non-convex optimisation problem with box constraints as well as an additional linear constraint. Again it can be solved, for exampled, via projected gradient methods. Projections onto the feasible set $\left\{\mathbf{s} \in [0,1]^d : \|\mathbf{s}\|_1 \leq k\right\}$ can be obtained as described in [Duc+08]. Finally, the distortion-constrained formulation yields an optimisation problem with linear objective but non-convex constraint. Such general constraints are typically hard to handle. We tried a primal log-barrier method [NW06] as well as a more involved trust-region SQP interior point method [BHN99], but found that (DC-RDE) behaves numerically rather unstable for high-dimensional problems, such as in the STL-10 experiments. We recommend to use (L-RDE) or (RC-RDE), except for very simple classifiers, such as in the synthetic binary string experiment.

D Invariant Distributions

D.1 Proof of lemma 6.12: Surjectivity of Ξ

For the remainder of the section let assumption 6.7 hold. We will prove lemma 6.12 and show that for any $m \in \mathbb{N}$ there exists a $\delta > 0$ so that the map $\Xi \colon \Omega_{\delta,m} \to \mathbb{R}_+^{m-1} \times \{1\}$ is surjective. We will do this by transforming (through a series of ReLU layers) any measure with infinite support into a measure that is supported on an arbitrary standard m-arc. Such an infinitely supported measure exists within the n-parameter family p by assumption. Due to the ReLU-invariance, the transformed measure is then also included in the family. As a consequence $\Omega_{\delta,m}$ must be non-empty and in fact we show that $\operatorname{arc} \circ p|_{\Omega_{\delta,m}} \colon \Omega_{\delta,m} \to \mathcal{A}_m$ is surjective. The choice of δ will depend on how "evenly spread" the support of the original, untransformed, measure is. Finally, $\mathcal{A}_m$ can be identified with $\mathbb{R}_+^{m-1} \times \{1\}$ via the scaling factors.

As a first step we observe how transforming measures by continuous functions affects their support. The support $\operatorname{supp}(\mu)$ of a Radon measure μ is defined as the complement of the largest open μ-null set. A point $\mathbf{x}$ belongs to $\operatorname{supp}(\mu)$ if and only if every open neighbourhood of $\mathbf{x}$ has positive measure. A Borel set contained in the complement of $\operatorname{supp}(\mu)$ is a μ-null set. The converse holds for open sets, i.e. an open set intersecting $\operatorname{supp}(\mu)$ has positive measure (in fact it suffices if the intersection is relatively open in $\operatorname{supp}(\mu)$).

Lemma D.1 For any $q, r \in \mathbb{N}$ let $\mu \in \mathcal{D}(\mathbb{R}^q)$ and let $f \colon \mathbb{R}^q \to \mathbb{R}^r$ be continuous. Then

$$\operatorname{supp}(f_*(\mu)) = \overline{f(\operatorname{supp}(\mu))},$$

where $f_*(\mu) \in \mathcal{D}(\mathbb{R}^r)$ is the pushforward of μ under f.

Proof. We begin by showing $f(\operatorname{supp}(\mu)) \subseteq \operatorname{supp}(f_*(\mu))$. Let $\mathbf{t} \in f(\operatorname{supp}(\mu))$ and $U \subseteq \mathbb{R}^r$ be any open neighbourhood of $\mathbf{t}$. There exists $\mathbf{x} \in \operatorname{supp}(\mu)$ so that $\mathbf{t} = f(\mathbf{x})$. Since $\mathbf{t} \in U$, we get $\mathbf{x} \in f^{-1}(U)$. Thus $f^{-1}(U)$ is an open neighbourhood of $\mathbf{x}$ and since $\mathbf{x} \in \operatorname{supp}(f)$ we get $f_*(\mu)(U) = \mu(f^{-1}(U)) > 0$. Since U was an arbitrary neighbourhood of $\mathbf{t}$ this shows

$\mathbf{t} \in \operatorname{supp}(f_*(\mu))$. This implies $f(\operatorname{supp}(\mu)) \subseteq \operatorname{supp}(f_*(\mu))$. But $\operatorname{supp}(f_*(\mu))$ is closed, so we can even conclude $\overline{f(\operatorname{supp}(\mu))} \subseteq \operatorname{supp}(f_*(\mu))$.

We will show the converse inclusion by showing $\overline{f(\operatorname{supp}(\mu))}^c \subseteq \operatorname{supp}(f_*(\mu))^c$. For this let $\mathbf{t} \in \overline{f(\operatorname{supp}(\mu))}^c$. Then there exists an open neighbourhood U of $\mathbf{t}$ such that $U \cap \overline{f(\operatorname{supp}(\mu))} = \emptyset$, which implies $f^{-1}(U) \cap \operatorname{supp}(\mu) = \emptyset$. But this is only possible if $f_*(\mu)(U) = \mu(f^{-1}(U)) = 0$ and therefore $\mathbf{t} \in \operatorname{supp}(f_*(\mu))^c$. □

To simplify notations in the following, we begin the transformation by establishing a standardised situation that takes places exclusively in the two-dimensional subspace $\operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\}$ of $\mathbb{R}^d$. In fact, all m-arcs are essentially one-dimensional objects embedded in a two-dimensional space. Therefore, we start by transforming the infinitely supported measure to the non-negative part of the one-dimensional space $\operatorname{span}\{\mathbf{e}_1\}$, which will then subsequently be “bent” into the correct arc living in $\operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\}$.

D.1.1 Projection to a One-Dimensional Subspace

Lemma D.2 Let $\mu \in \mathcal{D}(\mathbb{R}^d)$ such that $\operatorname{supp}(\mu)$ is not finite. Then there exists $\mathbf{W} \in \mathbb{R}^{d\times d}$ and $\mathbf{b} \in \mathbb{R}^d$ such that the image of $f\colon \mathbb{R}^d \to \mathbb{R}^d\colon \mathbf{x} \mapsto \varrho(\mathbf{W}\mathbf{x}+\mathbf{b})$ is contained in $\operatorname{span}\{\mathbf{e}_1\}$ and $\operatorname{supp}(f_*(\mu))$ is infinite.

Proof. Consider the coordinate projections $\operatorname{proj}_i\colon \mathbb{R}^d \to \mathbb{R}\colon \mathbf{x} \mapsto \mathbf{e}_i^\top \mathbf{x}$. Clearly we have

$$\operatorname{supp}(\mu) \subseteq \operatorname{proj}_1(\operatorname{supp}(\mu)) \times \cdots \times \operatorname{proj}_d(\operatorname{supp}(\mu)).$$

By assumption $\operatorname{supp}(\mu)$ is infinite and the product can only be infinite if at least one of its factors is infinite, for example $\operatorname{proj}_j(\operatorname{supp}(\mu))$. If $\operatorname{proj}_j(\operatorname{supp}(\mu)) \cap \mathbb{R}_+$ is infinite, we set $\sigma = +1$, otherwise $\operatorname{proj}_j(\operatorname{supp}(\mu)) \cap \mathbb{R}_-$ must be infinite and we set $\sigma = -1$. Now choosing

$$\mathbf{W} = \begin{bmatrix} \sigma\mathbf{e}_j & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}^\top \quad \text{and} \quad \mathbf{b} = \mathbf{0},$$

clearly yields $f(\mathbb{R}^d) \subseteq \operatorname{span}\{\mathbf{e}_1\}$. Further, by lemma D.1 we have

$$\begin{aligned}
\operatorname{supp}(f_*(\mu)) &= \overline{f(\operatorname{supp}(\mu))} \\
&= \overline{\varrho\Big(\sigma \operatorname{proj}_j(\operatorname{supp}(\mu))\Big)} \times \{0\} \times \cdots \times \{0\} \\
&= \begin{cases} \overline{\operatorname{proj}_j(\operatorname{supp}(\mu)) \cap \mathbb{R}_+} \times \{0\} \times \cdots \times \{0\}, & \text{if } \sigma = +1 \\ -\overline{\operatorname{proj}_j(\operatorname{supp}(\mu)) \cap \mathbb{R}_+} \times \{0\} \times \cdots \times \{0\}, & \text{if } \sigma = -1 \end{cases},
\end{aligned}$$

which by the choice of σ is infinite. □

Thus, for the next step we only need to consider measures with infinite support within the non-negative part of $\operatorname{span}\{\mathbf{e}_1\}$. Two dimensions will be enough for our construction to “bend” the measures into arc shape. Any two-dimensional ReLU layer $\mathbb{R}^2 \to \mathbb{R}^2\colon \mathbf{x} \mapsto \varrho(\mathbf{W}\mathbf{x} + \mathbf{b})$ can easily be extended to a d-dimensional ReLU layer

$$\mathbb{R}^d \to \mathbb{R}^d \colon \mathbf{x} \mapsto \varrho\left(\begin{bmatrix} \mathbf{W} & \mathbf{0}_{2\times(d-2)} \\ \mathbf{0}_{(d-2)\times 2} & \mathbf{0}_{(d-2)\times(d-2)} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{b} \\ \mathbf{0}_{(d-2)\times 1} \end{bmatrix}\right),$$

only operating on $\operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\} \subseteq \mathbb{R}^d$ by appropriately padding with zeros. To simplify the notation, we leave out these zero paddings and just assume without loss of generality $d = 2$ from now on.

D.1.2 Partitioning into Line Segments

Now we show how the infinite support on the non-negative real line can be partitioned into an arbitrary number of intervals, which in the end will correspond to the arc line segments. We can write $\operatorname{supp}(\mu) = S \times \{0\} \subseteq \mathbb{R}^2$ for some set $S \subseteq \mathbb{R}_{\geq 0}$.

Lemma D.3 Let $S \subseteq \mathbb{R}_{\geq 0}$ be infinite. Then for any $m \in \mathbb{N}$ there exist $m + 1$ distinct points $0 = b_0 < \cdots < b_m < \infty$ such that $S \cap (b_{j-1}, b_j) \neq \emptyset$ for all $j = 1, \ldots, m$.

Proof. Since S is infinite there exist m distinct points $s_1, \ldots, s_m \in S$ with $0 < s_1 < \cdots < s_m$ for any $m \in \mathbb{N}$. Setting $b_0 = 0$,

$$b_j = \frac{s_j + s_{j+1}}{2} \quad \text{for} \quad j = 1, \ldots, m - 1,$$

and $b_m = s_m + 1$ we get $s_j \in (b_{j-1}, b_j)$ for $j = 1, \ldots, m$. □

Further, we can restrict the support of the measure to a compact subset by clipping the set S to the range $[b_0, b_m] = [0, b_m]$. This can be achieved using the two layer ReLU network

$$\mathbb{R}^2 \to \mathbb{R}^2 \colon \mathbf{x} \mapsto \varrho\left(-\varrho\left(-\mathbf{x} + \begin{bmatrix} b_m \\ 0 \end{bmatrix}\right) + \begin{bmatrix} b_m \\ 0 \end{bmatrix}\right)$$

with b_m as in lemma D.3. Altogether, without loss of generality we can from now on assume that μ is supported on $[b_0, b_m] \times \{0\} \subseteq \mathbb{R}^2$ and the support intersects each subset $(b_{j-1}, b_j) \times \{0\}$ for $j = 1, \ldots, m$.

D.1.3 Resizing the Line Segments

After partitioning the support of a distribution into intervals that each contain a certain amount of the probability mass we now want to bend it into arc shape. Each interval in the partition will correspond to a line segment. However, the bending will distort the segment lengths, so in order to obtain an arc with specified segment lengths we first have to resize the intervals. This is also possible with ReLU layers.

Lemma D.4 For $m \geq 2$ let $0 = a_0 < a_1 < \cdots < a_m < \infty$ and $0 = b_0 < b_1 < \cdots < b_m < \infty$. Then there exists a collection of ReLU layer transformations $\{F_j : \mathbb{R}^2 \to \mathbb{R}^2\}_{j=0,\dots,m}$ such that their composition $F = F_m \circ F_{m-1} \circ \cdots \circ F_1 \circ F_0$ satisfies

$$F\left(\begin{bmatrix} b_i \\ 0 \end{bmatrix}\right) = \begin{bmatrix} a_i \\ 0 \end{bmatrix} \quad \text{for all} \quad i = 0, \dots, m,$$

and restricted to $\mathbb{R}_{\geq 0} \times \{0\}$ it is a piecewise linear map with breakpoints (possibly) at $b_0, \dots, b_m$.

Proof. The proof consists of two parts. First, we iteratively construct a collection of piecewise linear and strictly increasing functions $\{f_j : \mathbb{R} \to \mathbb{R}\}_{j=0,\dots,m}$ such that

$$(f_j \circ \cdots \circ f_0)(b_i) = a_i \quad \text{for all} \quad i = 0, \dots, j, \tag{D.1}$$

for any $j = 0, \dots, m$. In particular for $j = m$ and $f = f_m \circ f_{m-1} \circ \cdots \circ f_0$ we obtain $f(b_i) = a_i$ for all $i = 0, \dots, m$. Afterwards we show how to construct $\{F_j\}_j$ from $\{f_j\}_j$.

Since $b_0 = a_0 = 0$, we can start with

$$f_0(x) = x$$

and see that (D.1) is satisfied for $j = 0$. Now, assuming we have already constructed strictly increasing and piecewise linear functions $f_0, \dots, f_j$ satisfying (D.1), we now want to construct f_{j+1}. The idea is to choose it in such a way that it leaves the points already correctly mapped by $f_j \circ \cdots \circ f_0$ unchanged but linearly transforms the remaining points so that b_{j+1} is mapped to a_{j+1}. For brevity we denote $b_i^j = (f_j \circ \cdots \circ f_0)(b_i)$ for $i = 0, \dots, m$. By assumption $b_i^j = a_i$ for $i = 0, \dots, j$ and by strict monotonicity $b_{j+1}^j > b_j^j = a_j$. We set

$$f_{j+1}(x) = \begin{cases} x, & x \leq a_j, \\ x + \frac{a_{j+1} - b_{j+1}^j}{b_{j+1}^j - a_j}(x - a_j), & x > a_j. \end{cases}$$

It is easy to see that this function leaves all points $a_1 < a_2 \cdots < a_j$ unchanged, maps b_{j+1}^j to a_{j+1} and is continuous, monotonously increasing and piecewise linear. We can also express f_{j+1} using ϱ, i.e.

$$f_{j+1}(x) = x + \frac{a_{j+1} - b_{j+1}^j}{b_{j+1}^j - a_j} \varrho(x - a_j). \tag{D.2}$$

Then f_{j+1} maps b_{j+1}^j to a_{j+1} and acts as the identity map on the region $x \leq a_j$ where points are already correctly mapped, as illustrated in D.1. Hence clearly (D.1) is also satisfied for $j + 1$. To see that f_{j+1} is strictly increasing we observe that its slope is 1 on the region $x \leq a_j$ and

$$1 + \frac{a_{j+1} - b_{j+1}^j}{b_{j+1}^j - a_j} = \frac{a_{j+1} - a_j}{b_{j+1}^j - a_j} > 0$$

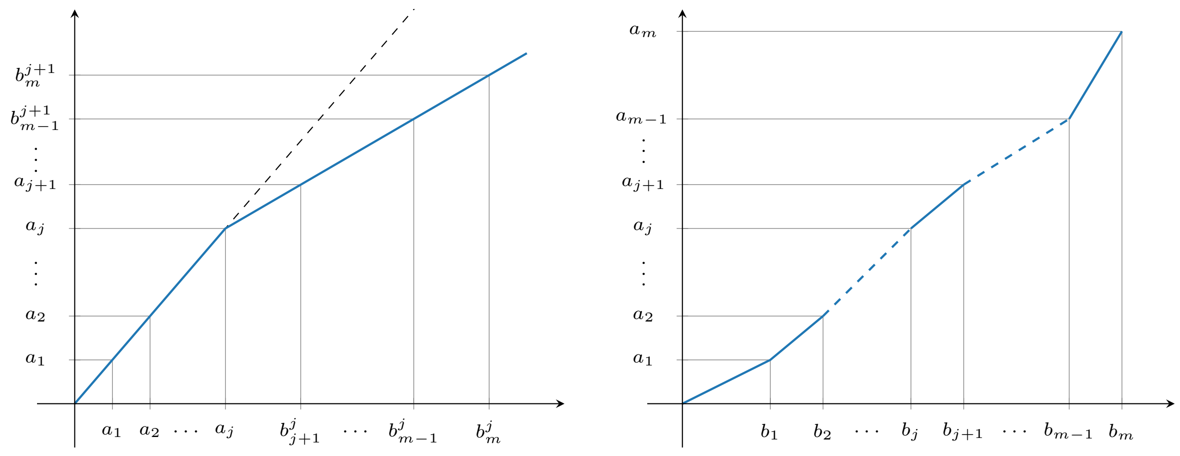


Figure D.1: The function $f_{j+1}(x)$ transforms the points $0 = a_0 = b_0^j, a_1 = b_1^j, \dots, a_j = b_j^j$ and $b_{j+1}^j, \dots, b_m^j$ (left). The region $x \le a_j$ is left unchanged and the region $x \ge a_j$ is linearly rescaled so that $f_{j+1}(b_{j+1}^j) = a_{j+1}$. The piecewise-linear function $f = f_m \circ \cdots \circ f_0$ maps b_j to a_j (right).

on the region $x \ge a_j$. This adds at most one new breakpoint at b_j to the composition $f_{j+1} \circ \cdots \circ f_0$. Continuing this process until $j = m$ finishes the first part.

Next we derive the ReLU layers $\{F_j\}_j$ from the functions $\{f_j\}_j$. For $j = 0, \dots, m-1$ we denote

$$w_{j+1} = \frac{a_{j+1} - b_{j+1}^j}{b_{j+1}^j - a_j}$$

and rewrite (D.2) as

$$f_{j+1}(x) = \begin{bmatrix} 1 & w_{j+1} \end{bmatrix} \varrho\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ -a_j \end{bmatrix} \right) \quad \text{for} \quad x \ge 0.$$

Two consecutive maps can be combined as

$$f_{j+1}(f_j(x)) = \begin{bmatrix} 1 & w_{j+1} \end{bmatrix} \varrho\left(\begin{bmatrix} 1 & w_j \\ 1 & w_j \end{bmatrix} \varrho\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ -a_{j-1} \end{bmatrix} \right) + \begin{bmatrix} 0 \\ -a_j \end{bmatrix} \right)$$

for $x \ge 0$, hence we set

$$F_j \colon \mathbb{R}^2 \to \mathbb{R}^2 \colon \mathbf{x} \mapsto \varrho\left(\begin{bmatrix} 1 & w_j \\ 1 & w_j \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ -a_j \end{bmatrix} \right)$$

for $j = 1, ..., m-1$. It is now easy to check that with

$$F_0(\mathbf{x}) = \varrho\left(\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \mathbf{x} \right)$$

and

$$F_m(\mathbf{x}) \mapsto \varrho\left(\begin{bmatrix} 1 & w_m \\ 0 & 0 \end{bmatrix} \mathbf{x} \right)$$

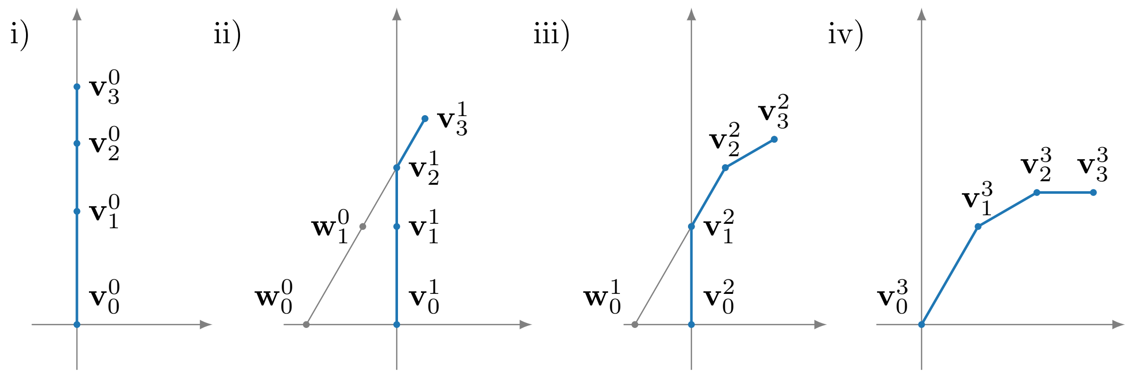


Figure D.2: Illustration of the 'bending' of a 3-arc in 4 steps. Step (i) shows G_0 (90° rotation) applied to $[a_m, a_0] \times \{0\}$. Steps (ii) and (iii) show the image of $G_1 \circ G_0$ and $G_2 \circ G_1 \circ G_0$ respectively (blue) as well as the corresponding transform just before the last ReLU activation (gray). Step (iv) shows the final arc arising as the image of $G = G_3 \circ G_2 \circ G_1 \circ G_0$.

we finally get

$$F\left(\begin{bmatrix} x \\ 0 \end{bmatrix}\right) = \begin{bmatrix} f(x) \\ 0 \end{bmatrix} \quad \text{for} \quad x \geq 0$$

for $F = F_m \circ F_{m-1} \circ \cdots \circ F_1 \circ F_0$. □

D.1.4 Bending the Arc

Let us next consider how to "bend" a one dimensional subspace into an m-arc. This iterative procedure is illustrated in fig. D.2. It starts with all vertices in a straight line on the x_2-axis. First the polygonal chain is rotated by $-\phi_m$ ("to the right") and translated in negative x_1 direction ("to the left") such that the second to last vertex lies on the x_2-axis. Then a ReLU is applied, which projects all vertices up until the second to last onto the x_2-axis. The last last arc segment now has the correct angle and the process is repeated until all segments are correctly "bent". Note that each projection shrinks the unbent line segments by a factor of $\cos(\phi_m)$. To describe this transformation formally, we denote rotation matrices in $\mathbb{R}^2$ by

$$R_\alpha = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}.$$

Furthermore, for $m \geq 2$ and $0 = a_0 < a_1 < \cdots < a_m < \infty$ we define

$$\mathbf{d}_j = \begin{bmatrix} \sin(\phi_m)\cos(\phi_m)^j a_{m-j} \\ 0 \end{bmatrix}, \quad \text{for} \quad j = 1, \ldots, m,$$

as well as

$$G_0 \colon \mathbb{R}^2 \to \mathbb{R}^2 \colon \mathbf{x} \mapsto \varrho\Big(R_{\frac{\pi}{2}}\mathbf{x}\Big)$$

and

$$G_j \colon \mathbb{R}^2 \to \mathbb{R}^2 \colon \mathbf{x} \mapsto \varrho(R_{-\phi_m}\mathbf{x} - \mathbf{d}_j) \quad \text{for} \quad j = 1, \ldots, m,$$

which clearly are ReLU transformations.

Lemma D.5 Let $m \geq 2$ and $0 = a_0 < a_1 < \cdots < a_m < \infty$ and G_j as above. Then

$$G\colon [a_0, a_m] \times \{0\} \subseteq \mathbb{R}^2 \to \mathbb{R}^2 \colon \mathbf{x} \mapsto (G_m \circ G_{m-1} \circ \cdots \circ G_1 \circ G_0)(\mathbf{x})$$

parametrises an m-arc with vertices $\mathbf{v}_i = G\left(\begin{bmatrix} a_i \\ 0 \end{bmatrix}\right)$, line segments $\ell_i = G([a_{i-1}, a_i] \times \{0\})$, and scaling factors $r_i = \cos^{m-i}(\phi_m)(a_i - a_{i-1})$ for $i = 1, \dots, m$.

Proof. We need to show that G is (in its first component) a piecewise-linear function with breakpoints a_i, and that $\mathbf{v}_0 = \mathbf{0}$ as well as

$$\mathbf{v}_i = \mathbf{v}_{i-1} + \cos^{m-i}(\phi_m)(a_i - a_{i-1}) \begin{bmatrix} \sin(i\phi_m) \\ \cos(i\phi_m) \end{bmatrix}, \quad \text{for} \quad i = 1, \dots, m. \tag{D.3}$$

For this we denote

$$\mathbf{v}_i^j = (G_j \circ \cdots \circ G_0)\left(\begin{bmatrix} a_i \\ 0 \end{bmatrix}\right) \quad \text{for} \quad i, j = 0, ..., m.$$

By the choice of ϕ_m we know $\sin(\phi_m) > 0$ and $\cos(\phi_m) > 0$. Hence, the first component of the bias vectors $\mathbf{d}_j$ satisfy $\sin(\phi_m)\cos(\phi_m)^j a_{m-j} \geq 0$ for $j = 1, \dots, m$. Therefore, $\mathbf{v}_0^j = \mathbf{0}$ for all j and in particular $\mathbf{v}_0 = \mathbf{v}_0^m = \mathbf{0}$. Further, we will show inductively over j that for any i, j we have

$$\mathbf{v}_i^j - \mathbf{v}_{i-1}^j = \begin{cases} \cos(\phi_m)^j (a_i - a_{i-1}) \begin{bmatrix} 0 \\ 1 \end{bmatrix}, & i \leq m - j, \\ \cos(\phi_m)^{m-i}(a_i - a_{i-1}) \begin{bmatrix} \sin((i+j-m)\phi_m) \\ \cos((i+j-m)\phi_m) \end{bmatrix}, & i > m - j. \end{cases} \tag{D.4}$$

Then (D.3) follows from (D.4) with $j = m$.

To start the inductive proof we observe that for $j = 0$ and $i = 1, \dots, m$ we have

$$\begin{aligned} \mathbf{v}_i^0 - \mathbf{v}_{i-1}^0 &= G_0\left(\begin{bmatrix} a_i \\ 0 \end{bmatrix}\right) - G_0\left(\begin{bmatrix} a_{i-1} \\ 0 \end{bmatrix}\right) \\ &= R_{\frac{\pi}{2}} \begin{bmatrix} a_i \\ 0 \end{bmatrix} - R_{\frac{\pi}{2}} \begin{bmatrix} a_{i-1} \\ 0 \end{bmatrix} \\ &= (a_i - a_{i-1}) \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \end{aligned}$$

satisfying (D.4).

Next, let us assume that (D.4) holds for some $j < m$. For any i we denote $\mathbf{w}_i^j = R_{-\phi_m}\mathbf{v}_i^j - \mathbf{d}_j$ and therefore $\mathbf{v}_i^{j+1} = G_{j+1}(\mathbf{v}_i^j) = \varrho(\mathbf{w}_i^j)$ and

$$\mathbf{w}_i^j - \mathbf{w}_{i-1}^j = \begin{cases} \cos(\phi_m)^j(a_i - a_{i-1})\begin{bmatrix}\sin(\phi_m)\\ \cos(\phi_m)\end{bmatrix}, & i \le m-j, \\ \cos(\phi_m)^{m-i}(a_i - a_{i-1})\begin{bmatrix}\sin((i+j+1-m)\phi_m)\\ \cos((i+j+1-m)\phi_m)\end{bmatrix}, & i > m-j. \end{cases}$$

We also have $\sin((i+j+1-m)\phi_m) > 0$ and $\cos((i+j+1-m)\phi_m) \ge 0$ for $i > m-j$ by the choice of ϕ_m and therefore $\mathbf{w}_m^j \ge \mathbf{w}_{m-1}^j \ge \cdots \ge \mathbf{w}_1^j \ge \mathbf{w}_0^j = -\mathbf{d}_j$ component-wise, where even all inequalities are strict in the first component. We immediatley see that all $\mathbf{w}_i^j$ lie in the upper half-plane. Further,

$$\begin{aligned}\mathbf{w}_{m-j}^j &= \mathbf{w}_0^j + \sum_{i=1}^{m-j}(\mathbf{w}_i^j - \mathbf{w}_{i-1}^j) \\ &= -\mathbf{d}_j + \sum_{i=1}^{m-j}\cos(\phi_m)^j(a_i - a_{i-1})\begin{bmatrix}\sin(\phi_m)\\ \cos(\phi_m)\end{bmatrix} \\ &= -\begin{bmatrix}\sin(\phi_m)\cos(\phi_m)^j a_{m-j}\\ 0\end{bmatrix} + \cos(\phi_m)^j a_{m-j}\begin{bmatrix}\sin(\phi_m)\\ \cos(\phi_m)\end{bmatrix} \\ &= \cos(\phi_m)^{j+1}a_{m-j}\begin{bmatrix}0\\ 1\end{bmatrix},\end{aligned}$$

from which we can conclude that

$$(\mathbf{w}_i^j)_1 \ge 0, \quad \text{for} \quad i = 1, \dots, m \tag{D.5}$$

$$(\mathbf{w}_i^j)_2 < 0, \quad \text{for} \quad i = 1, ..., m-j-1, \tag{D.6}$$

$$(\mathbf{w}_i^j)_2 \ge 0, \quad \text{for} \quad i = m-j, ..., m. \tag{D.7}$$

Now $\mathbf{v}_i^{j+1} = \varrho(\mathbf{w}_i^j)$ together with (D.5) to (D.7) implies

$$\begin{aligned}(\mathbf{v}_i^{j+1})_1 &= (\mathbf{w}_i^j)_1, && \text{for} \quad i = 1, ..., m, \\ (\mathbf{v}_i^{j+1})_2 &= 0, && \text{for} \quad i = 1, ..., m-j-1, \\ (\mathbf{v}_i^{j+1})_2 &= (\mathbf{w}_i^j)_2, && \text{for} \quad i = m-j, ..., m,\end{aligned}$$

and therefore

$$\mathbf{v}_i^{j+1} - \mathbf{v}_{i-1}^{j+1} = \begin{cases} \cos(\phi_m)^{j+1}(a_i - a_{i-1})\begin{bmatrix}0\\ 1\end{bmatrix}, & i \le m-(j+1), \\ \cos(\phi_m)^{m-i}(a_i - a_{i-1})\begin{bmatrix}\sin((i+j+1-m)\phi_m)\\ \cos((i+j+1-m)\phi_m)\end{bmatrix}, & i > m-(j+1), \end{cases}$$

which shows that (D.4) holds for $j+1$ and concludes the inductive step.

Each G_{j+1} adds only one new breakpoint at to the overall function, corresponding to point $\mathbf{w}^j_{m-j}$ where the second component of $(G_j \circ \cdots \circ G_0)(\mathbf{x})$ switches sign. This corresponds to the breakpoint a_{m-j} of G. $\square$

We are now ready to give the main proof of this section.

Proof of lemma 6.12. Let $m \in \mathbb{N}$ be arbitrary. By assumption there exists a measure $\mu \in \{p(\theta)\}_{\theta\in\Omega}$ with infinite support. As discussed in section D.1.1 and in particular by using lemma D.2 we can without loss of generality assume $d = 2$ and $\operatorname{supp}(\mu)$ is compact and contained in the non-negative part of $\operatorname{span}\{\mathbf{e}_1\}$. In other words we can rewrite $\operatorname{supp}(\mu) = S \times \{0\} \subseteq [0, b] \times \{0\}$ for an infinite set $S \subseteq \mathbb{R}_{\geq 0}$ and some $b > 0$. By lemma D.3 we can choose points $0 = b_0 < b_1 < \cdots < b_m = b$ so that S intersects each (b_{j-1}, b_j). By the choice of S and since $(b_{j-1}, b_j) \times \{0\}$ is relatively open in $S \times \{0\}$ we have $\mu((b_{j-1}, b_j) \times \{0\}) > 0$ for all j. Set $0 < \delta \leq \min_j \mu((b_{j-1}, b_j) \times \{0\})$.

Now let $A \in \mathcal{A}_m$ be an arbitrary standard m-arc and denote its set of vertices by $(\mathbf{v}_0, \ldots, \mathbf{v}_m) = \operatorname{vert}(A)$, its line segments by $(\ell_1, \ldots, \ell_m) = \operatorname{segm}(A)$, and its scaling factors by $(r_1, \ldots, r_m) = \operatorname{scal}(A)$. We set $a_0 = 0$ and iteratively $a_j = a_{j-1} + r_j \cos^{j-m}(\phi_m)$ for $j = 1, \ldots, m$. By lemma D.4 there is a ReLU neural network F satisfying

$$F\left(\begin{bmatrix} b_j \\ 0 \end{bmatrix}\right) = \begin{bmatrix} a_j \\ 0 \end{bmatrix} \quad \text{for} \quad j = 1, \ldots, m,$$

and then by lemma D.5 another ReLU neural network G satisfying

$$G\left(\begin{bmatrix} a_i \\ 0 \end{bmatrix}\right) = \mathbf{v}_j \quad \text{for} \quad j = 1, \ldots, m.$$

Altogether $G \circ F$ is a ReLU neural network transforming $\mathbf{b}_j = \begin{bmatrix} b_i & 0 \end{bmatrix}^\top$ to the vertex $\mathbf{v}_j$ and $[b_{j-1}, b_j] \times \{0\}$ to the line segment ℓ_j. By lemma D.1 we have

$$\operatorname{supp}((G \circ F)_*\mu) = \overline{(G \circ F)(\operatorname{supp}(\mu))} \subseteq \overline{(G \circ F)([b_0, b_m] \times \{0\})} \subseteq A.$$

Also, for each j we get

$$((G \circ F)_*\mu)(\ell_j) = \mu\Big((G \circ F)^{-1}(\ell_j)\Big) \geq \mu((b_{j-1}, b_j) \times \{0\}) \geq \delta$$

and therefore $(G \circ F)_*\mu \in \mathcal{D}_{\delta,m}(\mathbb{R}^2)$ and $\operatorname{arc}((G \circ F)_*\mu) = A$. Using the ReLU-invariance, we know that $(G \circ F)_*\mu \in \{p(\theta)\}_{\theta\in\Omega_{\delta,m}} \subseteq \{p(\theta)\}_{\theta\in\Omega}$. Altogether, since $A \in \mathcal{A}_m$ was arbitrary we conclude that $\operatorname{arc} \circ p|_{\Omega_{\delta,m}}$ is surjective onto $\mathcal{A}_m$. Clearly, $\operatorname{scal}\colon \mathcal{A}_m \to \mathbb{R}^{m-1}_+ \times \{1\}$ is a bijection, hence also $\Xi\colon \Omega_{\delta,m} \to \mathbb{R}^{m-1}_+ \times \{1\}$ is surjective. $\square$

D.2 Proof of lemma 6.13: Local Lipschitz Continuity of Ξ

We will now show that the map $\Xi\colon \Omega_{\delta,m} \to \mathbb{R}^m_+$ is locally Lipschitz continuous by showing this for each of its three composite parts $p|_{\Omega_{\delta,m}}\colon \Omega_{\delta,m} \to \mathcal{D}_{\delta,m}(\mathbb{R}^d)$, $\operatorname{arc}\colon \mathcal{D}_{\delta,m}(\mathbb{R}^d) \to \mathcal{A}_m$, and $\operatorname{scal}\colon \mathcal{A}_m \to \mathbb{R}^m_+$.

Firstly, $p|_{\Omega_{\delta,m}} : \Omega_{\delta,m} \to \mathcal{D}_{\delta,m}(\mathbb{R}^d)$ is locally Lipschitz continuous by assumption 6.7, so there is nothing to show.

Secondly, the local Lipschitz continuity of $\operatorname{arc}: \mathcal{D}_{\delta,m}(\mathbb{R}^d) \to \mathcal{A}_m$ can be derived from geometric observations.

Let us recall the definition of the Prokhorov metric [Pro56]:

$$\begin{aligned} d_P(\mu,\nu) = \inf\{\epsilon > 0 :& \mu(B) \leq \nu(B^\epsilon) + \epsilon \text{ and} \\ & \nu(B) \leq \mu(B^\epsilon) + \epsilon \text{ for any } B \in \mathcal{B}(\mathbb{R}^d)\}, \end{aligned}$$

where $\mathcal{B}(\mathbb{R}^d)$ denotes the Borel σ-Algebra on $\mathbb{R}^d$ and $B^\epsilon = \{\, x \in \mathbb{R}^d : \exists y \in B \text{ with } \|x-y\| < \epsilon \,\}$ is the open ϵ-neighbourhood of a Borel set $B \in \mathcal{B}(\mathbb{R}^d)$.[1]

Lemma D.6 The map $\operatorname{arc}: (\mathcal{D}_{\delta,m}(\mathbb{R}^d), d_P) \to (\mathcal{A}_m, d_{\mathcal{A}}): \mu \mapsto A_\mu$, is locally Lipschitz continuous with Lipschitz constant $\frac{2\sqrt{2}}{\sin(\phi_m)}$.

Proof. Let $\mu_0 \in \mathcal{D}_{\delta,m}(\mathbb{R}^d)$ be arbitrary. We will show Lipschitz continuity on the open ball of radius $\frac{\delta}{2}$ around μ_0. For this let $\mu_1, \mu_2 \in B_{\frac{\delta}{2}}(\mu_0) \cap \mathcal{D}_{\delta,m}(\mathbb{R}^d)$ and therefore $d_P(\mu_1,\mu_2) \leq \delta$. We know that μ_1 and μ_2 are δ-supported on two unique standard m-arcs $A_1 = \operatorname{arc}(\mu_1) = A_{\mu_1} \in \mathcal{A}_m$ and $A_2 = \operatorname{arc}(\mu_2) = A_{\mu_2} \in \mathcal{A}_m$ and by definition these lie in the same two-dimensional subspace $\operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\}$ of $\mathbb{R}^d$. So without loss of generality we carry out the remaining proof in $\mathbb{R}^2$. Let $(\ell_1^1, \dots, \ell_m^1) = \operatorname{segm}(A_1)$ and $(\ell_1^2, \dots, \ell_m^2) = \operatorname{segm}(A_2)$ denote the line segments of A_1 and A_2 respectively. We denote the affine hulls of individual line segments as $h_i^j = \operatorname{aff}\left(\ell_i^j\right)$ and further denote the Euclidean distance of two corresponding affine hulls by $d_i = \operatorname{dist}(h_i^1, h_i^2)$. An example for this can be seen in fig. D.3.

We want to upper bound the Euclidean distances $\|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2$ of all corresponding vertices of the two arcs. For any i one of four cases can occur, as visualised in fig. D.4. Firstly, if $\mathbf{v}_i^1 = \mathbf{v}_i^2$ (cf. Figure D.4, top left) we trivially get $\|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2 = 0$. Secondly, if $\mathbf{v}_i^1 \neq \mathbf{v}_i^2$ but $h_i^1 = h_i^2$ (cf. Figure D.4, top right) it is not hard to see that

$$\|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2 \leq \frac{d_{i+1}}{\sin(\phi_m)}.$$

Thirdly, if $\mathbf{v}_i^1 \neq \mathbf{v}_i^2$ but $h_{i+1}^1 = h_{i+1}^2$ (cf. Figure D.4, bottom left) we similarly get

$$\|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2 \leq \frac{d_i}{\sin(\phi_m)}.$$

In the fourth case, where $\mathbf{v}_i^1 \neq \mathbf{v}_i^2$, $h_i^1 \neq h_i^2$, and $h_{i+1}^1 \neq h_{i+1}^2$ the four intersections of the affine spaces form a parallelogram (cf. Figure D.4, bottom right). We obtain

$$\|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2 \leq \frac{\sqrt{2}(d_i + d_{i+1})}{\sin(\phi_m)},$$

using the parallelogram identity. Thus, it remains to bound all the distances d_i.

[1]The original definition by Prokhorov only considers closed sets B, however this results in the same metric.

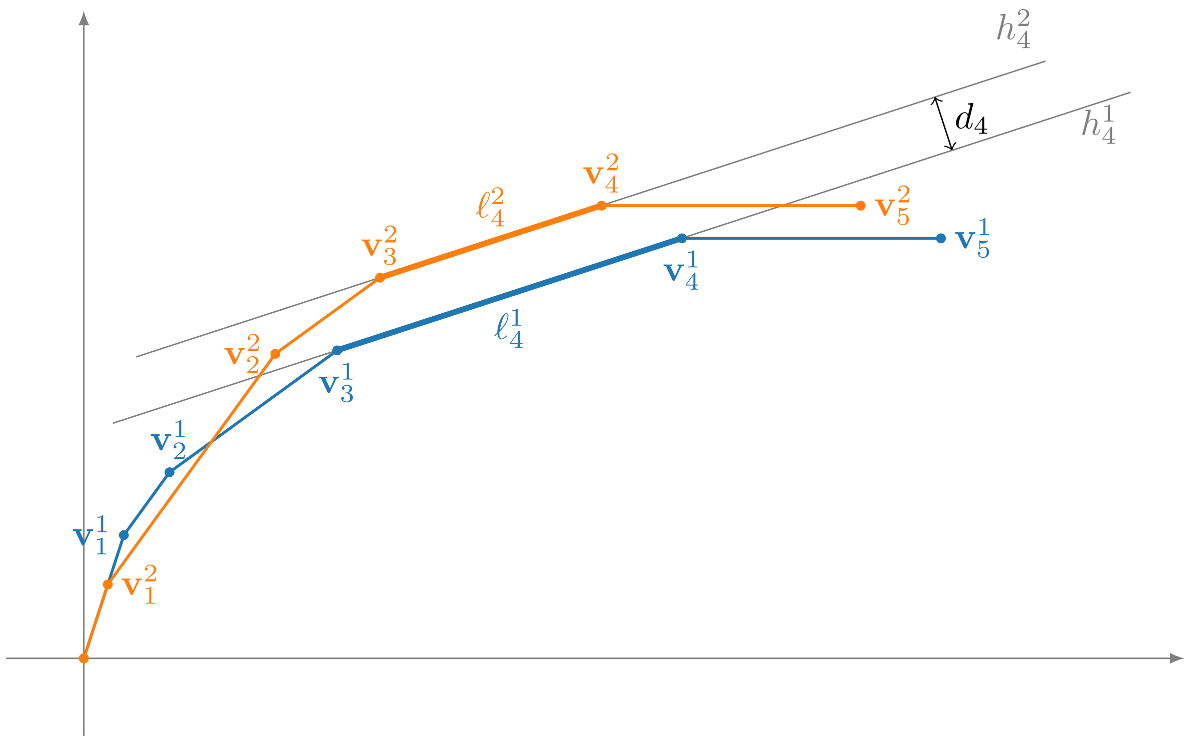


Figure D.3: Example of two standard m-arcs A_1 (blue) and A_2 (orange) for $m = 5$. Supporting affine subspaces h_4^1 and h_4^2 and their distance d_4 are shown exemplarily for the fourth line segments. At this segment A_2 is the outer arc and A_1 is the inner arc.

For this, assume that for some i we have $d_i > 0$. We say that ℓ_i^1 is the *outer* line segment and ℓ_i^2 is the *inner* line segment if $\operatorname{dist}(\mathbf{0}, h_i^1) > \operatorname{dist}(\mathbf{0}, h_i^2)$ and vice versa if $\operatorname{dist}(\mathbf{0}, h_i^1) < \operatorname{dist}(\mathbf{0}, h_i^2)$. Without loss of generality assume that ℓ_i^1 is the outer line segment. Then for any point $\mathbf{p} \in A_2$ we have $\operatorname{dist}(\mathbf{p}, \ell_i^1) \geq d_i$. Thus $\mu_2((\ell_i^1)^\epsilon) = 0$ for any $\epsilon < d_i$. But $\mu_1(\ell_i^1) \geq \delta$ and therefore by definition of the Prokhorov metric

$$d_i \leq d_P(\mu_1, \mu_2).$$

Altogether, we obtain

$$d_{\mathcal{A}}(A_1, A_2) = \max_i \|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2 \leq \frac{2\sqrt{2}}{\sin(\phi_m)} d_P(\mu_1, \mu_2). \qquad \square$$

Thirdly, the local Lipschitz continuity of $\operatorname{scal}\colon \mathcal{A}_m \to \mathbb{R}_+^m$ is a straight-forward calculation if $\mathbb{R}_+^m$ is equipped with the metric induced by the ℓ_∞-norm. By norm equivalence the same also holds for $\mathbb{R}_+^m$ viewed as a subspace of Euclidean $\mathbb{R}^m$.

Lemma D.7 The map $\operatorname{scal}\colon (\mathcal{A}_m, d_{\mathcal{A}}) \to (\mathbb{R}_+^m, \|\cdot\|_\infty)\colon A \mapsto (r_1, \dots, r_m)$, is Lipschitz continuous with Lipschitz constant 2.

Proof. Let $A_1, A_2 \in \mathcal{A}_m$ be arcs with vertices $(\mathbf{v}_1^1, \dots, \mathbf{v}_m^1) = \operatorname{vert}(A_1)$ and $(\mathbf{v}_1^2, \dots, \mathbf{v}_m^2) = \operatorname{vert}(A_2)$ and scaling factors $(r_1^1, \dots, r_m^1) = \operatorname{scal}(A_1)$ as well as $(r_1^2, \dots, r_m^2) = \operatorname{scal}(A_2)$

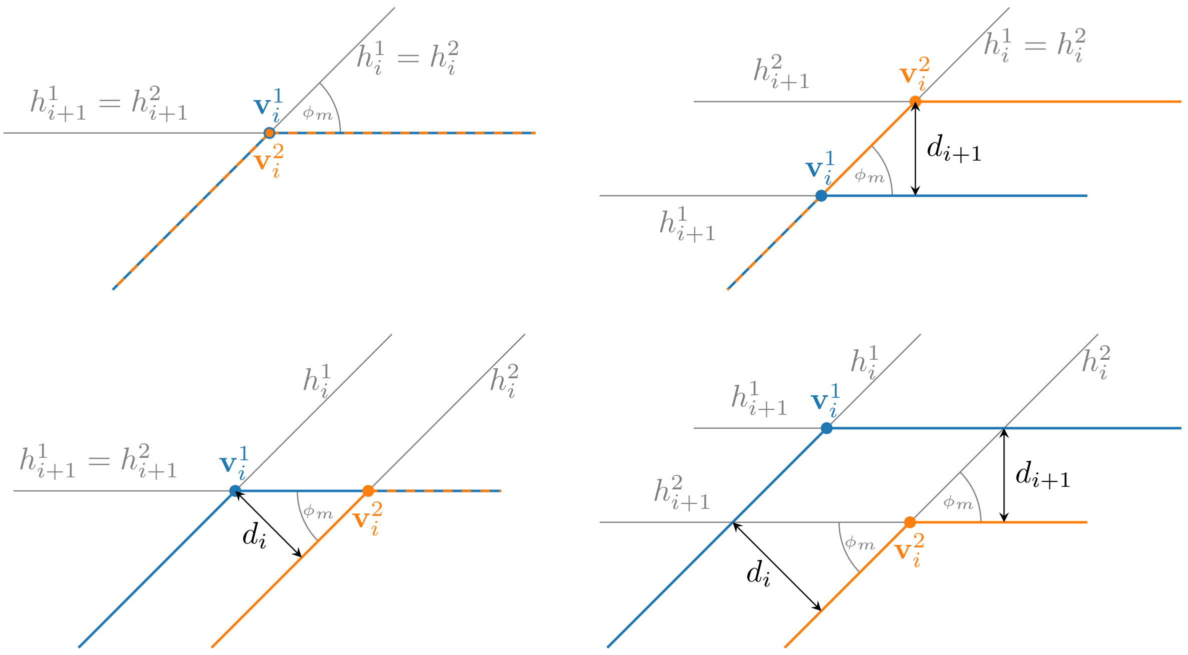


Figure D.4: Four possible arrangements of corresponding vertices $\mathbf{v}_i^1$ and $\mathbf{v}_i^2$ of two arcs. In case one both vertices are equal (top left). In cases two and three the vertices are different but either the affine subspaces h_i^1 and h_i^2 or h_{i+1}^1 and h_{i+1}^2 coincide (top right and bottom left). In the fourth case all affine subspaces differ and their intersections form a parallelogram with $\mathbf{v}_i^1$ and $\mathbf{v}_i^2$ at opposite corners (bottom right).

respectively. Since for all $i = 1, \dots, m$ we have

$$r_i^1 \leq r_i^2 + \operatorname{dist}\left(\mathbf{v}_{i-1}^1, \mathbf{v}_{i-1}^2\right) + \operatorname{dist}\left(\mathbf{v}_i^1, \mathbf{v}_i^2\right)$$

and vice versa

$$r_i^2 \leq r_i^1 + \operatorname{dist}\left(\mathbf{v}_{i-1}^1, \mathbf{v}_{i-1}^2\right) + \operatorname{dist}\left(\mathbf{v}_i^1, \mathbf{v}_i^2\right),$$

we obtain

$$\|\operatorname{scal}(A_1) - \operatorname{scal}(A_2)\|_\infty = \max_{i=1,\dots,m} \left| r_i^1 - r_i^2 \right| \leq 2 d_{\mathcal{A}}(A_1, A_2). \qquad \square$$

Altogether we can conclude this section by combining all pieces and proving the local Lipschitz continuity of $\Xi\colon \Omega_{\delta,m} \to \mathbb{R}_+^m$.

Proof of lemma 6.13. The claim follows directly from assumption 6.7 and lemmas D.6 and D.7 since the composition of locally Lipschitz continuous functions is again locally Lipschitz continuous. $\square$

D.3 Families of Distributions in One Dimension

In this section we will give a prove of the first part of theorem 6.6 corresponding to (R1). For this we explicitly construct a family of ReLU-invariant probability distributions on $\mathbb{R}$ with a locally Lipschitz continuous parametrisation map.

We proceed in three steps. We first analyse the set of one-dimensional ReLU networks and show that it can be completely described by only six parameters. We then use these to parametrise all ReLU neural networks as functions in $C(K, \mathbb{R})$ on some compact domain

$K \subseteq \mathbb{R}$ via a so called *realisation* map $R\colon \mathbb{R}^6 \to C(K, \mathbb{R})$. Finally, we use a pushforward map $Q\colon C(K, \mathbb{R}) \to \mathcal{D}(\mathbb{R})$ mapping neural network functions $f \in C(K, \mathbb{R})$ to the pushforward of a fixed *prototype* measure $\mu_0 \in \mathcal{D}(\mathbb{R})$ under f to generate the family of probability distribution. The prototype measure is assumed to be supported within K. The one-parameter family $p\colon \mathbb{R}^6 \to \mathcal{D}(\mathbb{R})$ is then simply given as $p = Q \circ R$.

We will follow a similar idea to obtain ReLU-invariant families in higher dimension when proving the third part of theorem 6.6 corresponding to (R3) in appendix D.4. However in this case the number of parameters to describe the neural networks grows with their depth leading to the need for an arbitrary large number of parameters to describe the distributions in the family. Although it is possible to generalise our construction to higher dimensions this comes at the cost of losing the local Lipschitz continuity of the parametrisation map.

One-dimensional ReLU networks: Let us first show that any one dimensional ReLU network can be rewritten as a three layer network. For this let $L \in \mathbb{N}$ and $f\colon \mathbb{R} \to \mathbb{R}$ be an L-layer ReLU network given as

$$f = f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1$$

with layers

$$f_i\colon \mathbb{R} \to \mathbb{R}\colon x \mapsto \varrho(w_i x + b_i) \quad \text{for} \quad i = 1, \dots, L.$$

We observe that f is constant as soon as any weight w_i is zero and a constant function can easily be rewritten as a three layer network. So from now on we assume $w_i \neq 0$ for all layers. We denote by

$$A_i = \{\, x \in \mathbb{R} \,:\, w_i x + b_i \geq 0 \,\} \quad \text{for} \quad i = 1, \dots, L$$

the domain on which the i-th layer is affine linear with slope w_i. It is constant on the complement A_i^c. Further we denote

$$B_i = (f_{i-1} \circ \cdots \circ f_1)^{-1}(A_i) \quad \text{for} \quad i = 2, \dots, L$$

and $B_1 = A_1$. Each A_i is a convex and closed subset of $\mathbb{R}$ and $f_{i-1} \circ \cdots \circ f_1$ is continuous and monotone as a composition of continuous and monotone functions. Hence, also all B_i are closed and convex. In $\mathbb{R}$ these are exactly the (possibly unbounded) closed intervals.

Lemma D.8 Let $U \subseteq \mathbb{R} \setminus \bigcap_{i=1}^{L} B_i$ be connected. Then $f|_U$ is constant.

Proof. We have $\mathbb{R} \setminus \bigcap_{i=1}^{L} B_i = \bigcup_{i=1}^{L} B_i^c$ and therefore $U = \bigcup_{i=1}^{L} U \cap B_i^c$. Since B_i is closed B_i^c is open and thus $U \cap B_i^c$ is relatively open in U. But by the choice of B_i we know $(f_i \circ \cdots \circ f_1)(U \cap B_i^c) \equiv 0$, hence $f|_{U \cap B_i^c}$ is constant. But since f is continuous and U is connected this already implies that $f|_U$ is constant. □

The one-dimensional ReLU network f is piecewise constant except for the (possibly empty) region $\bigcap_{i=1}^{L} B_i$ where all layers are non constant and act affine linearly. On this region f is affine linear with slope $\prod_{i=1}^{L} w_i$. As discussed above each B_i is convex and closed, hence the same holds for their intersection. Two different cases can occur, which determine the overall shape of the function f. This is also visualized in fig. 6.4 of the main paper.

Firstly, if $\bigcap_{i=1}^{L} B_i$ is bounded it must be empty, a single point, or a bounded and closed interval. Then f is bounded. In fact it is constant or a piecewise linear step function with one step and two constant regions. These functions can be represented by three layer networks.

Secondly, if $\bigcap_{i=1}^{L} B_i$ is an unbounded closed interval, then f looks like a shifted and scaled ReLU function and can again be represented by a three layer ReLU network.

Lemma D.9 The ReLU network f has one of the following forms.

- f is bounded and has two constant and one affine linear region:

$$f(x) = \begin{cases} c_1, & x \le a_1 \\ c_2, & x \ge a_2 \\ c_1 + \frac{c_2 - c_1}{a_2 - a_1}(x - a_1), & a_1 < x < a_2 \end{cases}$$

  for some $c_1, c_2 \ge 0$ and $a_1 < a_2$. This includes the degenerate case where f is constant by choosing $c_1 = c_2$.

- f is unbounded and has one constant and one affine linear region:

$$f(x) = \begin{cases} c, & w(x-a) \le 0 \\ c + w(x-a), & w(x-a) > 0 \end{cases}$$

  for some $c \ge 0$, $a \in \mathbb{R}$, and $w \ne 0$.

Proof. This directly follows from lemma D.8, the continuity and monotonicity of the ReLU network f and the fact that f is lower bounded by zero. □

Lemma D.10 The two types of piecewise linear functions from lemma D.9 can both be written as one dimensional ReLU networks with three layers.

Proof. We give explicit formulas for choosing the weights w_1, w_2, w_3 and biases b_1, b_2, b_3 of the three layer network.

In the first case of a bounded function we choose

$$\begin{aligned} w_1 &= 1, & b_1 &= -a_1, \\ w_2 &= -\frac{|c_2 - c_1|}{a_2 - a_1}, & b_2 &= |c_2 - c_1|, \\ w_3 &= -\operatorname{sign}(c_2 - c_1), & b_3 &= c_2, \end{aligned}$$

and in the second case of an unbounded function we choose

$$\begin{aligned} w_1 &= \operatorname{sign}(w), & b_1 &= -\operatorname{sign}(w)a, \\ w_2 &= |w|, & b_2 &= c, \\ w_3 &= 1, & b_3 &= 0. \end{aligned}$$

The claim now follows by straightforward calculations. □

Neural Network Parametrisation: Neural networks can be viewed algebraically (as collections of weights and biases) or analytically (as functions in some function space). This distinction was made in [PV18] and used in [PRV18] to study the dependence of neural network functions on their weights and biases. The mapping from weights and biases to the corresponding neural network function is referred to as the *realisation* map. In [PRV18] the continuity of the realisation map was shown for neural networks of fixed size with continuous activation function (in particular this includes the ReLU activation), when they are viewed as functions in $C(K, \mathbb{R}^d)$ on a compact domain K. If the activation function is Lipschitz continuous (which again is the case for the ReLU activation) also the realisation map is Lipschitz continuous.

As we have seen all ReLU neural networks in one dimension can be rewritten into a three layer network and can thus be parametrised by three weight and three bias parameters. Let $K \subseteq \mathbb{R}$ be any compact non-empty domain. For any collection of weights and biases $\theta = (w_1, b_1, w_2, b_2, w_3, b_3) \in \mathbb{R}^6$ we denote the function that is the ReLU neural network realisation of these weights and biases as $f_\theta \colon K \to \mathbb{R}$. Then the realisation map is

$$R \colon \mathbb{R}^6 \to C(K, \mathbb{R}^d) \colon \theta \mapsto f_\theta.$$

From [PRV18, Proposition 5.1] we obtain the following result.

Lemma D.11 The realisation map $R \colon \mathbb{R}^6 \to C(K, \mathbb{R}^d)$ is Lipschitz continuous.

Pushforward Mapping: We want to connect neural network functions with the effect they have on probability measures and obtain an invariant family of distributions. For this let $\mu_0 \in \mathcal{D}(\mathbb{R})$ be any probability measure on $\mathbb{R}$ with $\operatorname{supp}(\mu_0) \subseteq K$. We call this the *prototype* measure and will derive all other measures in the family as pushforwards of μ_0 under ReLU neural networks. We define the pushforward function

$$Q \colon C(K, \mathbb{R}) \to \mathcal{D}(\mathbb{R}) \colon f \mapsto f_* \mu_0.$$

The restriction of Q to the subset of ReLU neural network functions will be used to generate the invariant family of distributions.

Lemma D.12 The pushforward function $Q \colon C(K, \mathbb{R}) \to \mathcal{D}(\mathbb{R})$ is Lipschitz continuous.

Proof. Let $f, g \in C(K, \mathbb{R})$ and $A \in \mathcal{B}(\mathbb{R})$. Assuming $f^{-1}(A) \neq \emptyset$, for any $x \in f^{-1}(A)$ we have $f(x) \in A$ and $|f(x) - g(x)| \leq \|f - g\|_\infty$. Consequently, for any $\epsilon > \|f - g\|_\infty$ we get $g(x) \in A^\epsilon$ and therefore $x \in g^{-1}(A^\epsilon)$. Thus $f^{-1}(A) \subseteq g^{-1}(A^\epsilon)$. Conversely, if $f^{-1}(A) = \emptyset$ then the same inclusion trivially holds. Analogously $g^{-1}(A) \subseteq f^{-1}(A^\epsilon)$ can be shown for any $\epsilon > \|f - g\|_\infty$. This directly implies

$$\begin{aligned}
d_P(Q(f), Q(g)) &= d_P(f_*\mu_0, g_*\mu_0) \\
&= \inf \{\, \epsilon > 0 \,:\, f_*\mu_0(A) \leq g_*\mu_0(A^\epsilon) + \epsilon \text{ and} \\
&\qquad\quad g_*\mu_0(A) \leq f_*\mu_0(A^\epsilon) + \epsilon \text{ for any } A \in \mathcal{B}(\mathbb{R}) \,\} \\
&= \inf \Big\{ \epsilon > 0 \,:\, \mu_0(f^{-1}(A)) \leq \mu_0(g^{-1}(A^\epsilon)) + \epsilon \text{ and} \\
&\qquad\quad \mu_0(g^{-1}(A)) \leq \mu_0(f^{-1}(A^\epsilon)) + \epsilon \text{ for any } A \in \mathcal{B}(\mathbb{R}) \Big\} \\
&\leq \|f - g\|_\infty.
\end{aligned}$$

□

ReLU-Invariance: We can now define the family of distributions as

$$p \colon \mathbb{R}^6 \to \mathcal{D}(\mathbb{R}) \colon \theta \mapsto (Q \circ R)(\theta).$$

The local Lipschitz continuity of p is clear from the previous steps. It remains to establish the ReLU-invariance.

Lemma D.13 The family of distributions p is ReLU-invariant.

Proof. Let $\theta \in \mathbb{R}^6$ be arbitrary and $p(\theta) \in \mathcal{D}(\mathbb{R})$ the corresponding probability measure. Further, let $f \colon \mathbb{R} \to \mathbb{R} \colon x \mapsto \varrho(wx + b)$ for some $w, b \in \mathbb{R}$ be any ReLU layer. We need to show that there exists a $\omega \in \mathbb{R}^6$ such that $p(\omega) = f_*p(\theta)$.

By construction $p(\theta) = (Q \circ R)(\theta) = (R(\theta))_*\mu_0 = (f_\theta)_*\mu_0$, where $f_\theta = R(\theta)$ is a ReLU neural network. Clearly also $f \circ f_\theta$ is a ReLU neural network, and since any ReLU network can be rewritten as a three-layer network, there exists $\omega \in \mathbb{R}^6$ such that $R(\omega) = f \circ f_\theta$. Finally,

$$p(\omega) = (f \circ f_\theta)_*\mu_0 = f_*(f_\theta)_*\mu_0 = f_*p(\theta)$$

yields the claim. □

D.4 Families of Distributions Without Local Lipschitz Continuity

In this section we will prove the third part of theorem 6.6 corresponding to (R3).

We define a family of distributions each of which can be described by a finite but arbitrarily large number of parameters. We use space-filling curves to fuse these arbitrarily many parameters into a single parameter.

The high-level proof idea is quite straightforward and analogous to appendix D.3: If we have an arbitrarily large number of parameters available to describe each distribution in the family we can simply use these parameters to specify the weights of the neural network transformations with respect to which we want to obtain invariance. The challenge in the general d-dimensional setting is to show that this leads to a continuous parametrisation.

We start by clarifying what we mean by a finite but arbitrarily large number of parameters. We use the conventional notation $\mathbb{R}^\omega$ for the product of countably many copies of $\mathbb{R}$ equipped with the product topology. In other words, $\mathbb{R}^\omega$ is the space of all real-valued sequences. Further we denote by

$$\mathbb{R}^\infty = \{ (x_1, x_2, \dots) \in \mathbb{R}^\omega \ : \ x_i \neq 0 \text{ for only finitely many } i \}$$

the subset of sequences that are eventually zero. The space $\mathbb{R}^\infty$ will serve as the parameter space. Each element in $\mathbb{R}^\infty$ effectively uses only a finite number of parameters (since all the remaining ones are zero), however this number can be arbitrarily large.

We proceed in three steps. We first introduce a one-parameter description $\Gamma\colon \mathbb{R} \to \mathbb{R}^\infty$ of the finitely-but-arbitrarily-many parameter set $\mathbb{R}^\infty$. We then use a subset $\Omega_\infty \subseteq \mathbb{R}^\infty$ to parametrise ReLU neural networks as functions in $C(K, \mathbb{R}^d)$ using a realisation map $R\colon \Omega_\infty \to C(K, \mathbb{R}^d)$. Finally, we use a pushforward map $Q\colon C(K, \mathbb{R}^d) \to \mathcal{D}(\mathbb{R}^d)$ mapping neural network functions $f \in C(K, \mathbb{R}^d)$ to the pushforward of a fixed *prototype* measure $\mu_0 \in \mathcal{D}(\mathbb{R}^d)$ under f to generate the family of probability distribution. As before the prototype measure is assumed to be supported within the domain K. The one-parameter family $p\colon \Omega \to \mathcal{D}(\mathbb{R}^d)$ is then given as $p = Q \circ R \circ \Gamma$ for some suitable domain Ω. We will now discuss each of the steps in more detail.

Space-Filling Curves: We want to continuously describe $\mathbb{R}^\infty$ with a single parameter. This can be achieved by gluing together continuous space-filling curves from the unit interval to cubes of arbitrary dimension. The existence of such maps is guaranteed by the Hahn-Mazurkiewicz theorem, see for example [Sag94, Chapter 6.8].

Lemma D.14 There exists a continuous and surjective function $\Gamma\colon \mathbb{R} \to \mathbb{R}^\infty$.

The construction of Γ is deferred to appendix D.6.

Neural Network Parametrisation: We need to extend the realisation map from appendix D.3 to arbitrary dimensions and arbitrary numbers of layers. Further we will show that the results in [PRV18] concerning continuous network parametrisations can be extended to networks of arbitrary depth.

A ReLU neural network in d dimensions is characterised by the number of its layers as well as $d^2 + d$ parameters for the weights and biases for each of the layers. Using the first component in $\mathbb{R}^\infty$ to encode the number of layers we can parametrise all such neural networks with the set

$$\Omega_\infty = \left\{ (L, x_1, x_2, \dots) \in \mathbb{R}^\infty \ : \ L \in \mathbb{N} \text{ and } x_i = 0 \text{ for all } i > L\Big(d^2 + d\Big) \right\} \subseteq \mathbb{R}^\infty.$$

It will be convenient to introduce notations for the subsets of parameters with a fixed number of layers. For $L \in \mathbb{N}$ we write $\Omega_L = \{L\} \times \mathbb{R}^{L(d^2+d)} \times \{0\}^\infty \subseteq \Omega_\infty$ and observe that

$$\Omega_\infty = \bigcup_{L \in \mathbb{N}} \Omega_L.$$

In fact this is the partition of Ω_∞ into its connected components.

For any $\theta = (L, x_1, x_2, \dots) \in \Omega_\infty$ we can regroup its leading non-zero components into alternating blocks of size d^2 and d and write

$$\theta = (L, \mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \dots, \mathbf{W}_L, \mathbf{b}_L, 0, \dots)$$

with $\mathbf{W}_i \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_i \in \mathbb{R}^d$. As before let $K \subseteq \mathbb{R}^d$ be any non-empty compact domain and denote the function that is the ReLU neural network realisation of the weights $\{\mathbf{W}_1, \dots, \mathbf{W}_d\}$ and biases $\{\mathbf{b}_1, \dots, \mathbf{b}_d\}$ as $f_\theta \colon K \to \mathbb{R}^d$. Then we can define the extended realisation map

$$R \colon \Omega_\infty \to C(K, \mathbb{R}^d) \colon \theta \mapsto f_\theta.$$

Lemma D.15 The extended realisation map $R \colon \Omega_\infty \to C(K, \mathbb{R}^d)$ is continuous.

Proof. To show the continuity of $R \colon \Omega_\infty \to C(K, \mathbb{R}^d)$ it suffices to show the continuity on all connected components Ω_L of the domain. But for each fixed $L \in \mathbb{N}$ we know from [PRV18, Proposition 5.1] that the realisation map is continuous from Ω_L to $C(K, \mathbb{R}^d)$. This already proves the overall continuity of R. □

Pushforward Mapping: We need to extend our pushforward map from appendix D.3 to arbitrary dimension. As before let $\mu_0 \in \mathcal{D}(\mathbb{R}^d)$ be any probability measure on $\mathbb{R}^d$ with $\operatorname{supp}(\mu_0) \subseteq K$. We define the pushforward function analogous to before as

$$Q \colon C(K, \mathbb{R}^d) \to \mathcal{D}(\mathbb{R}^d) \colon f \mapsto f_* \mu_0.$$

The restriction of Q to the subset of ReLU neural network functions will be used to generate the invariant family of distributions.

Lemma D.16 The pushforward function $Q \colon C(K, \mathbb{R}^d) \to \mathcal{D}(\mathbb{R}^d)$ is continuous.

The proof works exactly as in the one-dimensional case in appendix D.3.

ReLU-Invariance: We can now define the one-parameter family of distributions as

$$p \colon \Omega \to \mathcal{D}(\mathbb{R}^d) \colon \theta \mapsto (Q \circ R \circ \Gamma)(\theta)$$

with the domain $\Omega = \Gamma^{-1}(\Omega_\infty)$ chosen so that Γ maps it exactly to the feasible neural network parameters Ω_∞ and thus $R \circ \Gamma$ maps it to all ReLU neural networks in d dimensions. The continuity of p is clear from the three previous steps. It remains to establish the ReLU-invariance.

Lemma D.17 The one-parameter family of distributions p is ReLU-invariant.

Proof. Let $\theta \in \Omega$ be arbitrary and $p(\theta) \in \mathcal{D}(\mathbb{R}^d)$ the corresponding probability measure. Further, let $f\colon \mathbb{R}^d \to \mathbb{R}^d\colon \mathbf{x} \mapsto \varrho(\mathbf{W}\mathbf{x} + \mathbf{b})$ for some $\mathbf{W} \in \mathbb{R}^{d\times d}$ and $\mathbf{b} \in \mathbb{R}^d$ be any ReLU layer. We need to show that there exists a $\omega \in \Omega$ such that $p(\omega) = f_* p(\theta)$.

By construction we have $p(\theta) = (Q \circ R \circ \Gamma)(\theta) = (R(\Gamma(\theta)))_* \mu_0 = \widetilde{f}_* \mu_0$, where $\widetilde{f} = R(\Gamma(\theta))$ is a ReLU neural network. Clearly also $f \circ \widetilde{f}$ is a ReLU neural network, so there exists $\omega \in \Omega$ such that $R(\Gamma(\omega)) = f \circ \widetilde{f}$. Finally,

$$p(\omega) = (f \circ \widetilde{f})_* \mu_0 = f_* \widetilde{f}_* \mu_0 = f_* p(\theta)$$

yields the claim. □

D.5 Metric Spaces and Hausdorff Dimension

In this section we will proof lemma 6.14. For this we will first review some concepts and results regarding general metric spaces and their Hausdorff dimension, which we denote by $\dim_H$. Applying these to Euclidean $\mathbb{R}^n$ or subspaces thereof will yield the desired result.

A topological space is called a Lindelöf space if every open cover of it has a countable subcover. This is weaker than compactness, where the existence of finite subcovers is required. For metric spaces the notions Lindelöf, separable, and second-countable are all equivalent. It is easy to see that any σ-compact space is Lindelöf. Subspaces of separable metric spaces are again separable. We start with a collection of useful properties of the Hausdorff dimension, see for example [Edg08, Chapter 6].

Lemma D.18 Let (X, d_X) and (Y, d_Y) be metric spaces, $A, B \subseteq X$ Borel sets, $(A_i)_{i\in\mathbb{N}}$ a countable collection of Borel sets $A_i \subseteq X$, and $f\colon X \to Y$ (globally) Lipschitz continuous. Then

(i) $A \subseteq B \Rightarrow \dim_H(A) \leq \dim_H(B)$,

(ii) $\dim_H(A \cup B) = \max\{\dim_H(A), \dim_H(B)\}$,

(iii) $\dim_H(\bigcup_{i\in\mathbb{N}} A_i) = \sup_{i\in\mathbb{N}}\{\dim_H(A_i)\}$,

(iv) $\dim_H(f(X)) \leq \dim_H(X)$.

If (X, d_X) is a separable space, then the last part can be generalised to locally Lipschitz continuous maps.

Lemma D.19 Let (X, d_X) and (Y, d_Y) be metric spaces, $f\colon X \to Y$ locally Lipschitz continuous, and X separable. Then

$$\dim_H(f(X)) \leq \dim_H(X).$$

Proof. For every $x \in X$ there exists an open neighbourhood U_x of x such that f restricted to U_x is Lipschitz continuous. Also $(U_x)_{x\in X}$ is an open cover of X. As X is separable and thus Lindelöf there exists a countable subcover $(U_{x_i})_{i\in\mathbb{N}}$. From lemma D.18 we conclude

$$\dim_H(f(U_{x_i})) \leq \dim_H(U_{x_i}) \leq \dim_H(X)$$

for all $i \in \mathbb{N}$. Since $f(X) = f(\bigcup_{i\in\mathbb{N}} U_{x_i}) = \bigcup_{i\in\mathbb{N}} f(U_{x_i})$, we can again use lemma D.18 to conclude

$$\dim_H(f(X)) = \dim_H(\bigcup_{i\in\mathbb{N}} f(U_{x_i})) \leq \sup_{i\in\mathbb{N}} \dim_H(f(U_{x_i})) \leq \dim_H(X). \qquad \square$$

We now come to the special case of the Euclidean space $\mathbb{R}^n$. Every Borel subset of $\mathbb{R}^n$ with non-empty interior has Hausdorff dimension n, again see for example [Edg08, Chapter 6]. A direct consequence of this is the following result.

Lemma D.20 We have $\dim_H(\mathbb{R}^n) = n$ and $\dim_H(\mathbb{R}^n_+) = n$.

We are now ready to prove lemma 6.14.

Proof of lemma 6.14. Since $B \subseteq \mathbb{R}^n$ is a separable metric space, f is locally Lipschitz continuous, and $f(B) \subseteq \mathbb{R}^m$ has non-empty interior, we can use lemmas D.19 and D.20 to obtain

$$m = \dim_H(f(B)) \leq \dim_H(B) \leq \dim_H(\mathbb{R}^n) = n. \qquad \square$$

D.6 Space-Filling Curves in Arbitrary Dimensions

In this section we describe the construction of the continuous and surjective function $\Gamma\colon \mathbb{R} \to \mathbb{R}^\infty$ and thus prove lemma D.14.

We start with constructing a map from the unit interval to the n-dimensional cube for any $n \in \mathbb{N}$. We can then glue these maps together to obtain Γ.

Let $g\colon [0,1] \to [0,1]^2$ be any continuous and surjective space-filling curve. Examples for such curves are the Sierpiński curve, the Hilbert curve, or the Peano curve, see [Sag94] for an overview of various space-filling curves. We extend this to higher-dimensional cubes by iteratively defining curves $g_n\colon [0,1] \to [0,1]^n$ for $n \geq 2$ as

$$\begin{aligned} g_2 &= g \\ g_n &= (\mathrm{id}_{n-2} \times g) \circ g_{n-1}, \qquad n \geq 3, \end{aligned}$$

which are again continuous and surjective. From these we can obtain continuous and surjective curves $h_n\colon [0,1] \to [-n,n]^n$ from the unit interval to the scaled symmetric n-dimensional cubes by scaling and translating. Since we ultimately want glue all these h_n together we also want to assure $h_n(0) = h_n(1) = \mathbf{0}$ for all $n \geq 2$. Thus we define

$$h_n(t) = \begin{cases} 4nt(2g_n(0) - \mathbf{1}), & t \in \left[0, \frac{1}{4}\right], \\ n\Big(2g_n\Big(2t - \frac{1}{2}\Big) - \mathbf{1}\Big), & t \in \left[\frac{1}{4}, \frac{3}{4}\right], \\ 4n(1-t)(2g_n(1) - \mathbf{1}), & t \in \left[\frac{3}{4}, 1\right]. \end{cases}$$

Finally, we glue the pieces together to get a continuous map from one real parameter to any arbitrary finite number of parameters. We define the map $\Gamma\colon \mathbb{R} \to \mathbb{R}^\omega$ that maps the interval $[n, n+1]$ surjectively to the $[-n, n]^n$ cube on the first n coordinates of $\mathbb{R}^\omega$, that is

$$\Gamma(t) = \begin{cases} h_{\lfloor t \rfloor}(t - \lfloor t \rfloor) \times \{0\}^\infty, & t \geq 2, \\ \{0\}^\infty \quad , & t < 2. \end{cases}$$

The only critical points regarding the continuity of Γ are the integers where we transition from one interval to the next and thus $\lfloor t \rfloor$ changes. But by assuring $h_n(0) = h_n(1) = \mathbf{0}$ for all $n \geq 2$ we achieve a continuous gluing at the interval transitions. Thus Γ is continuous restricted to each of the intervals $(-\infty, 2], [2, 3], [3, 4], \ldots$ respectively. These form a locally finite cover of $\mathbb{R}$ by closed sets, hence we can use the pasting Lemma, see for example [Dug66, Chapter III.9], to conclude the continuity of Γ on all of $\mathbb{R}$. The map is not surjective onto $\mathbb{R}^\omega$. However for our purpose we only require surjectivity onto $\mathbb{R}^\infty$ and this is clearly satisfied, since

$$\mathbb{R}^\infty = \bigcup_{n \geq 2} [-n, n]^n \times \{0\}^\infty = \bigcup_{n \geq 2} \Gamma([n, n+1]).$$

D.7 Restricting the Set of Weight Matrices

In this section we briefly discuss some variations of characterisations of $\mathcal{F}$-invariant families of distributions, where $\mathcal{F}$ is not the entire set of all ReLU layers. If the considered collection $\mathcal{F}$ of functions is much more restricted, it might be possible to obtain invariant families of distributions.

D.7.1 General Restrictions of the Weight Matrices

The setting we considered so far has no restrictions on the weight matrices. We make use of that in our constructive proof by repeatedly relying on rotation and projection matrices. Putting restrictions on the matrices, such as non-negativity, symmetry, positive-definiteness, or allowing only diagonal matrices, would prohibit our proof strategy. Using only diagonal matrices renders the functions in $\mathcal{F}$ separable in their input components and thus effectively reduces to the one-dimensional case. For this we have already discussed the invariant distributions in section 6.4.1. For all other matrix restrictions it remains to be investigated whether our approach can be adapted.

D.7.2 Restrictions on the Number of Weight Matrices and Bias Vectors

Another practical concern can be the number of different possible weight matrices. So far we considered a continuum of matrices and biases. One possible restriction is to consider a finite collections $\mathcal{F}$ instead.

We begin with the extreme case, in which the collection contains only a single measurable function $\mathcal{F} = \{f : \mathbb{R}^d \to \mathbb{R}^d\}$, for example a ReLU layer $f(\mathbf{x}) = \varrho(\mathbf{W}\mathbf{x} + \mathbf{b})$ with a fixed weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$. As in section 6.4.1 and appendix D.4 we can construct an invariant family of distributions starting with a prototype distribution $\mu_0 \in \mathcal{D}(\mathbb{R}^d)$. Next, we iteratively define

$$\mu_n = f_* \mu_{n-1}, \quad n \in \mathbb{N},$$

and using $\eta(t) = t - \lfloor t \rfloor$ also the intermediate interpolations

$$\mu_t = (1 - \eta(t))\mu_{\lfloor t \rfloor} + \eta(t)\mu_{\lfloor t \rfloor + 1}, \quad t \geq 0. \tag{D.8}$$

We quickly show the Lipschitz continuity of $t \mapsto \mu_t$. Let $0 \leq t_1 < t_2 < \infty$. Without loss of generality, we can assume $t_2 \leq t_1 + 1$, since the Prokhorov metric of two probability distributions is always bounded by 1. First, we consider the case $\lfloor t_2 \rfloor = \lfloor t_1 \rfloor = m$. Then

$$\begin{aligned} d_P(\mu_{t_2}, \mu_{t_1}) &\leq \|\mu_{t_2} - \mu_{t_1}\|_{\mathrm{TV}} \\ &\leq \|(\eta(t_2) - \eta(t_1))(\mu_{m+1} - \mu_m)\|_{\mathrm{TV}} \\ &\leq |\eta(t_2) - \eta(t_1)| \|\mu_{m+1} - \mu_m\|_{\mathrm{TV}} \\ &\leq |t_2 - t_1| (\|\mu_{m+1}\|_{\mathrm{TV}} + \|\mu_m\|_{\mathrm{TV}}) \\ &= 2|t_2 - t_1|, \end{aligned}$$

where $\|\cdot\|_{\mathrm{TV}}$ is the total variation norm.

Second, we consider the case $\lfloor t_2 \rfloor = \lfloor t_1 \rfloor + 1 = m$. Then

$$\mu_{t_2} - \mu_{t_1} = (1 - \eta(t_2) - \eta(t_1))\mu_m + \eta(t_2)\mu_{m+1} - (1 - \eta(t_1))\mu_{m-1}$$

and since

$$|1 - \eta(t_2) - \eta(t_1)| \leq |1 - \eta(t_1)| + |\eta(t_2)| = t_2 - t_1$$

we obtain

$$\begin{aligned} d_P(\mu_{t_2}, \mu_{t_1}) &\leq \|\mu_{t_2} - \mu_{t_1}\|_{\mathrm{TV}} \\ &\leq |1 - \eta(t_2) - \eta(t_1)| + |\eta(t_2)| + |1 - \eta(t_1)| \\ &\leq 2|t_2 - t_1|. \end{aligned}$$

Thus, the one-parameter mapping $t \mapsto \mu_t$ is Lipschitz continuous. It also satisfies $f_* \mu_t = \mu_{t+1}$ for any $t \geq 0$, which means that it is f-invariant (hence $\mathcal{F}$-invariant).

This idea can also be extended to finite (or even countable) collections $\mathcal{F}$. For brevity we only illustrate the concept for a collection $\mathcal{F} = \{f_0, f_1\}$ of two functions. Again, we start by choosing an arbitrary prototype measure, which we will simply denote μ in this case. For any

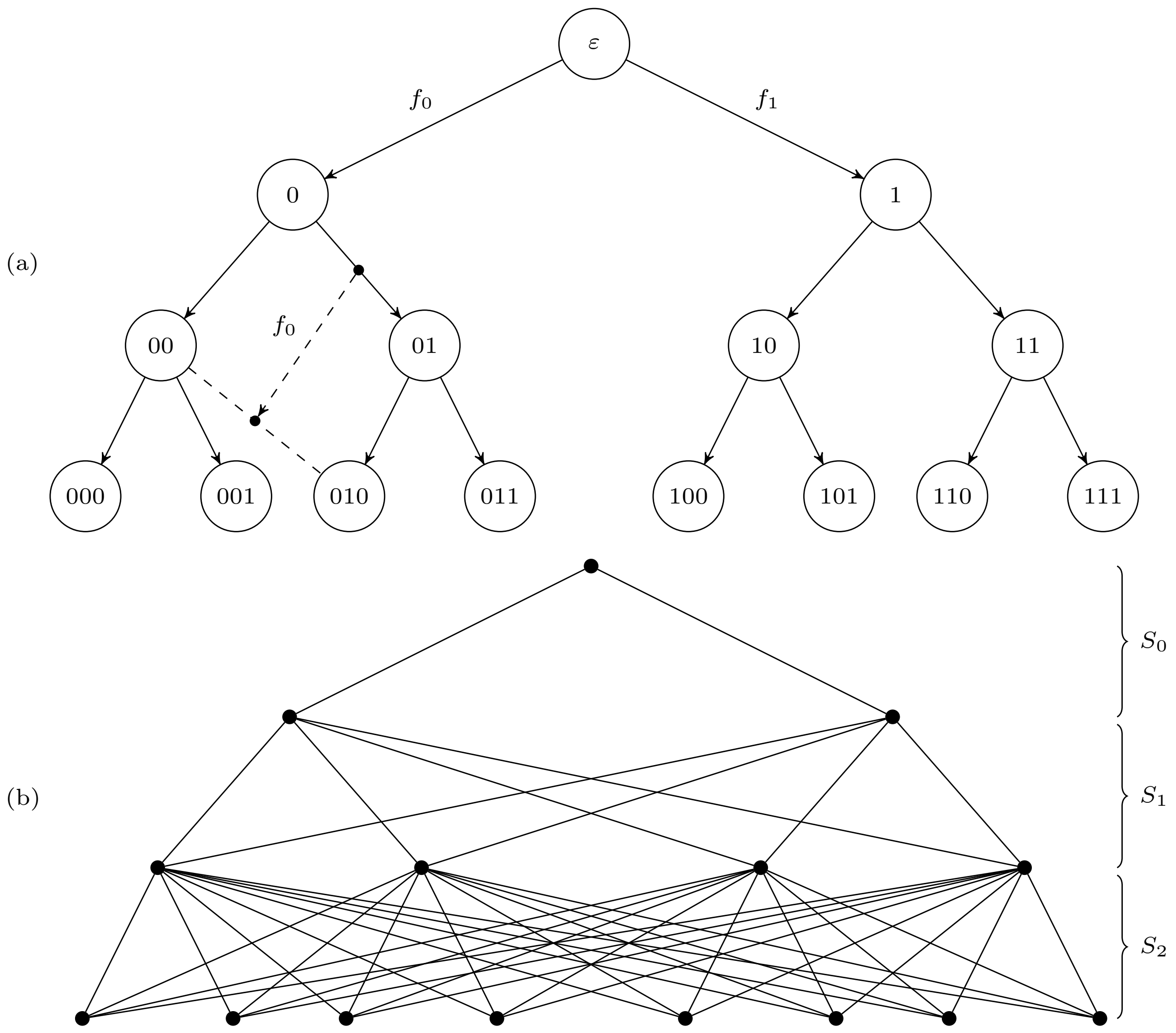


Figure D.5: (a) Consecutively applying either f_0 or f_1 to a prototype measure μ results in an infinite tree structure (only the first three levels are shown). Transforming an intermediate (interpolated) point by f_0 or f_1 (dashed) results in a point outside the tree structure. (b) Adding additional edges to represent also the transformations of all interpolated points results in a multipartite graph in which all vertices of consecutive levels are connected. The subgraphs S_i induced by two consecutive levels of vertices are complete bipartite graphs.

binary string $\mathbf{z} \in \{0,1\}^*$, where ε refers to the empty string, we define

$$\begin{aligned}\mu_\varepsilon &= \mu, \\ \mu_{\mathbf{z}} &= (f_{z_n} \circ \cdots \circ f_{z_1})_* \mu.\end{aligned}$$

The procedure of obtaining measures this way can be associated to a perfect binary tree of infinite depth with root ε. A binary string $\mathbf{z}_1$ is the child of another string $\mathbf{z}_2$ if it extends it by exactly one digit 0 or 1, see fig. D.5. Note that different strings can result in the same measure so that the set of measures $\mu_{\mathbf{z}}$ is not in one-to-one correspondence to the vertices of the tree. However, each measure is represented by at least on vertex.

In order to find a parametrisation that includes all the measures $\mu_{\mathbf{z}}$ and that is locally Lipschitz continuous and $\mathcal{F}$-invariant, we will turn the tree into an undirected graph. The parametrisation will extend the interpolation idea from (D.8) and results from a walk through all nodes of the graph.

Let $l(\mathbf{z})$ denote the length of a binary string $\mathbf{z}$. We define the graph $G = (V, E)$ with vertices $V = \{0,1\}^*$ and edges $E = \{(\mathbf{z}_1, \mathbf{z}_2) \in V \times V \ : \ |l(\mathbf{z}_1) - l(\mathbf{z}_2)| = 1\}$, i.e. every vertex is a binary string and all strings of consecutive lengths are connected by an edge. The resulting graph is a multipartite graph where the independent sets are strings of the same length. Let us define subgraphs S_i of G for $i \in \mathbb{N}_0$ that are induced by vertex sets

$$V_i = \{0,1\}^i \cup \{0,1\}^{i+1}.$$

The resulting subgraphs are complete bipartite graphs, see fig. D.5. We define a walk $\mathcal{W} \in \{V\}^*$ on G as a sequence of vertices where two consecutive vertices are connected by an edge. We are now looking for an infinite walk on G that passes through all edges at least once.

Since we can pass an edge multiple times such a walk is easy to construct: Each subgraph S_i has only finitely many edges, which means there exists a walk that passes through them (except for S_0 we can even find an Eulerian cycle for each subgraph). Without loss of generality we can assume that the walk on S_i starts and ends in $\mathbf{0}_i$ (the string of i zeros) and denote it $\mathcal{W}_i$. Let $\sqcap$ symbolise the concatenation of two walks. We set

$$\mathcal{W} = \prod_{k=0}^{\infty} \mathcal{W}_i,$$

which is possible since in G the two vertices $\mathbf{0}_i$ and $\mathbf{0}_{i+1}$ are connected by an edge for every $i \in \mathbb{N}$. Denoting the i-th vertext visited by $\mathcal{W}$ as $\mathcal{W}(i)$ we can now define

$$\mu_t = (1 - \eta(t))\mu_{\mathcal{W}(\lfloor t \rfloor)} + \eta(t)\mu_{\mathcal{W}(\lfloor t \rfloor + 1)}, \tag{D.9}$$

with $\eta(t)$ as before.

To see that this fulfils our criteria consider the following. All distributions in the parametrised family are of the form

$$(1-s)\mu_{\mathbf{z}_1} + s\mu_{\mathbf{z}_2}, \quad \text{with} \quad \mathbf{z}_1 \in \{0,1\}^k, \mathbf{z}_2 \in \{0,1\}^{k+1}, k \in \mathbb{N}_0, s \in [0,1].$$

Transforming it by the function f_i for $i \in \{0,1\}$ results in the pushforward measure

$$(1-s)\mu_{\mathbf{z}_1'} + s\mu_{\mathbf{z}_2'}, \quad \text{with} \quad \mathbf{z}_1' = (\mathbf{z}_1, i), \mathbf{z}_2' = (\mathbf{z}_2, i).$$

But since $\mathbf{z}_1'$ and $\mathbf{z}_2'$ still have length difference 1, they share an edge in S^{k+1}. The walk $\mathcal{W}$ passes through all edges, so we can define $m \in \mathbb{N}$ as the smallest number where either $\mathcal{W}_m = \mathbf{z}_1'$ and $\mathcal{W}_{m+1} = \mathbf{z}_2'$ or $\mathcal{W}_m = \mathbf{z}_2'$ and $\mathcal{W}_{m+1} = \mathbf{z}_1'$. Let us without loss of generality assume that the former holds. Then

$$(1-s)\mu_{\mathbf{z}_1'} + s\mu_{\mathbf{z}_2'} = \mu_{m+s},$$

which shows $t \mapsto \mu_t$ is $\mathcal{F}$-invariant. The Lipschitz continuity follows analogously to the case with only a single function above.

This idea can even be extended to countably many different measurable functions. Instead of a single binary tree and corresponding graph G, we could define a sequence of trees T_i and corresponding graphs G_i where T_i is i-ary, corresponding to the first i functions. Every tree T_i is a subtree of T_j whenever $j > i$ and correspondingly the associated graph G_i is a subgraph of G_j. Thus any walk on G_i is also valid on G_j. If now $\mathcal{W}_i$ denotes a walk on G_i starting at the root ε, covering every edge up to the i-th level in G_i, and returning back to the root ε, then we can set

$$\mathcal{W} = \prod_{k=1}^{\infty} \mathcal{W}_i.$$

It follows that every edge of every graph in the sequence will eventually be reached. Defining the parametrisation as in (D.9), ensures that both $\mathcal{F}$-invariance as well as Lipschitz continuity holds.

E Numerical Evaluation

E.1 Detailed Description of the Synthetic Binary Strings Experiment

E.1.1 Network Architecture

Recall that the underlying binary classifier is given by the Boolean function

$$\Psi\colon \{0,1\}^d \to \{0,1\}, \quad \mathbf{x} \mapsto \bigvee_{i=1}^{d-k+1} \bigwedge_{j=i}^{i+k-1} x_j,$$

that checks binary strings of length d for the existence of a block of k consecutive ones. A ReLU network with two hidden layers that interpolates Ψ can be constructed as

$$\Phi(\mathbf{x}) = \mathbf{W}_3 \varrho \left(\mathbf{W}_2 \varrho(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2\right) + \mathbf{b}_3$$

with

$$\begin{aligned}
\mathbf{W}_1 &= \left[\sum_{j=i}^{i+k-1} \mathbf{e}_j^T\right]_{i=1}^{d-k+1} \in \mathbb{R}^{(d-k+1)\times d} && \text{and} \quad \mathbf{b}_1 = -(k-1)\cdot \mathbf{1}_{d-k+1} \in \mathbb{R}^{d-k+1}, \\
\mathbf{W}_2 &= -\mathbf{1}_{d-k+1}^\top \in \mathbb{R}^{1\times(d-k+1)} && \text{and} \quad \mathbf{b}_2 = 1 \in \mathbb{R}^1, \\
\mathbf{W}_3 &= -1 \in \mathbb{R}^{1\times 1} && \text{and} \quad \mathbf{b}_3 = 1 \in \mathbb{R}^1,
\end{aligned}$$

where $\mathbf{e}_j$ is the j-th unit vector in $\mathbb{R}^d$. This network is purely constructed and not trained on any data. We use $d = 16$ and $k = 5$ in our experiment.

E.1.2 RDE Optimisation

For the RDE optimisation we used the regularisation parameter $\lambda = 1.67 \cdot 10^{-3}$ and solved the resulting box-constrained optimisation problem via L-BFGS-B [Byr+95]. For the low-rank variant we used the full rank $r = 16$.

The initial guess for $\mathbf{s}$ was simply chosen as the mean of $\mathcal{U}\Big([0,1]^d\Big)$ and not further tuned. As reference distribution $\mathcal{V}$ we used the Gaussian distribution with mean and variance equal to the mean and variance of $\mathcal{U}\Big([0,1]^d\Big)$. To estimate a good value for the regularisation parameter λ we solved the RDE optimisation problem for values $\lambda = 10^q$ with ten values of q spaced evenly in $[-5, 0]$. We compared the results visually and saw that $1.67 \cdot 10^{-3}$ yields a relevance map with a sparsity that corresponds well to the true block size $k = 5$.

E.1.3 Comparison Methods

We used the Innvestigate toolbox for generating relevance mappings according to SmoothGrad [Smi+17] with a noise scale of 0.5 and 64 noise samples. We used the SHAP toolbox to generate relevance mappings according to SHAP [LL17b] and used the `DeepExplainer` method for deep network models with 1024 reference inputs drawn randomly from $\mathcal{U}\Big([0,1]^d\Big)$. Finally, we used the LIME toolbox to generate relevance mappings according to LIME [RSG16a]. We used the local explanations of the `LimeTabularExplainer` method with 1024 reference inputs drawn randomly from $\mathcal{U}\Big([0,1]^d\Big)$.

E.2 Detailed Description of the An8flower Experiment

E.2.1 Network Architecture and Training

For the an8flower experiments we used a convolutional neural network with three convolution layers each followed by average-pooling and finally two fully-connected layers and softmax output, see Table E.1 for a detailed description of the architecture. We trained the randomly initialised network end-to-end for 100 epochs to a final test accuracy of 0.99. We used cross-entropy loss and stochastic gradient descent with mini-batches of size 64 and a learning rate of $1 \cdot 10^{-3}$. A training/testing split of 90% / 10% of the data set was used.

E.2.2 RDE Optimisation

We used the pre-softmax score of the class with the highest activation as the prediction $\Phi(\mathbf{x})$. Since the pre-softmax scores are not guaranteed to lie in the range $[0, 1]$ we normalised the distortion function by $\Phi(\mathbf{x})$. For the RDE optimisation we used the regularisation parameter $\lambda = 5.0$ and L-BFGS-B with initial guess $\mathbf{s} = 0.5 \cdot \mathbf{1}_d$. For the low-rank variant we used the rank $r = 10$.

We estimated good values for the regularisation parameter λ and the rank r of the low-rank variant as follows: We used a single randomly chosen image signal from the test set and solved the RDE optimisation problem for values $\lambda = 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}$ to get a first rough estimate. We compared the results visually and determined 10^0 and 10^1 as good candidates. Thus, we refined the search using values $\lambda = 1, 2, \dots, 10$ and finally selected $\lambda = 5$. Once the regularisation parameter was determined we proceeded similarly for the rank and tested the values $r = 5, 10, 30, 50, 100$. We observed that choosing ranks larger than 10 had negligible effect on the relevance mappings and selected $r = 10$ as a promising option.

Table E.1: Architecture of the convolutional neural network for the an8flower data set.

layer	feature maps	size	kernel size	strides	activation
input	3	$128 \times 128 \times 3$	-	-	-
convolutional	32	$128 \times 128 \times 32$	5×5	1×1	ReLU
average pooling	32	$64 \times 64 \times 32$	2×2	2×2	-
convolutional	64	$64 \times 64 \times 64$	5×5	1×1	ReLU
average pooling	64	$32 \times 32 \times 64$	2×2	2×2	-
convolutional	64	$32 \times 32 \times 64$	5×5	1×1	ReLU
average pooling	64	$16 \times 16 \times 64$	2×2	2×2	-
flatten	-	16384	-	-	-
fully connected	-	1024	-	-	ReLU
fully connected	-	12	-	-	Softmax
output	-	12	-	-	-

E.2.3 Comparison Methods

We used the Innvestigate toolbox for generating relevance mappings according to Layer-wise Relevance Propagation (LRP) [Bac+15b], Deep Taylor decompositions [MSM18], Sensitivity Analysis [SVZ14], SmoothGrad [Smi+17], and Guided Backprop [Spr+15]. We used LRP-α-β with the parameter preset `SequentialPresetAFlat` recommended for convolutional networks and numerical stability parameter $\epsilon = 0.2$. We used the bounded Deep Taylor method with lower and upper bounds given by $\min_i x_i$ and $\max_i x_i$ respectively. For SmoothGrad we used a noise scale of $0.3 \cdot (\max_i x_i - \min_i x_i)$ and 64 noise samples. We used the SHAP toolbox to generate relevance mappings according to SHAP [LL17b] and used the `DeepExplainer` method for deep network models with 8 reference inputs generated as the component-wise mean of 8 mini-batches from the training data set. Finally, we used the LIME toolbox to generate relevance mappings according to LIME [RSG16a]. We used the local explanations of the `LimeImageExplainer` method recommended for image data.

E.3 Detailed Description of the Relevance Ordering Test Experiment for MNIST

E.3.1 Network Architecture and Training

For the MNIST experiment we used a convolutional neural network with three convolution layers each followed by average-pooling and finally two fully-connected layers and softmax output, see Table E.2 for a detailed description of the architecture. We trained the randomly initialised network end-to-end for 100 epochs to a final test accuracy of 0.99. We used cross-entropy loss and stochastic gradient descent with mini-batches of size 128 and a learning rate of $3 \cdot 10^{-3}$. The standard training/validation/testing split of the data set was used. We augmented the training data by random shifts up to 0.05 of the image width and height respectively, rotations up to 5 degree, shearing up to 0.05 degree, and zoom by a factor in the range $[0.995, 1.05]$. Further, we pre-processed all data by sample-wise mean centering.

E.3.2 RDE Optimisation

We used the pre-softmax score of the class with the highest activation as the prediction $\Phi(\mathbf{x})$. Since the pre-softmax scores are not guaranteed to lie in the range $[0, 1]$ we normalised the distortion function by $\Phi(\mathbf{x})$. For the RDE optimisation we used the regularisation parameter $\lambda = 0.5$ and projected gradient descent with initial guess $\mathbf{s} = 0.25 \cdot \mathbf{1}_d$, a momentum term with factor 0.85, and the step size determined according to a backtracked Armijo line search [NW06]. For the low-rank variant we used the rank $r = 30$.

The momentum factor was chosen based on previous experiences and not further tuned. Similarly, the initial guess was chosen rather arbitrarily and was not further tuned. We estimated good values for the regularisation parameter λ and the rank r of the low-rank variant as follows: We used a single randomly chosen image signal from the test set and solved the RDE optimisation problem for values $\lambda = 10^{-3}, 10^{-2}, \dots, 10^{2}, 10^{3}$ to get a first rough estimate. We compared the results visually as well as using the relevance ordering test described below and determined 10^{-1} and 10^{0} as good candidates. Thus, we refined the search using values $\lambda = 0.1, 0.2, \dots, 1.0$ and finally selected $\lambda = 0.5$. Once the regularisation parameter was determined we proceeded similarly for the rank and tested the values $r = 10, 30, 50, 100, 300, 500, 784$. We observed that choosing ranks larger than 30 had negligible effect on the relevance mappings and selected $r = 30$ as a promising option.

E.3.3 Comparison Methods

We used the Innvestigate toolbox for generating relevance mappings according to Layer-wise Relevance Propagation (LRP) [Bac+15b], Deep Taylor decompositions [MSM18], Sensitivity Analysis [SVZ14], SmoothGrad [Smi+17], and Guided Backprop [Spr+15]. We used LRP-α-β with the parameter preset `SequentialPresetAFlat` recommended for convolutional networks and numerical stability parameter $\epsilon = 0.2$. We used the bounded Deep Taylor method with lower and upper bounds given by $\min_i x_i$ and $\max_i x_i$ respectively. For SmoothGrad we used a noise scale of $0.3 \cdot (\max_i x_i - \min_i x_i)$ and 64 noise samples. We used the SHAP toolbox to generate relevance mappings according to SHAP [LL17b] and used the `DeepExplainer` method for deep network models with 8 reference inputs generated as the component-wise mean of 8 mini-batches from the training data set. Finally, we used the LIME toolbox to generate relevance mappings according to LIME [RSG16a]. We used the local explanations of the `LimeImageExplainer` method recommended for image data.

E.3.4 Relevance Ordering Comparison Test

For the relevance ordering based comparison test we sorted components according to their relevance scores (breaking ties randomly). Then, starting with a completely random signal, we replaced increasingly large parts of it by the original input, and observed the change in the classifier score. Here, we increased the set of fixed components always in groups of size 5 (resulting in 157 steps until all 784 components have been fixed) and used 512 random samples drawn from $\mathcal{U}\left([0, 1]^d\right)$ per step. This procedure was repeated and the results were averaged over 50 different input signals spaced evenly over the test data set (5 images for each of the 10 classes).

Table E.2: Architecture of the convolutional neural network for the MNIST data set.

layer	feature maps	size	kernel size	strides	activation
input	1	$28 \times 28 \times 1$	-	-	-
convolutional	32	$28 \times 28 \times 32$	5×5	1×1	ReLU
average pooling	32	$14 \times 14 \times 32$	2×2	2×2	-
convolutional	64	$14 \times 14 \times 64$	5×5	1×1	ReLU
average pooling	64	$7 \times 7 \times 64$	2×2	2×2	-
convolutional	64	$7 \times 7 \times 64$	5×5	1×1	ReLU
average pooling	64	$3 \times 3 \times 64$	2×2	2×2	-
flatten	-	576	-	-	-
fully connected	-	1024	-	-	ReLU
dropout (30%)	-	1024	-	-	-
fully connected	-	10	-	-	Softmax
output	-	10	-	-	-

E.4 Detailed Description of the Relevance Ordering Test Experiment for STL-10

E.4.1 Network Architecture and Training

For the STL-10 experiment we used a VGG-16 network [SZ14] pre-trained on the Imagenet dataset as a baseline, see Table E.3 for a detailed description of the architecture. We refined the network for the STL-10 dataset in three stages. First, we fixed the convolutional blocks of the Imagenet network and trained only the fully-connected layers for 500 epochs. Then we trained the complete network end-to-end for another 500 epochs. Finally, after replacing all max-pooling layers by average-pooling layers, we again trained end-to-end for another 500 epochs to a final test accuracy of 0.935. We used cross-entropy loss and stochastic gradient descent with mini-batches of size 64, a learning rate of $3 \cdot 10^{-7}$, and momentum term with factor 0.9 in all three stages. For the weight matrices of the fully connected layers we used a combined ℓ_2- and ℓ_1-regularisation term with regularisation parameters $5 \cdot 10^{-4}$ and $5 \cdot 10^{-5}$ respectively. The standard training/validation/testing split of the data set was used. We augmented the training data by random shifts up to 0.2 of the image width and height respectively, rotations up to 20 degree, shearing up to 0.2 degree, zoom by a factor in the range $[0.8, 1.2]$, and horizontal flipping.

E.4.2 RDE Optimisation

We used the pre-softmax score of the class with the highest activation as the prediction $\Phi(\mathbf{x})$. Since the pre-softmax scores are not guaranteed to lie in the range $[0, 1]$ we normalised the distortion function by $\Phi(\mathbf{x})$. For the RDE optimisation we used the regularisation parameter $\lambda = 0.1$ and projected gradient descent with initial guess $\mathbf{s} = 0.25 \cdot \mathbf{1}_d$, a momentum term with factor 0.85, and the step size determined according to a backtracked Armijo line search [NW06]. For the low-rank variant we used the rank $r = 30$.

The momentum factor was chosen based on previous experiences and not further tuned. Similarly, the initial guess was chosen rather arbitrarily and was not further tuned. We estimated good values for the regularisation parameter λ and the rank r of the low-rank variant as follows: We used a single randomly chosen image signal from the test set and

Table E.3: Architecture of the VGG-16 based convolutional neural network for the STL-10 data set.

layer	feature maps	size	kernel size	strides	activation
input	3	$224 \times 224 \times 3$	-	-	-
convolutional	64	$224 \times 224 \times 64$	3×3	1×1	ReLU
convolutional	64	$224 \times 224 \times 64$	3×3	1×1	ReLU
average pooling	64	$112 \times 112 \times 64$	2×2	2×2	-
convolutional	128	$112 \times 112 \times 128$	3×3	1×1	ReLU
convolutional	128	$112 \times 112 \times 128$	3×3	1×1	ReLU
average pooling	128	$56 \times 56 \times 128$	2×2	2×2	-
convolutional	256	$56 \times 56 \times 256$	3×3	1×1	ReLU
convolutional	256	$56 \times 56 \times 256$	3×3	1×1	ReLU
convolutional	256	$56 \times 56 \times 256$	3×3	1×1	ReLU
average pooling	256	$28 \times 28 \times 256$	2×2	2×2	-
convolutional	512	$28 \times 28 \times 512$	3×3	1×1	ReLU
convolutional	512	$28 \times 28 \times 512$	3×3	1×1	ReLU
convolutional	512	$28 \times 28 \times 512$	3×3	1×1	ReLU
average pooling	512	$14 \times 14 \times 512$	2×2	2×2	-
convolutional	512	$14 \times 14 \times 512$	3×3	1×1	ReLU
convolutional	512	$14 \times 14 \times 512$	3×3	1×1	ReLU
convolutional	512	$14 \times 14 \times 512$	3×3	1×1	ReLU
average pooling	512	$7 \times 7 \times 512$	2×2	2×2	-
flatten	-	25088	-	-	-
dropout (50%)	-	25088	-	-	-
fully connected	-	4096	-	-	ReLU
dropout (50%)	-	4096	-	-	-
fully connected	-	4096	-	-	ReLU
dropout (50%)	-	4096	-	-	-
fully connected	-	10	-	-	Softmax
output	-	10	-	-	-

solved the RDE optimisation problem for values $\lambda = 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}$ to get a first rough estimate. We compared the results visually as well as using the relevance ordering test described below and determined 10^{-1} and 10^{0} as good candidates. Thus, we refined the search using values $\lambda = 0.1, 0.2, \ldots, 1.0$ and finally selected $\lambda = 0.1$. Once the regularisation parameter was determined we proceeded similarly for the rank and tested the values $r = 10, 30, 50, 100, 300, 500$. We observed that choosing ranks larger than 30 had negligible effect on the relevance mappings and selected $r = 30$ as a promising option.

E.4.3 Comparison Methods

See above. We used the same settings as in the MNIST experiment for all methods.

E.4.4 Relevance Ordering Comparison Test

For the relevance ordering based comparison test we sorted components according to their relevance scores (breaking ties randomly). Then, starting with a completely random signal, we replaced increasingly large parts of it by the original input, and observed the change in the classifier score. Here, we increased the set of fixed components always in groups of size 512 (resulting in 294 steps until all 150528 components have been fixed) and used 128 random samples drawn from $\mathcal{U}\Big([0,1]^d\Big)$ per step. This procedure was repeated and the results were averaged over 50 different input signals spaced evenly over the test data set (5 images for each of the 10 classes).

E.5 Additional Experimental Results

Figure E.1 is a larger and annotated version of Figure 7.5 in the main paper, showing the relevance ordering test results for the MNIST and STL-10 experiments. This is complemented by Figure E.2, showing the results of the corresponding comparison test using classification accuracy instead of the squared distance distortion as performance measure. Finally, Figures E.3 to E.12 show further mappings from the STL-10 experiment (one image for each of the ten classes).

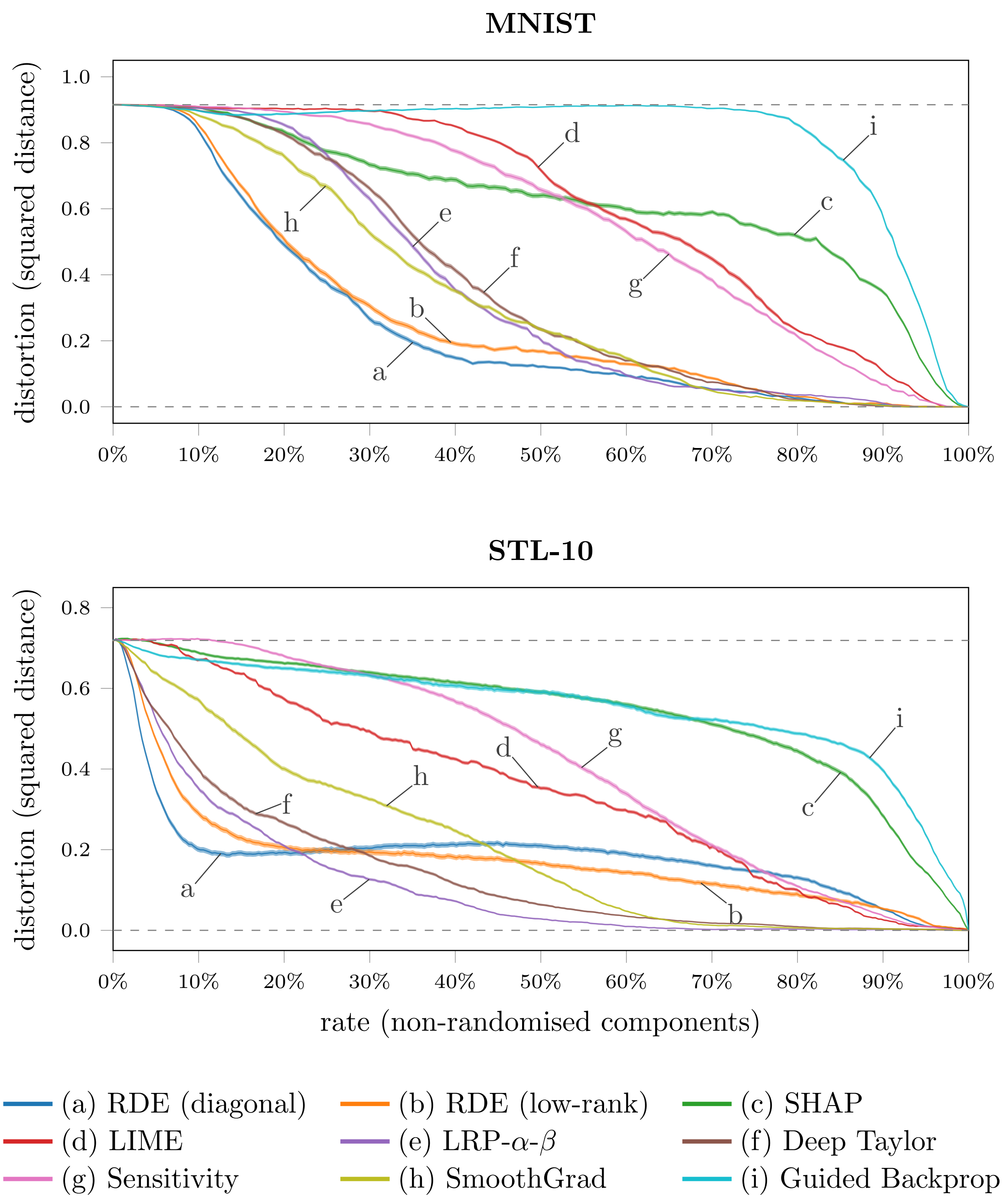


Figure E.1: Relevance ordering test results of several methods for the MNIST (top) and STL-10 (bottom) dataset using squared distance as performance measure. An average result over 50 images from the respective test set (5 images per class randomly selected) and 512 (MNIST) and 64 (STL-10) random input samples per image is shown (shaded regions mark $\pm$ standard deviation). This a larger version of Figure 7.5 in the main paper. See Figure E.2 for a complementing evaluation of the same relevance maps using classification accuracy as an alternative performance measure.

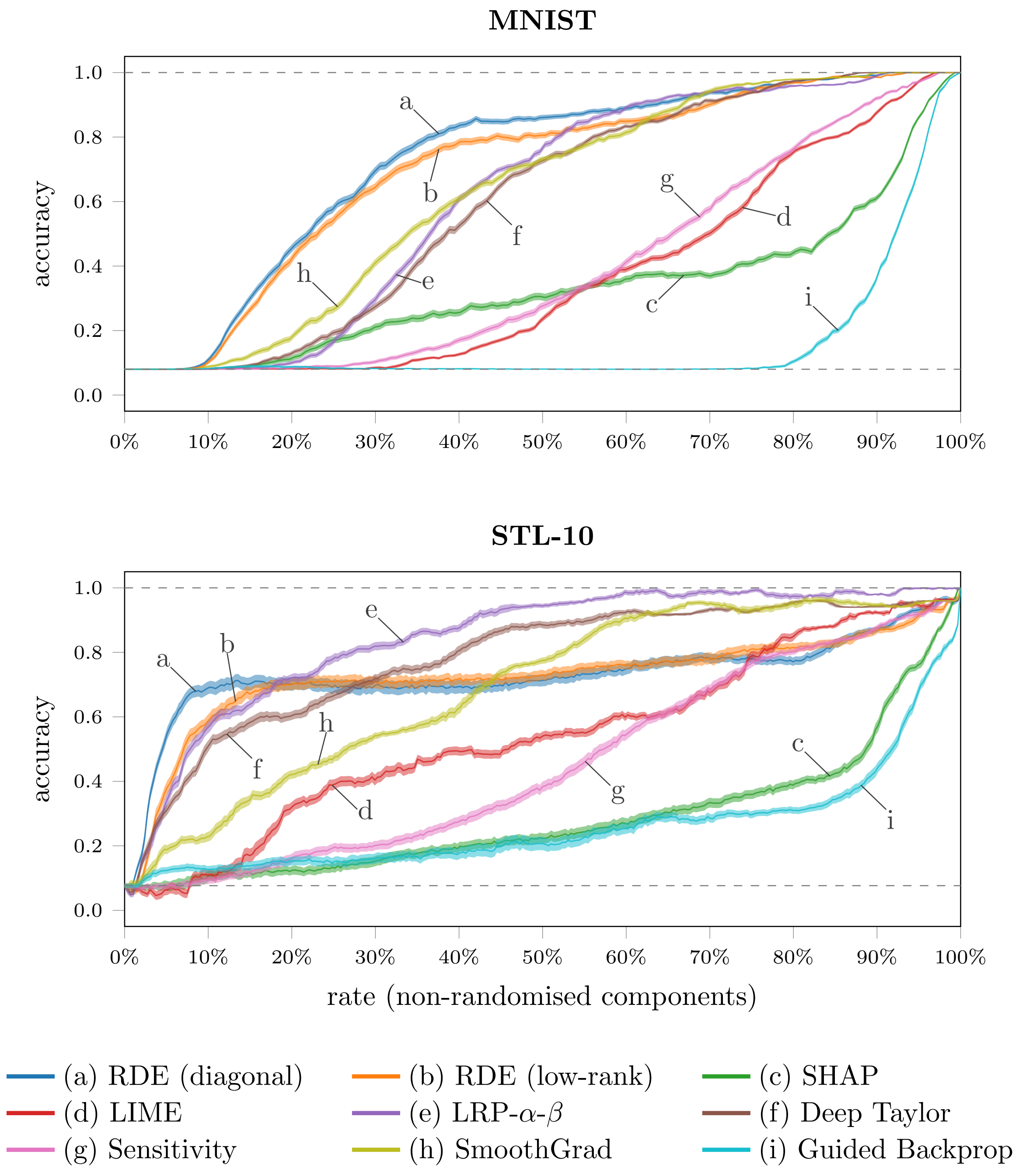


Figure E.2: Relevance ordering test results of several methods for the MNIST (top) and STL-10 (bottom) dataset using classification accuracy as performance measure. An average result over 50 images from the respective test set (5 images per class randomly selected) and 512 (MNIST) and 64 (STL-10) random input samples per image is shown (shaded regions mark ± standard deviation). This complements Figure E.1, which shows an evaluation of the same relevance maps using squared distance (distortion) as performance measure.

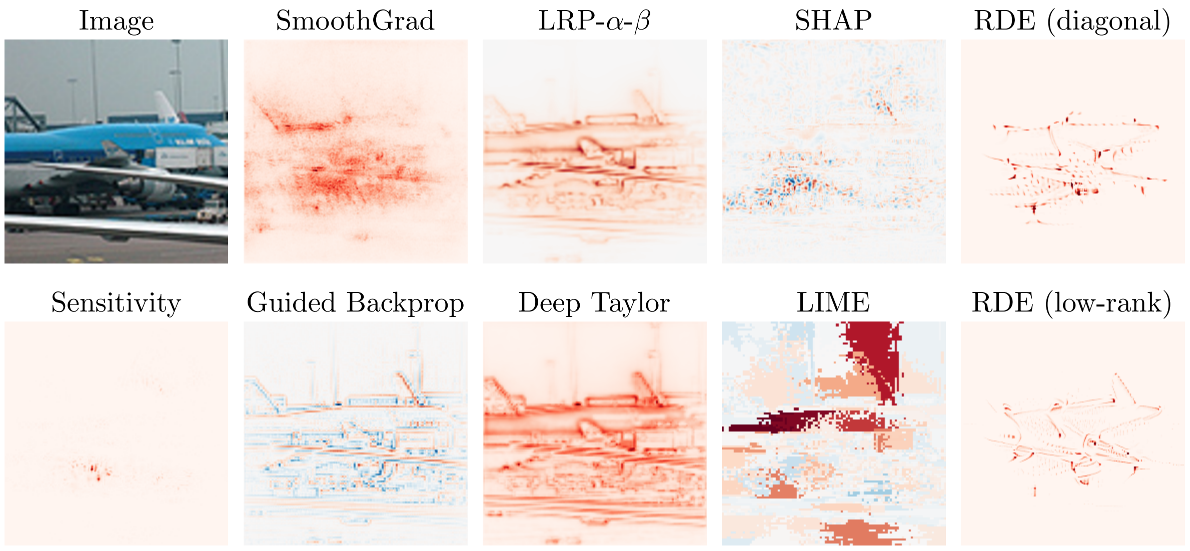


Figure E.3: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *airplane* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.

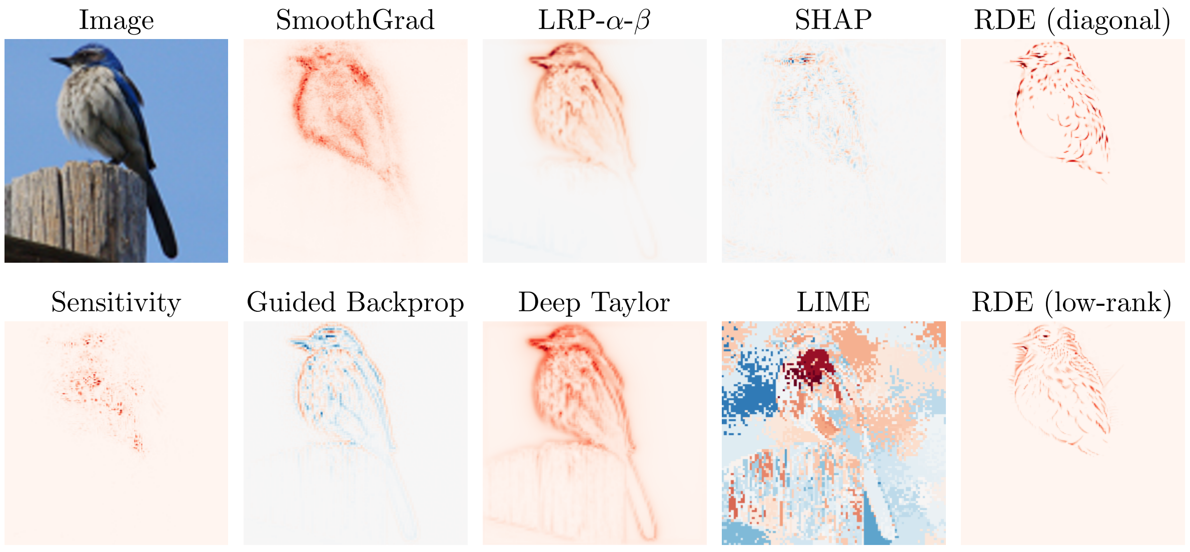


Figure E.4: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *bird* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.

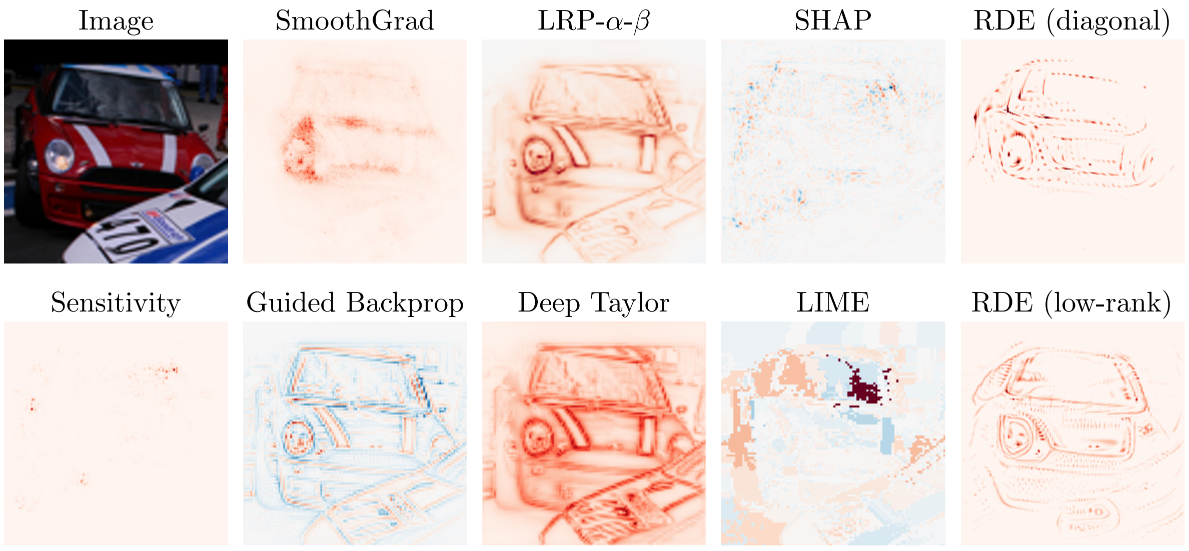


Figure E.5: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *car* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.

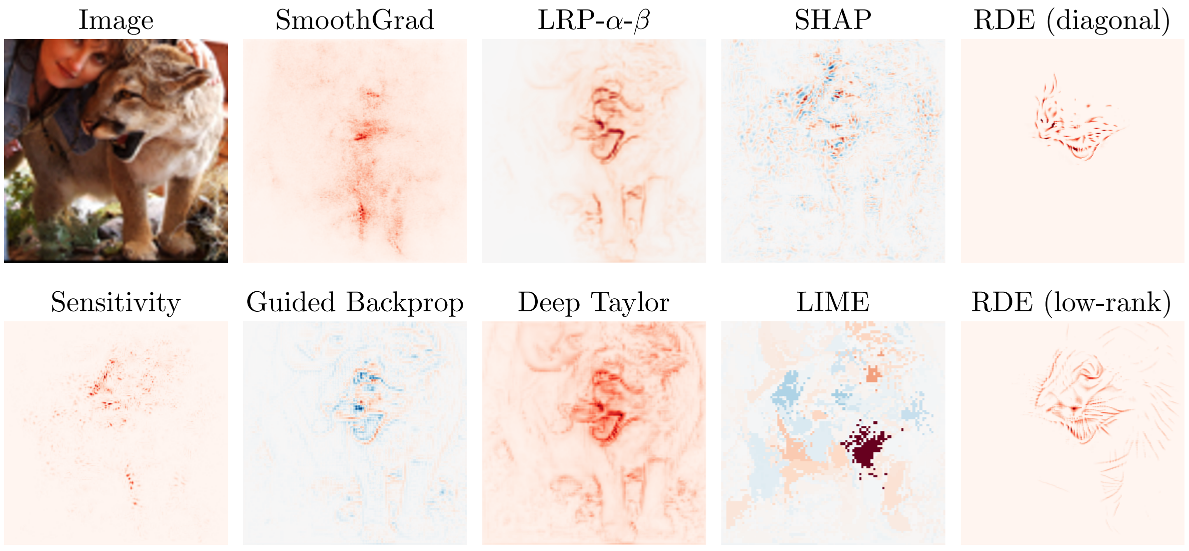


Figure E.6: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *cat* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.

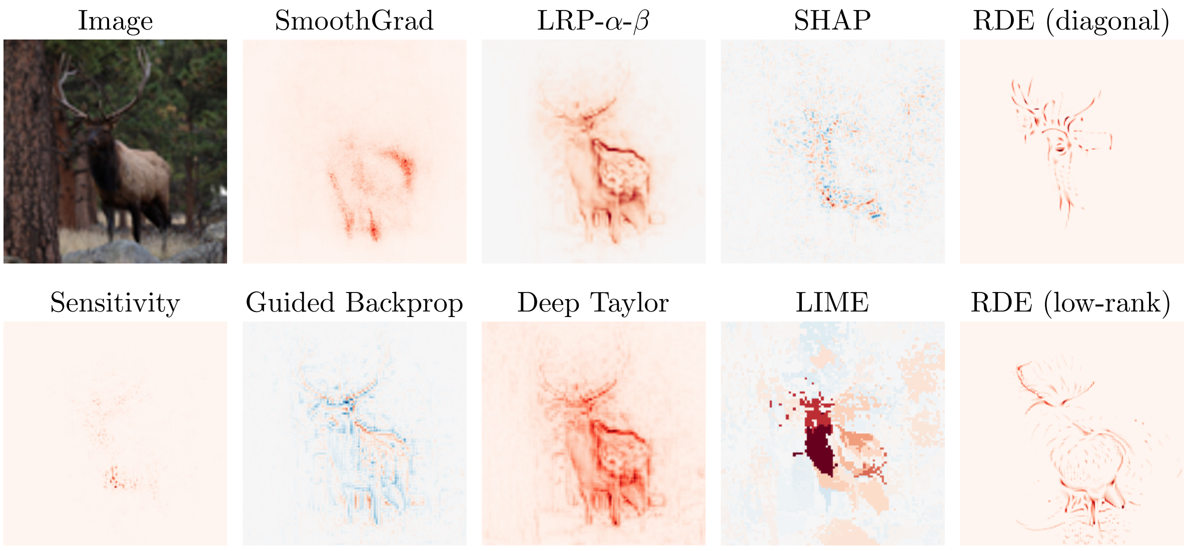


Figure E.7: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *deer* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.

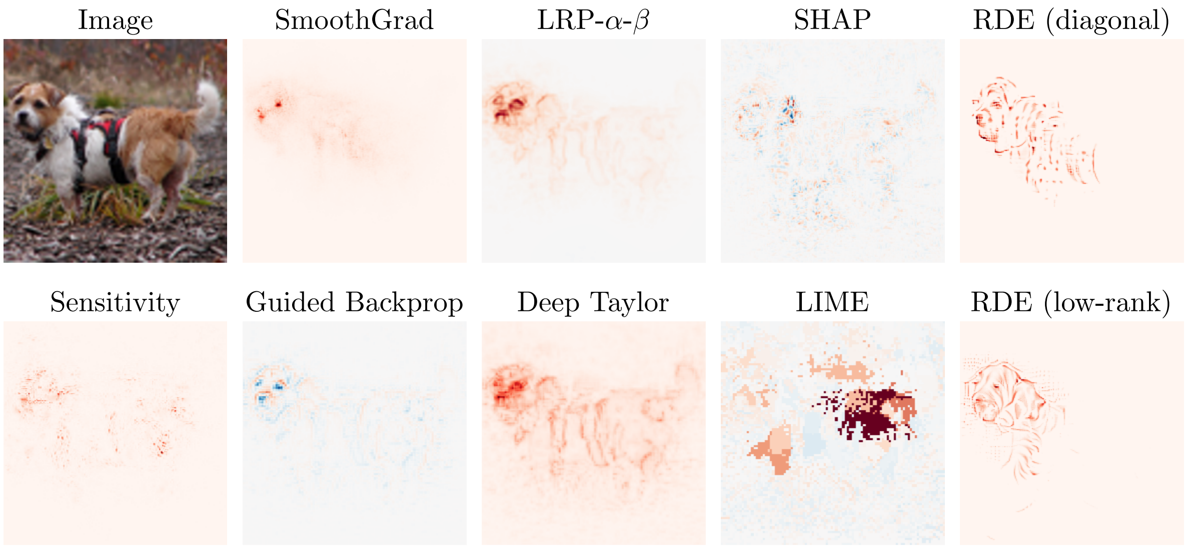


Figure E.8: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *dog* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.

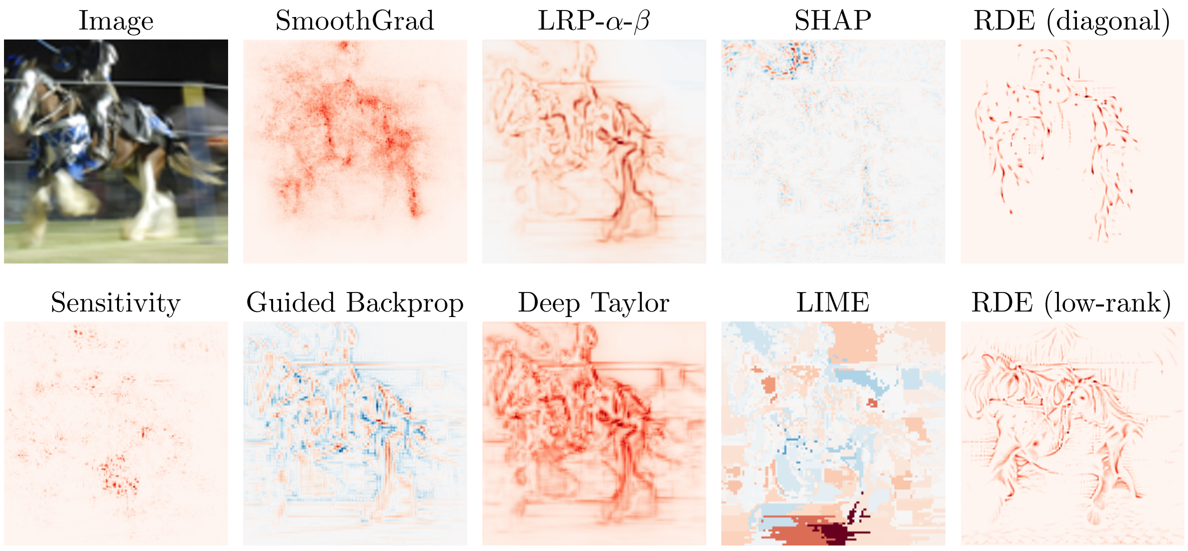


Figure E.9: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *horse* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.

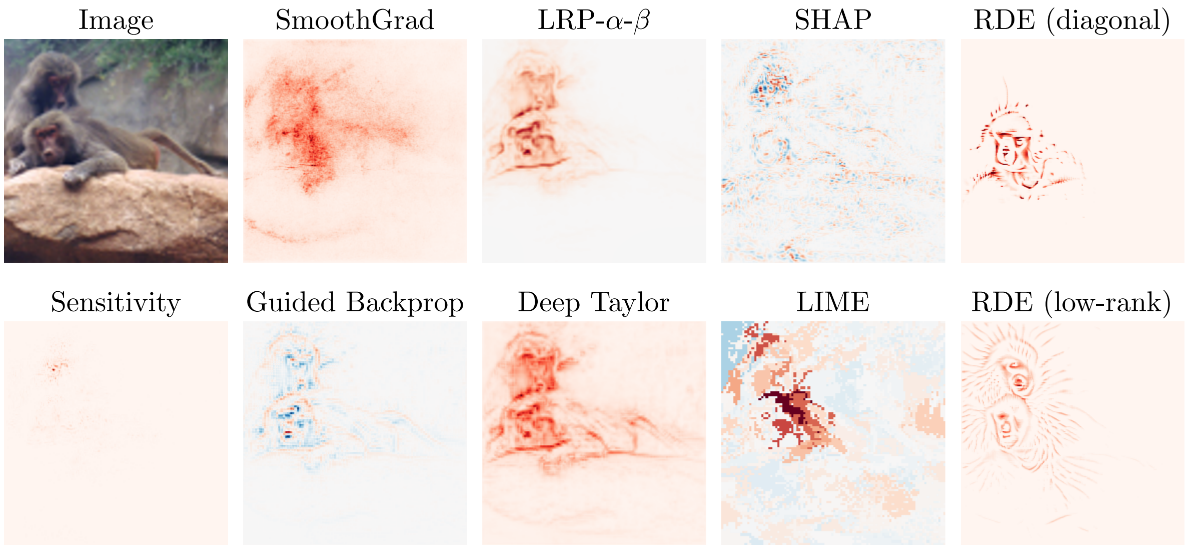


Figure E.10: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *monkey* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.

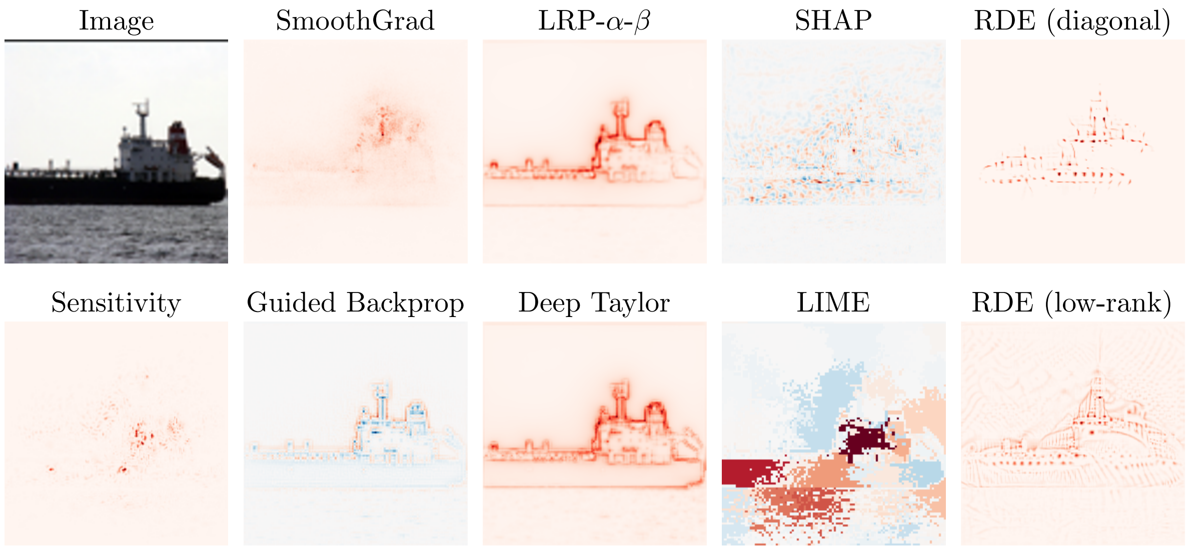


Figure E.11: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *ship* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.

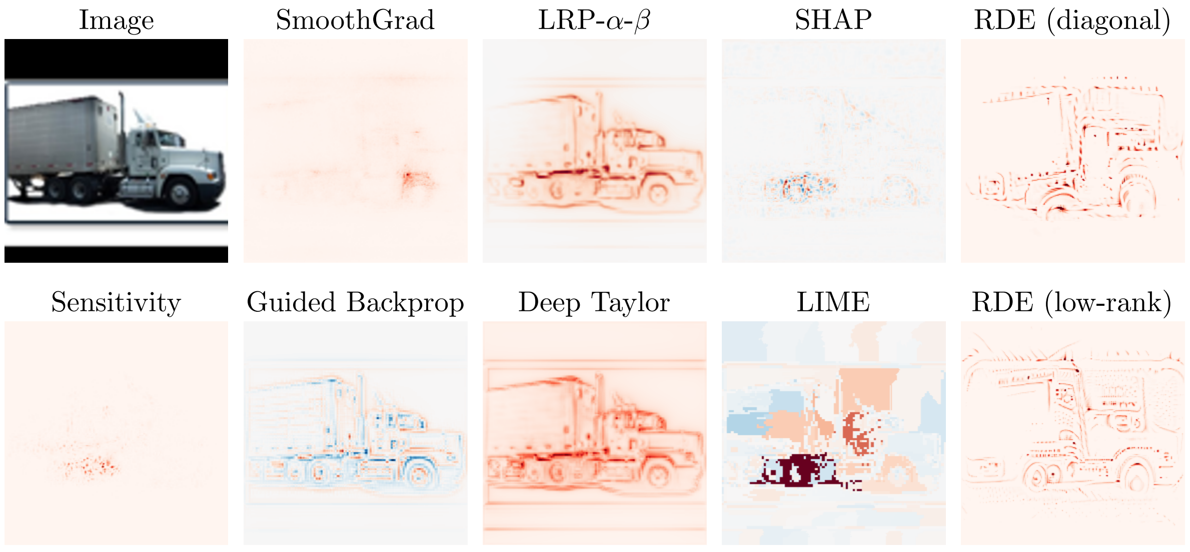


Figure E.12: Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *truck* by our network. The colourmap indicates positive relevance scores as red and negative relevance scores as blue.