The Reliability of Deep Learning for Signal and Image Processing: Interpretability, Robustness, and Accuracy

vorgelegt von

Jan Lukas Macdonald (M. Sc.)

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

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Tobias Breiten
Gutachter:	Prof. Dr. Martin Skutella
Gutachter:	Prof. Dr. Sebastian Pokutta
Gutachter:	Prof. Dr. Reinhard Heckel
Gutachter:	Prof. Dr. Lars Ruthotto

Tag der wissenschaftlichen Aussprache: 13. Oktober 2022

Berlin, 2022

Abstract

This thesis investigates several aspects of using data-driven methods for image and signal processing tasks, particularly those aspects related to the reliability of approaches based on deep learning. It is organized in two parts.

The first part studies the interpretability of predictions made by neural network classifiers. A key component for achieving interpretable classifications is the identification of relevant input features for the predictions. While several heuristic approaches towards this goal have been proposed, there is yet no generally agreed-upon definition of relevance. Instead, these heuristics typically rely on individual (often not explicitly stated) notions of interpretability, making comparisons of results difficult. The contribution of the first part of this thesis is the introduction of an explicit definition of relevance of input features for a classifier prediction and an analysis thereof. The formulation is based on a rate-distortion trade-off and derived from the observation and identification of common questions that practitioners would like to answer with relevance attribution methods. It turns out that answering these questions is extremely challenging: A computational complexity analysis reveals the hardness of determining the most relevant input features (even approximately) for Boolean classifiers as well as for neural network classifiers. This hardness in principle justifies the adoption of heuristic strategies and the explicit rate-distortion formulation inspires a novel approach that specifically aims at answering the identified questions of interest. Furthermore, it allows for a quantitative evaluation of relevance attribution methods, revealing that the newly proposed heuristic performs best in identifying the relevant input features compared to previous methods.

The second part studies the accuracy and robustness of deep learning methods for the reconstruction of signals from undersampled indirect measurements. Such inverse problems arise for example in medical imaging, geophysics, communication, or astronomy. While widely used classical variational solution methods come with reconstruction guarantees (under suitable assumptions), the underlying mechanisms of data-driven methods are mostly not well understood from a mathematical perspective. Nevertheless, they show promising results and frequently empirically outperform classical methods in terms of reconstruction quality and speed. However, several doubts remain regarding their reliability, in particular questions concerning their robustness to perturbations. Indeed, for classification tasks it is well known that neural networks are vulnerable to adversarial perturbations, i.e., tiny modifications that are visually imperceptible but mislead the neural network to make a wrong prediction. This raises the question if similar effects also occur in the context of signal recovery. The contribution of the second part of this thesis is an extensive numerical study of the robustness of a representative selection of end-to-end neural networks for solving inverse problems. It is demonstrated that for such regression problems (in contrast to classification) neural networks can be remarkably robust to adversarial and statistical perturbations. Furthermore, they show state-of-the-art performance resulting in highly accurate reconstructions: In the idealistic scenario of synthetic and perturbation-free data neural networks have the potential to achieve near-perfect reconstructions, i.e., their reconstruction error is close to numerical precision.

Zusammenfassung

In dieser Dissertation werden verschiedene Aspekte der Verwendung datengestützter Methoden für die Bild- und Signalverarbeitung untersucht, insbesondere die Zuverlässigkeit von Deep Learning Ansätzen. Die Arbeit ist in zwei Teile gegliedert.

Der erste Teil untersucht die Interpretierbarkeit von Klassifikationsvorhersagen, die von neuronalen Netzen gemacht werden. Eine Schlüsselkomponente für eine interpretierbare Klassifikation ist die Identifizierung der relevanten Eingabegrößen für eine Vorhersage. Es wurden zwar bereits zahlreiche heuristische Ansätze zur Erreichung dieses Ziels vorgeschlagen, doch gibt es keine allgemein anerkannte Definition für die Relevanz. Stattdessen beruhen diese Heuristiken in der Regel auf individuellen (oft nicht explizit genannten) Auffassungen von Interpretierbarkeit, was einen Vergleich der Ergebnisse erschwert. Der wissenschaftliche Beitrag des ersten Teils dieser Arbeit ist die Einführung sowie die Analyse einer expliziten Definition für die Relevanz von Eingabegrößen für die Vorhersage einer Klassifikationsfunktion. Die Formulierung basiert auf einem Rate-Distortion-Trade-Off und leitet sich aus der Feststellung und Identifizierung von gängigen Fragen ab, die in Anwendungen mit Hilfe von Relevanzbewertungsmethoden beantwortet werden sollen. Wie sich herausstellt, ist die Beantwortung dieser Fragen jedoch äußerst schwierig: Eine Untersuchung der rechnerischen Komplexität zeigt, wie aufwendig es ist, die relevantesten Eingabegrößen für Boolesche Klassifikatoren und für Klassifikatoren auf Basis von neuronalen Netzen (auch nur approximativ) zu bestimmen. Diese Schwierigkeit rechtfertigt prinzipiell die Anwendung heuristischer Strategien. Ein neuartiger Ansatz, der speziell auf die Beantwortung der identifizierten Fragen von praktischem Interesse abzielt, lässt sich direkt aus der expliziten Rate-Distortion-Trade-Off Formulierung ableiten. Darüber hinaus ermöglicht er eine quantitative Evaluation von Methoden zur Relevanzbewertung und zeigt, dass die neu vorgeschlagene Heuristik im Vergleich zu früheren Methoden die besten Ergebnisse bei der Identifizierung von relevanten Eingabegrößen erzielt.

Der zweite Teil untersucht die Genauigkeit und Robustheit von Deep Learning Methoden für die Rekonstruktion von Signalen aus unzureichend abgetasteten indirekten Messungen. Solche inversen Probleme treten zum Beispiel in der medizinischen Bildgebung, Geophysik, Nachrichtentechnik oder Astronomie auf. Während weit verbreitete klassische variationelle Lösungsmethoden (unter geeigneten Annahmen) Rekonstruktionsgarantien bieten, sind die zugrunde liegenden Mechanismen der datengestützten Methoden aus mathematischer Sicht meist nicht gut verstanden. Dennoch zeigen sie vielversprechende Ergebnisse und übertreffen empirisch häufig die klassischen Methoden in ihrer Rekonstruktionsqualität und -geschwindigkeit. Allerdings bestehen nach wie vor einige Zweifel an ihrer Zuverlässigkeit, insbesondere hinsichtlich ihrer Robustheit gegenüber Störungen der Eingaben. In der Tat ist bekannt, dass Klassifikatoren auf Basis von neuronalen Netzen anfällig gegenüber absichtlich herbeigeführten Störungen sind. Das heißt, dass winzige, visuell nicht wahrnehmbare, Veränderungen des Eingabesignals das neuronale Netz zu einer falschen Vorhersage verleiten können. Daher stellt sich die Frage, ob ähnliche Effekte auch im Zusammenhang mit der Signalrekonstruktion auftreten. Der wissenschaftliche Beitrag des zweiten Teils dieser Arbeit ist eine umfangreiche numerische Untersuchung der Ro-

bustheit von einer repräsentativen Auswahl von End-to-End Lösungsmethoden für inverse Probleme auf Basis von neuronalen Netzen. Es wird gezeigt, dass neuronale Netze für solche Regressionsproblemen (im Gegensatz zu den Klassifikationsproblemen) durchaus sehr robust gegenüber absichtlich herbeigeführten und auch unvermeidbaren statistischen Störungen sein können. Darüber hinaus können sie als State-of-the-Art angesehen werden und führen zu äußerst genauen Rekonstruktionen: Unter idealisierten Bedingungen mit synthetischen und störungsfreien Daten haben neuronale Netze das Potenzial, nahezu perfekte Rekonstruktionen zu erzielen, das heißt, ihr Rekonstruktionsfehler erreicht fast die numerische Maschinengenauigkeit.

Acknowledgments

Writing this thesis would not have been possible without the continued support and encouragement of a number of people, to whom I would like to express my sincere gratitude.

First of all, I would like to thank the members of the doctoral committee. In particular, Reinhard Heckel, Sebastian Pokutta, Lars Ruthotto, and Martin Skutella, for reviewing and evaluating my work, and Tobias Breiten for chairing the committee.

Furthermore, I want to express my gratitude to my further collaborators and co-authors, in particular, to Martin Genzel, Maximilian März, and Stephan Wäldchen. You were a great source of inspiration and it has been a privilege working with you during my time at the Technische Universität Berlin. I am glad for all the fruitful discussions and countless opportunities to learn from you and with you.

I would also like to thank my colleagues of the former "Applied Functional Analysis" group for the years of mutual support and the friendships that were formed during these years, especially, thanks to Martin Genzel, Ingo Gühring, Sandra Keiper, Maximilian März, Philipp Petersen, Mones Raslan, and Stephan Wäldchen. You have made working at TU Berlin so much more enjoyable. My former doctoral advisor Gitta Kutyniok I want to thank for the opportunities and freedoms that she gave me to work on an exciting research topic. I am especially indebted to Martin Skutella and Sebastian Pokutta for their support and guidance during the final stages of my doctoral studies.

Specifically with regard to the completion of this dissertation, I would like to thank Martin Genzel, Maximilian März, and Stephan Wäldchen for several discussions and comments that led to the improvement of this thesis, and Nicole Frost and Julian Kuhlmeier for proofreading parts of it and providing valuable feedback. As always in these cases, any errors or typos that remain are of course mine alone.

I am grateful for the support I received from the "Berlin Mathematical School" (BMS) and "Biophysical Quantitative Imaging Towards Clinical Diagnosis" (BIOQIC) graduate programs, both of which I was a part of during my doctoral studies.

On the personal side, I am grateful to my friend Jonathan Trantow, with whom I have shared an apartment during most of this time. Last but not least, my warmest thanks go to my family, especially to my parents and my sister, for their patience and endless support, and for accompanying me on my journey over all these years.

Contents

List of Figures

List of Tables

1 Introduction

Artificial intelligence methods based on deep learning consistently achieve outstanding performance in solving computational tasks arising in applications across various fields, ranging from image analysis [KSH12; STE13], to natural language processing [Cho+14; Vas+17], to medical diagnosis [SWS17; McB+18]. Previous "classical" methods for signal and image processing mostly rely on hand-crafted mathematical models that are specifically designed for a certain task. In contrast, the concept of deep learning is fitting generic universal function approximators, named artificial neural networks, to large sets of training data. This turn toward data-driven approaches was made possible by the immensely increasing availability of data and the computational resources to process it. Frequently, this led to a paradigm shift of the respective fields, with deep learning methods now constituting the state of the art.

Along with this success comes an almost equally extensive debate on the drawbacks of deep learning and concerns regarding its usage, especially in high-stakes applications, such as medical diagnosis or autonomous driving. These concerns are rooted in the fact that neural networks are still mostly considered opaque "black-box" models. It remains unclear how they operate internally, which raises doubts about their reliability, interpretability, and robustness.

In this work, we will consider two specific signal and image processing tasks: classification and reconstruction from indirect measurements. The focus is primarily on analyzing the interpretability and reliability of existing and commonly used types of neural networks and not the development of new state-of-the-art network architectures. The main questions that will be addressed are:

I. *How can the decisions of complex and learned classifiers be made more human interpretable? How difficult or computationally costly is it to achieve this goal? And how can the quality of different interpretability methods be evaluated and compared?*

II. *How reliable are data-driven and learned methods for the reconstruction of signals from indirect measurements? Can they compete with classical and provably robust reconstruction methods? In which situations is it possible to learn accurate and robust schemes and when is it not?*

Answering these questions constitutes the two main parts of this thesis. In the following, we give a more detailed introduction to both aspects.

1.1 Interpretability of Learned Classifier Functions

The task of classifying or labeling data arises in numerous applications, such as character recognition, voice recognition, medical diagnosis, autonomous driving, employee recruitment and hiring, or credit scoring.

Traditional machine learning methods for classification tasks include support vector machines [BGV92; CV95], decision trees [Bre+84; Qui86], naive Bayes classifiers [DH73; LIT92], k-nearest-neighbor classifiers [FH85], or linear classifiers in combination with handcrafted features that are first extracted from the data (e.g., Fisher vectors [PD07; PSM10] or histograms of oriented gradients (HOG) [McC86; Wil94]). In many cases, they allow for a straight-forward human interpretation of the model predictions.

Even though different versions of artificial neural networks were already introduced many decades ago [MP43; Iva71; Fuk80] they became largely popular around 2010, after achieving super-human performance at an image recognition contest for the first time and subsequently winning several other classification challenges [Cir+12; KSH12] against previous state-of-the-art methods. The continued success in the years that followed, up to the present day, led to the adoption of neural networks and deep learning as a core component of many modern machine learning and artificial intelligence systems.

In contrast to the traditional approaches, neural networks learn an *end-to-end* composition of a feature extractor and a final classifier in a combined way. They are universal function approximators [Hor91] and capable of hierarchical reasoning [Fuk80]. The expressiveness of these highly non-linear and parameter-rich classifiers makes understanding and interpreting their decisions challenging. The reasoning behind the decisions remains generally inaccessible. However, the ability to render these models less opaque by providing interpretable explanations of their predictions is essential for a reliable use of neural networks.

In an abstract sense, *interpretability* refers to the ability to *explain* the predictions made by a deep learning model (or more generally: a machine learning model) to humans in an *understandable* way [DK17]. Here, "explain" and "understandable" are rather vague terms and can mean different things depending on the context and application. In an effort to address this task from any possible angle, there has been a surge of research related to explainable artificial intelligence (XAI) and various attempts to categorize interpretability methods, see e.g., the recent surveys by [Fan+21; Cha+17; Gil+18; Zha+21] for an overview.

Zhang et al. propose a taxonomy to distinguish methods according to three main characteristics [Zha+21]:

i) *Passive* methods give post-hoc explanations for already trained classifiers while *active* methods require changing the classifier model already during training.

ii) The *type of explanation* (in increasing order of explanatory power) ranges from extracting prototypical examples, to feature attribution, to extracting hidden semantics, and finally extracting specific logical decision rules.

iii) *Local* methods explain predictions for individual data samples while *global* methods aim to explain classifier functions as a whole.

We focus on passive and local feature attribution, which is among the most widely used forms of explanations. In this case, the goal is to partition the input features of a classifier into *relevant* and *non-relevant* ones or to alternatively assign scores to each input feature of a data sample indicating its *relevance* for a prediction, as illustrated in Figure 1.1. Feature attribution methods are particularly popular in the context of image classification, where the scores are visualized as so-called *heatmaps* or *relevance maps*. Recent years have seen progress on this front with the introduction of multiple explanation methods for deep

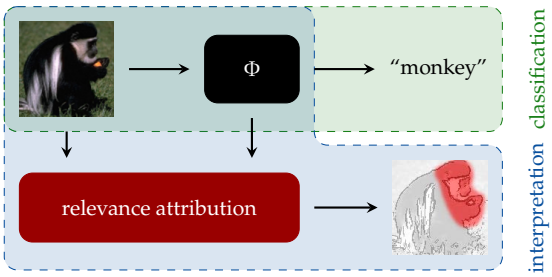


Figure 1.1: **Illustration of Relevance Attribution.** Local attribution methods aim at rendering decisions of a black-box classifier Φ more interpretable by providing heatmaps of the input features that contribute most to an individual prediction.

neural networks [Bac+15; LL17; RSG16; SGK17; SVZ13; ZF14]. Most commonly, the algorithms that produce these maps are motivated by heuristic arguments, yet there is no agreed upon formal notion of relevance. Instead of answering a precise question, these maps are mostly compared to human intuition about what part of the input variables should be of importance. Notable exceptions are explanations based on Shapley values [Sha53] that are required to satisfy certain game theoretic properties.

Nevertheless, relevance maps have been compared numerically using, e.g., pixel-flipping [Sam+17] and input perturbations [FV17]. This points us toward what practitioners understand as "relevant" and what information they expect to be conveyed by relevance maps. The common criterion for relevance that we identified can be summarized by the following question.

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

A more quantitative version of the question is the following.

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

A reasonable explanation algorithm should provide sufficient information to answer these questions.

Contribution

The first part of the thesis introduces a rigorous formulation of the concept of relevance in the form of probabilistic prime implicants. First, this is done for binary classifier functions on discrete domains. Second, it is extended to continuous classifiers on continuous domains, including the important example of neural networks. In both cases we provide a computational complexity analysis for the problem of finding small relevant sets of variables.

In the discrete setting we show that the problem is complete for the complexity class $\mathsf{NP}^{\mathsf{PP}}$, which includes many important problems arising in artificial intelligence research, such as probabilistic conformant planning [LGM98], calculating maximum expected utility (MEU) solutions [CJ08], and maximum a posteriori (MAP) hypotheses [Par02]. A further analysis of the relevance mapping problem (and several variations of it) reveals that even approximating solutions remains NP-hard for any non-trivial approximation factor.

In the continuous setting, the problem can be rephrased 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. Answering the questions *Q1* and *Q2* in this context amounts to evaluating a rate-distortion function. We analyze the computational complexity of this task and see that the hardness result from the discrete setting carries over to the continuous setting. Hence, even approximating the rate-distortion function is generally a computationally hard problem:

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

This has important implications for the design and evaluation of relevance mapping algorithms. Since no computationally feasible algorithm can be proven to succeed, we have to rely on heuristic motivations and numerically evaluate the resulting relevance maps in relation to our questions.

We propose our own heuristic, called RDE (Rate-Distortion Explanation). It combines a problem relaxation that uses continuous relevance scores instead of a hard partition into relevant and non-relevant variables with the rate-distortion formalism, and thus precisely aims at answering the questions *Q1* and *Q2*. The heuristic relies on the approximate propagation of random variables through neural network functions. We show the necessity to use approximations for this step for (almost) all probability distributions.

Finally, we discuss existing techniques for the evaluation of relevance mapping methods and propose two new evaluation tests. The first is a benchmark test based on a classification task with synthetic data and an explicitly constructed, hence transparent, classifier. The second is a numerical post-hoc evaluation on real-world data based on "pixel-flipping" [Sam+17] but adapted to our rate-distortion framework. We compare RDE to various existing relevance mapping methods.

In summary, the main contributions of the first part are three-fold:

1. We formalize the relevance mapping problem for classifiers on discrete as well as continuous domains and analyze its computational complexity.
2. We propose a novel heuristic for obtaining relevance maps for neural network classifiers that directly aims to answer the questions *Q1* and *Q2*.
3. We compare our proposed and several established heuristics with respect to existing and two newly introduced evaluation methods and benchmark tests.

1.2 Accuracy and Robustness of Learned Reconstruction Methods

The reconstruction of signals from indirect measurements is an important task arising in applications such as medical imaging [Lus+08], geophysics [TV81], communication [Hau+08], or astronomy [SPM02]. Such tasks are typically formulated as an inverse problem. In its prototypical finite dimensional form it reads as follows:

$$\left\{\begin{array}{c}\textit{Given a linear forward operator } \mathbf{A} \in \mathbb{R}^{m\times n} \\ \textit{and corrupted measurements } \mathbf{y} = \mathbf{A}\mathbf{x}_0 + \mathbf{e} \\ \textit{with } \|\mathbf{e}\|_2 \leq \eta, \textit{ reconstruct the signal } \mathbf{x}_0.\end{array}\right\} \tag{IP}$$

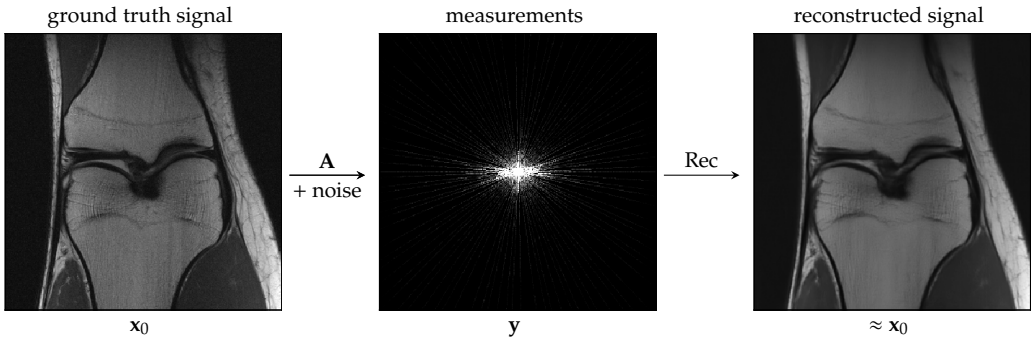


Figure 1.2: **Illustration of Signal Reconstruction.** Example of an inverse problem arising in magnetic resonance imaging (MRI). The goal is to reconstruct the ground truth signal from indirect and incomplete measurements (subsampled Fourier measurements in the case of MRI). The data is taken from the NYU fastMRI knee dataset [Zbo+18; Kno+20b], see Chapter 6 for more details.

This is illustrated in Figure 1.2. The forward operator models a physical measurement process and $\mathbf{e}$ is the measurement noise. The measurement process is often time-consuming, costly, or potentially harmful (e.g., exposure to x-ray radiation in computed tomography). Hence, it is desirable to take as few measurements as possible. Therefore, the underdetermined regime where $m \ll n$ is of utmost importance and has gained much attention in the last decades. In this regime the inverse problem is "ill-posed" and has no unique solution unless further restrictions (e.g., prior knowledge about $\mathbf{x}_0$) are imposed. Restrictions on $\mathbf{x}_0$ can either be modeled explicitly by introducing handcrafted priors and regularization methods or they can be inferred implicitly from sampled data.

The error of any given reconstruction method $\mathrm{Rec}\colon \mathbb{R}^m \to \mathbb{R}^n$ for (IP) can be decomposed as

$$\|\mathbf{x}_0 - \mathrm{Rec}(\mathbf{y})\|_2 \leq \underbrace{\|\mathbf{x}_0 - \mathrm{Rec}(\mathbf{A}\mathbf{x}_0)\|_2}_{\text{(a)}} + \underbrace{\|\mathrm{Rec}(\mathbf{A}\mathbf{x}_0) - \mathrm{Rec}(\mathbf{y})\|_2}_{\text{(b)}}. \tag{1.1}$$

The first term (a) is associated to the *accuracy* of the solution map and measures how well $\mathbf{x}_0$ can be estimated in the idealistic setting of noiseless measurements. The ubiquitous presence of noise makes it indispensable that a reconstruction method has to be robust against additive perturbations $\mathbf{e}$. This *robustness* is captured by the second term (b).

Among the handcrafted priors and regularization methods, the notion of sparsity plays an important role: under the assumption that $\mathbf{x}_0$ is sparse or can be sparsely represented in some appropriate basis or dictionary, the methodology of *compressed sensing* has proven that accurate and robust reconstruction from incomplete measurements is still possible [FR13]. In this case the solution map satisfies an error bound of the form

$$\|\mathbf{x}_0 - \mathrm{Rec}(\mathbf{y})\|_2 \leq C \cdot \eta \tag{1.2}$$

for some constant $C > 0$. In other words, the reconstruction error scales linearly with the strength of the noise corrupting the measurements.[1] Although state of the art in various real-world applications, the practicability of the associated reconstruction algorithms

[1] In particular, perfect recovery can be achieved in the case of noiseless measurements, i.e., the first term (a) in (1.1) must be arbitrarily small.

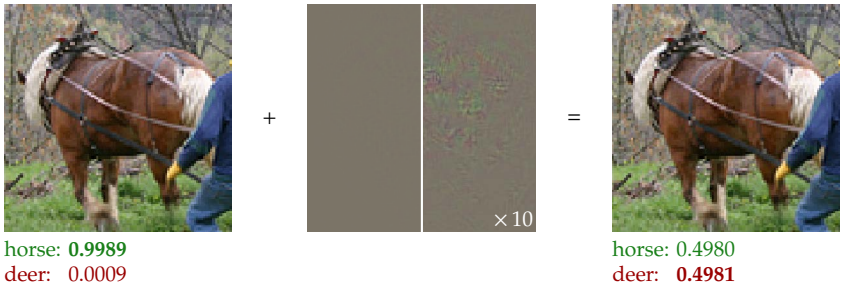


Figure 1.3: **Illustration of Adversarial Perturbations.** Example of an adversarial perturbation for a VGG-16 [SZ14] based neural network trained on the STL-10 [CNL11] dataset (see Chapter 4 for details). Adding a small visually imperceptible perturbation to a correctly classified *horse* image changes the prediction to *deer*. The softmax scores for both classes before and after adding the perturbation are shown below each respective image. For visual purposes the perturbation is shown scaled by a factor ten in its right half.

is often limited by computational costs, the need for manual parameter tuning, and a mismatch between the chosen sparsity model and the data.

Following the recent success of artificial intelligence in computer vision [KSH12; LBH15; GBC16], there has been a considerable effort to solve the inverse problem (IP) using deep learning approaches, e.g., see [GL10; Yan+16; Don+16; KMY17; Jin+17; Ham+18; Che+17a; Bor+17; Zhu+18; Bub+19] and [Arr+19] for a recent survey. Considering the inverse problem in the context of statistical leaning problems, we aim at inferring a solution map by fitting a neural network to sampled data $\{(\mathbf{y}^i, \mathbf{x}_0^i)\}_{i=1}^M$ in a supervised learning procedure. Here the assumption on $\mathbf{x}_0$ is rather implicit: it should be drawn from the same distribution as $\{(\mathbf{y}^i, \mathbf{x}_0^i)\}_{i=1}^M$.

It is fair to say that data-driven approaches can significantly outperform classical methods in terms of reconstruction accuracy and speed in many situations. On the other hand, one may argue that the underlying mechanisms of the trained neural networks remain largely unclear [Ela17]. In the absence of theoretical guarantees of the form (1.2), an empirical verification of their accuracy and robustness against measurement noise is crucial.

Recently, Sidky et al. pointed out a lack of evidence for the reliability and accuracy of deep-learning-based solution strategies and demonstrated that a simple post-processing approach with the prominent U-Net architecture [RFB15] may not yield satisfactory recovery precision in sparse-view computed tomography (CT) [Sid+21b].

Similarly, while a number of works report a remarkable resilience against noise [Che+17b; Zhu+18; Hau+20], several alarming findings indicate that deep-learning-based reconstruction schemes are typically unstable [Hua+18; Ant+20; Got+20; RBL20]. In particular, the recent study of Antun et al. suggests that deep learning for inverse problems comes at the cost of instabilities, in the sense that „[...] certain tiny, almost undetectable perturbations, both in the image and sampling domain, may result in severe artifacts in the reconstruction [...]“ [Ant+20]. In machine learning research on classification, such a sensitivity of neural networks is a well-established phenomenon. Initiated by Szegedy et al. [Sze+14], a substantial body of literature is devoted to *adversarial attacks* (and their defenses), i.e., the computation of a visually imperceptible change to the input that fools the network, as shown in Figure 1.3. Typically, an “attacker” exploits gradient-based information in

order to cross the discontinuous decision boundary of a classifier. This can be a serious issue for sensitive applications where wrong predictions impose a security risk—imagine a misclassified stop sign in autonomous driving [KGB17; Eyk+18].

Despite these findings, it appears peculiar that solving inverse problems by deep-learning-based schemes might become unstable. Learning a reconstruction algorithm can be seen as a regression task, where measurements are mapped to a high-dimensional signal manifold (e.g., medical images). In contrast, a neural network classifier maps to a low-dimensional, discrete output domain, resulting in a "vulnerable" decision boundary. Moreover, it is well known that robust and accurate algorithms exist for many inverse problems. Since these are often used as templates for network architectures, it seems surprising that the latter should suffer from severe instabilities.

Clearly, the accuracy and robustness against noise is quintessential for an application of deep learning in practice, especially in sensitive fields such as biomedical imaging. Therefore, a profound study of this topic is indispensable.

Contribution

The second part of this thesis is dedicated to a comprehensive numerical study of the accuracy and robustness of neural-network-based methods for solving underdetermined inverse problems.

We consider the aspect of accuracy in the context of the "AAPM Deep Learning Sparse-View Computed Tomography" grand challenge that was initiated following the findings of Sidky et al. with the goal of identifying „[. . .] the state-of-the-art in solving the CT inverse problem with data-driven techniques" [Sid+21a]. We show that data-driven approaches can achieve near-exact (close to numerical precision) recovery in the noiseless case.

The primary objective of our experiments regarding the aspect of robustness is to analyze how much the reconstruction error grows with the noise level η. We investigate this relationship in terms of statistical and adversarial noise: the former means that measurement noise is drawn from an appropriate probability distribution, while the latter explores worst-case perturbations that maximize the reconstruction error for a fixed η. Similar to adversarial attacks in classification, computing worst-case noise is based on a non-convex formulation that is addressed by automatic differentiation and a gradient descent scheme.

In the absence of an empirical certificate of robustness, a central and distinctive component of our analysis is the systematic comparison with a classical benchmark method with provable guarantees, namely total-variation (TV) minimization. In this case, evaluating the gradient is non-trivial and carried out by unrolling the underlying optimization problem.

Our experiments consider several prototypical inverse problems as use cases. This includes classical compressed sensing with Gaussian measurements as well as the reconstruction of phantom images from Radon and Fourier measurements. Furthermore, a real-world scenario for magnetic resonance imaging (MRI) is investigated, based on the NYU-fastMRI dataset [Zbo+18; Kno+20b]. We examine a representative selection of learned reconstruction architectures, reaching from simple post-processing networks to iterative schemes. In total, this presents a robustness analysis of more than 25 neural networks, each of them trained in-house with publicly available code.

The main findings of the second part may be summarized as follows:

1. It is possible to achieve near-prefect accuracy using end-to-end neural networks for underdetermined inverse problems (as exemplarily demonstrated for the AAPM Sparse-View CT task).

2. In every considered scenario, we find deep-learning-based methods that are at least as robust as TV minimization with respect to adversarial noise. This does not require sophisticated architectures or defense strategies.

3. All trained neural networks are remarkably robust against statistical noise. Although TV minimization may yield exact recovery for noiseless measurements, it is still outperformed by learned methods in mid- to high-noise regimes.

4. The reconstruction performance is affected by the underlying network architecture. For instance, promoting data consistency in iterative schemes may improve both accuracy and robustness.

5. One should not commit the "inverse crime" of training a neural network with *noiseless* data, which may cause an unstable behavior for higher noise levels. We demonstrate that simply adding white Gaussian noise to the training measurements is an effective remedy—a regularization technique that is commonly known as *jittering* in machine learning research. This adaption has a virtually imperceptible impact on the in-distribution accuracy, but might affect out-of-distribution features.

Apart from these observations, our work is, to the best of our knowledge, the first to empirically characterize the performance gap between adversarial and statistical noise in the context of (IP). In particular, this gap is not exclusive to deep-learning-based schemes but also appears for classical methods such as TV minimization. Our central conclusion is:

> The existence of adversarial examples in classification tasks does not naturally carry over to neural-network-based solvers for inverse problems. Such reconstruction schemes may not only supersede classical methods as state of the art, but they can also exhibit a similar degree of robustness.

Since our study as it is has required massive computational resources, some aspects have to remain unexplored. In particular, given the sheer number of network architectures, we explicitly do *not* claim that every deep-learning-based method is stable. Nevertheless, our findings suggest that fairly standard workflows allow for surprisingly robust reconstruction schemes. This offers an alternative and novel perspective on the reliability of deep learning strategies in inverse problems. Therefore, we believe that the present work takes an important step toward their safe use in practice.

1.3 Previous Publication of the Results in this Thesis

The results presented in this thesis have previously been published by the author and his collaborators or have been submitted for publication and are currently under review. All publications and preprints on which the content of this thesis is based are listed below:

[Wäl+21] S. Wäldchen, J. Macdonald, S. Hauch, G. Kutyniok,
"The Computational Complexity of Understanding Binary Classifier Decisions",
Journal of Artificial Intelligence Research (70), pp. 351–387, 2021.

This work introduces the problem of determining small sets of relevant variables for discrete classifiers (Boolean circuits) and provides an analysis of its computational complexity. This is incorporated in Chapter 3.

[Mac+19] J. Macdonald, S. Wäldchen, S. Hauch, G. Kutyniok,
"A Rate-Distortion Framework for Explaining Neural Network Decisions",
Preprint [arXiv: 1905.11092], 2019.

This work introduces the problem of determining relevance scores for the variables of continuous classifiers (ReLU neural networks) and proposes a heuristic solution strategy. This is incorporated in Chapter 4, in particular Sections 4.3 and 4.4.

[Mac+20] J. Macdonald, S. Wäldchen, S. Hauch, G. Kutyniok,
"Explaining Neural Network Decisions Is Hard",
Presented at *XXAI Workshop, 37th International Conference on Machine Learning (ICML)*, 2020.

This work extends the problem of determining small sets of relevant variables from discrete (Boolean circuits) to continuous classifiers (ReLU neural networks) and provides an analysis of its computational complexity. This is incorporated in Chapter 4, in particular Sections 4.1 and 4.2. Together with [MBP21] this work also forms the basis for parts of the introduction relating to interpretable classification, in particular in Section 1.1.

[MBP21] J. Macdonald, M. Besançon, S. Pokutta,
"Interpretable Neural Networks with Frank-Wolfe: Sparse Relevance Maps and Relevance Orderings",
Preprint [arXiv: 2110.08105], 2021.
Accepted for publication in the *Proceedings of the 39th International Conference on Machine Learning (ICML)*.

This work extends the heuristic solution strategy for determining relevance scores for the variables of continuous classifiers (ReLU neural networks). It proposes a constrained optimization formulation and evaluates different algorithmic approaches for solving it. Additionally, a variation based on relevance orderings instead of scores is introduced. This is incorporated in Chapter 4, in particular Sections 4.3 and 4.4. Together with [Mac+20] this

work also forms the basis for parts of the introduction relating to interpretable classification, in particular in Section 1.1.

[MW22] J. Macdonald, S. Wäldchen,
"A Complete Characterisation of ReLU-Invariant Distributions",
Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) (151), pp. 1457–1484, 2022.

This work justifies an approximation step in the above heuristic solution strategies by introducing the concept of ReLU-invariant families of probability distributions and showing that these can only exist in a few degenerate cases. This is incorporated in Chapter 5.

[GMM20] M. Genzel, J. Macdonald, M. März,
"Solving Inverse Problems With Deep Neural Networks – Robustness Included?",
Preprint [arXiv: 2011.04268], 2020.
Accepted for publication in *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

This work provides an extensive empirical study regarding the robustness of deep-learning-based reconstruction methods for ill-posed inverse problems. This is incorporated in Chapter 6. Together with [GMM21] this work also forms the basis for parts of the introduction relating to robust and accurate reconstructions, in particular in Section 1.2.

[GMM21] M. Genzel, J. Macdonald, M. März,
"AAPM DL-Sparse-View CT Challenge Submission Report: Designing an Iterative Network for Fanbeam-CT with Unknown Geometry",
Preprint [arXiv: 2106.00280], 2021.

This work shows that deep-learning-based reconstruction methods can achieve near-exact recovery for an exemplary ill-posed inverse problem in an idealized situation (with synthetic data and noiseless measurements). This is incorporated in Chapter 7. Together with [GMM20] this work also forms the basis for parts of the introduction relating to robust and accurate reconstructions, in particular in Section 1.2.

The author of this thesis is a main contributor to all of these works. He is jointly responsible for their presentation and was involved in the development of all mathematical results (including their proofs) as well as in the design and execution of the experiments as well as the evaluation and interpretation of the experimental results.

1.4 Availability of Code

The code for the numerical experiments in this thesis is publicly available.

1. **Rate-Distortion Explanations (RDE)**
   https://github.com/jmaces/rde
   This repository provides the code for computing RDE relevance scores. It forms the basis for all experiments in Chapter 4 that do *not* involve Frank-Wolfe algorithms.

2. **Frank-Wolfe for RDE**
   https://github.com/ZIB-IOL/fw-rde
   This repository provides the code for solving the constrained optimization formulation of computing RDE relevance scores and orderings with Frank-Wolfe algorithms. It forms the basis for all experiments in Chapter 4 that *do* involve Frank-Wolfe algorithms.

3. **Assumed Density Filtering (ADF)**
   https://github.com/jmaces/keras-adf
   This repository provides a `Tensorflow` [Mar+15]/`Keras` [Cho+15] implementation of Assumed Density Filtering (ADF) for several types of common neural network layers. It is used as a building block for the experiments in Chapter 4.

4. **Statistics Utilities**
   https://github.com/jmaces/statstream
   This repository provides utility code for computing various statistics from streaming data. It is used as a building block for the experiments in Chapter 4 and described in more detail in Appendix B.7.

5. **Robust Neural Networks**
   https://github.com/jmaces/robust-nets
   This repository provides the code for our analysis of the robustness of deep-learning-based approaches to solving inverse problems. It forms the basis for all experiments in Chapter 6.

6. **AAPM Sparse-View CT Challenge**
   https://github.com/jmaces/aapm-ct-challenge
   This repository provides the code for our winning submission to the AAPM Sparse-View CT Challenge. It forms the basis for all experiments in Chapter 7.

The author of this thesis is a main contributor to all of these code repositories. In addition, we rely on the following external software packages and toolboxes.

1. **Tensorflow/Keras** [Mar+15; Cho+15]
   https://github.com/tensorflow/tensorflow
   https://github.com/keras-team/keras
   These repositories provide tools for automatic differentiation and for building and training neural networks. We use `Tensorflow/Keras` as our main computational framework for the experiments in Chapter 4.

2. **PyTorch** [Pas+17]
   https://github.com/pytorch/pytorch
   This repository provides tools for automatic differentiation and for building and training neural networks. We use `PyTorch` as our main computational framework for the experiments in Chapters 6 and 7.

3. **iNNvestigate** [Alb+18]
   https://github.com/albermax/innvestigate
   This repository provides implementations of multiple relevance mapping methods. It is used for most comparison baselines in the experiments in Chapter 4.

4. **SHAP** [LL17]
   https://github.com/slundberg/shap
   This repository provides an implementation of the Shapley Additive Explanation (SHAP) relevance mapping method. It is used for the `SHAP` comparison baselines in the experiments in Chapter 4.

5. **LIME** [RSG16]
   https://github.com/marcotcr/lime
   This repository provides an implementation of the Local Interpretable Model-Agnostic Explanations (LIME) relevance mapping method. It is used for the `LIME` comparison baselines in the experiments in Chapter 4.

6. **FrankWolfe.jl** [BCP21]
   https://github.com/ZIB-IOL/FrankWolfe.jl
   This repository provides a `Julia` implementation of various variants of Frank-Wolfe and Conditional Gradient algorithms for constrained optimization. It is used as a building block for the Frank-Wolfe experiments in Chapter 4.

2

Preliminaries and Notation

This chapter introduces general notations and notational conventions as well as definitions that are required throughout the thesis. A lot of them are quite standard, so we will keep this part short and refrain from providing lengthy formal definitions.

Vectors and Matrices. While vectors and matrices are denoted by lower- and uppercase boldface symbols respectively, their components are correspondingly denoted as lowercase symbols with subscript indices, e.g., $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ and $\mathbf{A} = \left[a_{ij}\right] \in \mathbb{R}^{m\times n}$ with $m, n \in \mathbb{N}$. The identity matrix is $\mathbf{I}_{n\times n} \in \mathbb{R}^{n\times n}$ and the n-dimensional vectors of all zeros or ones are denoted $\mathbf{0}_n$ and $\mathbf{1}_n$ respectively. The standard basis vectors of $\mathbb{R}^n$ are $\mathbf{e}_1, \dots, \mathbf{e}_n$. Further, $\mathbf{A}^\top \in \mathbb{R}^{n\times m}$ and $\mathbf{A}^\dagger \in \mathbb{R}^{m\times n}$ are the transpose[1] and the Moore-Penrose pseudo-inverse of $\mathbf{A}$ respectively.

We set $[n] = \{1, \dots, n\}$ and for a vector $\mathbf{x} \in \mathbb{R}^n$ and a subset $S \subseteq [n]$ we denote by $\mathbf{x}_S$ the restriction of $\mathbf{x}$ to components indexed by S, i.e., $\mathbf{x}_S = (x_j)_{j\in S} \in \mathbb{R}^{|S|}$. The set complement of S is $S^c = [n] \setminus S$.

We denote the diagonal matrix with entries given by $\mathbf{x}$ as $\operatorname{diag}(\mathbf{x})$ and use $\odot$ (resp. $\oslash$) for the component-wise Hadamard product (resp. quotient) of vectors or matrices of the same dimension. We write $\mathbf{x}^2 = \mathbf{x} \odot \mathbf{x}$ for brevity and consider univariate functions applied to vectors or matrices to act component-wise.

Spaces and Norms. Unless stated otherwise, we are working in Euclidean $\mathbb{R}^n$ with the standard scalar product $\langle \mathbf{x}, \mathbf{y}\rangle = \mathbf{x}^\top \mathbf{y}$ and induced norm $\|\mathbf{x}\|_2 = \sqrt{\langle \mathbf{x}, \mathbf{x}\rangle}$. Similarly, $\|\mathbf{x}\|_p$ denotes the usual ℓ_p-norm for any $p \in [1, \infty]$.

The space $\mathbb{R}^{m\times n}$ of matrices is equipped with the scalar product $\langle \mathbf{A}, \mathbf{B}\rangle = \operatorname{tr}(\mathbf{A}^\top \mathbf{B})$ where $\operatorname{tr}(\cdot)$ denotes the trace of a square matrix. For a subset $U \subseteq \mathbb{R}^n$ we denote the linear subspace spanned by U as $\operatorname{span}(U)$, the affine subspace spanned by U as $\operatorname{aff}(U)$, and the convex hull of U as $\operatorname{conv}(U)$.

Probability Distributions and Random Variables. We use the notation $\mathcal{B}(\mathbb{R}^n)$ for the Borel σ-Algebra on $\mathbb{R}^n$ and $\mathcal{D}(\mathbb{R}^n)$ for the set of Radon[2] Borel probability measures on $\mathbb{R}^n$. Given a measure $\mu \in \mathcal{D}(\mathbb{R}^n)$ and a measurable function $f : \mathbb{R}^n \to \mathbb{R}^n$ the pushforward of μ with respect to f is denoted $f_*\mu$ and defined as $f_*\mu(B) = \mu(f^{-1}(B))$ for $B \in \mathcal{B}(\mathbb{R}^n)$.

The uniform distribution on a domain $\Omega \subseteq \mathbb{R}^n$ is $\mathcal{U}(\Omega)$. Most important for us are $\mathcal{U}(\{0,1\}^n)$ and $\mathcal{U}([0,1]^n)$, that is the uniform distributions on $\{0,1\}^n$ and $[0,1]^n$

[1] For a complex matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ we denote the conjugate transpose as $\mathbf{A}^* \in \mathbb{C}^{m\times n}$. This will only arise in the context of Fourier measurements in Chapter 6. Everywhere else we consider real matrices and vectors.

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

respectively. The normal distribution with mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ is notated as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Finally, $\mathbb{P}[\ldots]$, $\mathbb{E}[\ldots]$, and $\mathbb{V}[\ldots]$ refer to the probability of some event, and the expected value and (co-)variance of a random variable (or random vector) respectively.

Boolean Functions and Circuits. Throughout, $\Psi\colon \{0,1\}^n \to \{0,1\}$ denotes an n-ary Boolean function. In contrast, functions on continuous domains, such as neural networks, will be denoted Φ (see below). We refer to Ψ as a Boolean circuit when its description is in terms of standard logical gates like And, Or, and Not. The description length is the number of gates.

We use the usual symbols $\wedge$, $\vee$, $\neg$, and $\oplus$ for the logical conjunction, disjunction, negation, and exclusive disjunction respectively. Further, we will use Boolean functions also interchangeably as logical propositions, in the sense that $\Psi(\mathbf{x})$ is shorthand for the logical proposition $\Psi(\mathbf{x}) = 1$. Whenever we discuss statements concerning probabilities of logical propositions to hold, we assume independent uniform distributions for all involved variables. Thus,

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

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

$$\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z}) \mid A(\mathbf{z})] = \mathbb{P}_{\mathbf{z} \sim \mathcal{U}(\{0,1\}^n)}[\Psi(\mathbf{z}) \mid A(\mathbf{z})] = \frac{|\{\, \mathbf{z} \in \{0,1\}^n \,:\, \Psi(\mathbf{z}) = 1, A(\mathbf{z}) = 1 \,\}|}{|\{\, \mathbf{z} \in \{0,1\}^n \,:\, A(\mathbf{z}) = 1 \,\}|}.$$

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 $\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z})]$.

Neural Networks. There are countless different variations of neural network architectures, however for the most part we will only require a precise notation for sets of simple feed-forward neural networks. In their basic form these consist of alternating affine linear maps and simple component-wise non-linear functions, called activation functions. Both the domain and co-domain are assumed to be (subsets of) finite dimensional Euclidean spaces.

Definition 2.1. Let $L \in \mathbb{N}$ and $n_0, \ldots, n_L \in \mathbb{N}$. Then

$$\begin{aligned}\mathcal{NN}_{L,\varrho,\sigma}^{n_0,\ldots,n_L} = \big\{ \Phi\colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_L} \,:\, & \Phi = \Phi_L \circ \cdots \circ \Phi_1, \\ & \Phi_j(\mathbf{x}) = \varrho(\mathbf{W}_j \mathbf{x} + \mathbf{b}_j), \mathbf{W}_j \in \mathbb{R}^{n_j \times n_{j-1}}, \mathbf{b}_j \in \mathbb{R}^{n_j} \text{ for } j = 1, \ldots, L-1, \\ & \Phi_L(\mathbf{x}) = \sigma(\mathbf{W}_L \mathbf{x} + \mathbf{b}_L), \mathbf{W}_L \in \mathbb{R}^{n_L \times n_{L-1}}, \mathbf{b}_L \in \mathbb{R}^{n_L} \big\}\end{aligned}$$

denotes the set of *neural networks* with L layers of widths $(n_0, \ldots, n_L)$ and with activation functions $\varrho\colon \mathbb{R} \to \mathbb{R}$ and $\sigma\colon \mathbb{R} \to \mathbb{R}$.

Each Φ_j is referred to as a network *layer* and L as the *depth* of the networks. The matrix $\mathbf{W}_j$ and vector $\mathbf{b}_j$ are called the *weight* matrix and *bias* vector of the j-th layer respectively. The

collection of all weights and biases are the free parameters of the networks that need to be optimized when using a neural network in a supervised empirical risk minimization problem. They are also sometimes collectively only referred to as the weights of the network. 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]. Allowing a different activation σ in the last layer is often useful, especially in classification problems (here the softmax function is frequently used [Bri89; Bri90]). For the special case of networks with only one activation function and equally wide layers we introduce the abbreviated notation $\mathcal{NN}^{n}_{L,\varrho} = \mathcal{NN}^{n,\dots,n}_{L,\varrho,\varrho}$. For the set of all neural networks with specified input and output dimensions but arbitrary (finite) depth and intermediate layer widths we denote $\mathcal{NN}^{n_{\text{in}},*,n_{\text{out}}}_{*,\varrho,\sigma} = \bigcup_{L\in\mathbb{N}} \bigcup_{n_1,\dots,n_{L-1}\in\mathbb{N}} \mathcal{NN}^{n_{\text{in}},n_1,\dots,n_{L-1},n_{\text{out}}}_{L,\varrho,\sigma}$.

Besides the simple affine linear (also called fully-connected) layers, many other types of network layers have been proposed: convolutional layers [Fuk80; LeC+89] enforce a sparsity pattern on the weight matrix so that it represents a convolution operation. This drastically reduces the number of free parameters and is frequently used in large networks (especially for image signals). Pooling layers [Fuk80; Yam+90] are alternative non-linear maps that do not act component-wise but aggregate information over small neighborhoods of components. This reduces the width of subsequent layers and thus also reduces the number of free parameters. Other commonly used building-blocks in deep learning architectures are normalization layers, such as batch-normalization [IS15] or group-normalization [WH18], skip connections that link non-adjacent layers, or probabilistic layers such as dropout [Sri+14]. A more detailed description of these operations is not necessary at this point. For a comprehensive overview of neural networks the interested reader is referred to [GBC16; Nie18].

I

Classification Tasks

3

Interpretability of Boolean Circuit Classifiers

Even though ultimately our goal is to analyze the interpretability of the immensely important class of (ReLU) neural network classifiers, we begin our analysis for Boolean function classifiers instead. For this, we extend the concept of prime implicants [Mar91; Mar00] for Boolean functions to a probabilistic setting, which formalizes existing practical attempts to interpret neural network classifiers. Neural networks can be seen as continuous generalizations of Boolean circuits, as visualized in Figure 3.1, and the extension of the results of this chapter to neural networks is discussed in Chapter 4.

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 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].

In this chapter we connect these probabilistic reasoning tasks to a key problem in machine learning, namely the problem of interpreting the decisions of classifier functions. A significant first step toward understanding classifier decisions is to distinguish the relevant input variables from the less relevant ones for a specific prediction, as illustrated in Figure 3.2.

A stringent logical concept that captures the idea of relevance for Boolean circuits are prime implicant explanations [SCD18]. These consist of subsets of the input variables that, if held fixed, guarantee that the function value remains unchanged, independent of the assignment to the rest of the variables. The problem of finding small prime implicants is $\mathsf{NP}^{\mathsf{coNP}}$-hard [EG95], and practical algorithms rely on highly optimized Sat or MILP-solvers even for relatively low-dimensional cases [INM19]. In general, since the classifier function is considered fixed within each problem instance, such complexity results carry over to all types of classifiers that are able to efficiently represent Boolean circuits. As briefly mentioned above, this includes the prominent example of ReLU neural networks.

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.

In this work, 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 feasible for binary classifiers of a few variables, but it is too rigid for very high dimensional cases, such as modern image classification. In many cases, it would lead to unnecessarily large sets of relevant variables that are not able to pinpoint the true importance of variables. This is explained in more

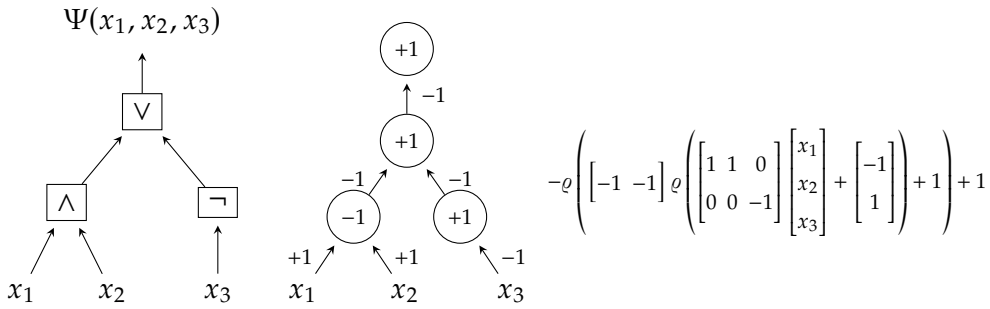


Figure 3.1: **Boolean Circuits and Neural Networks.** The ternary Boolean function defined as $\Psi(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (\neg x_3)$ viewed as a Boolean circuit (a) and a rectified linear unit (ReLU) neural network in its graphical (b) and algebraic representation (c). The neural network weights and biases are denoted at the edges and nodes respectively. The ReLU activation $\varrho(x) = \max\{x, 0\}$ is applied component-wise.

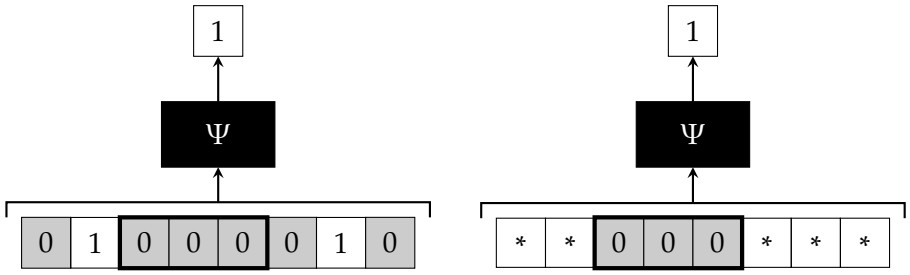


Figure 3.2: **Illustration of Relevant Sets.** The Boolean function $\Psi\colon \{0,1\}^8 \to \{0,1\}$ decides if a binary input string contains a substring of three consecutive zeros. A relevant subset of input variables for an exemplary input is highlighted by a box. In this simple case the three consecutive 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.

detail in Section 3.1.1. 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; ZF14], 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. Thus, practical interpretation algorithms necessarily need to solve the problem defined in the next section and are subject to our hardness results. A rigorous analysis of this setting is long overdue and of high importance.

3.1 The δ-Relevant-Input Problem

Let us now give a formal definition of our probabilistic notion of prime implicant explanations and state the two main results of this chapter.

For this let Ψ be an n-ary Boolean function and $\mathbf{x} \in \{0,1\}^n$. A subset $S \subseteq [n]$ of variables is *relevant* for the function value $\Psi(\mathbf{x})$ if fixing $\mathbf{x}$ on S and randomizing 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.

Definition 3.1. Let $\Psi\colon \{0,1\}^n \to \{0,1\}$, $\mathbf{x} \in \{0,1\}^n$, and $\delta \in [0,1]$. We call $S \subseteq [n]$ a *δ-relevant* set for Ψ and $\mathbf{x}$, if $\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z}) = \Psi(\mathbf{x}) \mid \mathbf{z}_S = \mathbf{x}_S] \geq \delta$.

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.

Remark 3.2. For $0 \leq \delta_1 \leq \delta_2 \leq 1$, any δ_2-relevant set is also δ_1-relevant. Further, any $S \subseteq [n]$ is always zero-relevant and the set $S = [n]$ is always one-relevant. In particular $[n]$ is δ-relevant for all $\delta \in [0,1]$.

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.

Definition 3.3. For $\delta \in (0,1]$ we define the δ-Relevant-Input problem as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^n \to \{0,1\}$, a signal $\mathbf{x} \in \{0,1\}^n$, and $k \in [n]$.

Decide: Does there exist $S \subseteq [n]$ 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 for different values of k. We show that the problem is hard for any fixed δ. The minimization formulation of the above decision problem can be defined in the obvious way.

Definition 3.4. For $\delta \in (0,1]$ we define the Min-δ-Relevant-Input problem as follows.

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

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

The majority of the remainder of the chapter will deal with analyzing the computational complexity of δ-Relevant-Input, Min-δ-Relevant-Input, and related variants thereof. The first main contribution of this chapter shows that the δ-Relevant-Input problem is generally hard to solve.

Theorem 3.5. *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 $S \subseteq [n]$ 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

[1] Checking if a subset is one-relevant is in coNP instead of PP. Thus, we excluded $\delta = 1$ in Theorem 3.5.

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 3.3. We will see that for positive γ this reduces the problem complexity from $\mathsf{NP}^{\mathsf{PP}}$ to $\mathsf{NP}^{\mathsf{BPP}}$. The associated optimization 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 3.6. *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 $n^{1-\alpha}$ unless* $\mathsf{P} = \mathsf{NP}$.

The complete proofs of both main theorems as well as formal definitions, detailed discussions, and analyses of the problem variants are given in Section 3.2 and Section 3.3. Already here, we can draw an important corollary which follows from Theorem 3.6 for the special case $\gamma = 0$.

Corollary 3.7. *Let $\delta \in (0,1)$. Then for any $\alpha \in (0,1)$ there is no polynomial time approximation algorithm for the Min-δ-Relevant-Input problem with an approximation factor of $n^{1-\alpha}$ unless* $\mathsf{P} = \mathsf{NP}$.

3.1.1 Related Concepts

Prime Implicant Explanations. A concept closely related to δ-relevant sets are prime implicant explanations [SCD18]. An implicant explanation of $\Psi(\mathbf{x})$ is a subset S of the variables such that $\mathbf{x}_S$ is sufficient for $\Psi(\mathbf{x})$. In other words, any completion $\mathbf{z}$ satisfying $\mathbf{z}_S = \mathbf{x}_S$ yields $\Psi(\mathbf{z}) = \Psi(\mathbf{x})$. In our terminology, implicants are precisely the one-relevant sets. A prime implicant is an implicant that is minimal with respect to set inclusion and can therefore not be reduced further. The δ-Relevant-Input problem with $\delta = 1$ answers the question if there exists a prime implicant of size at most k. This is known to be hard for $\mathsf{NP}^{\mathsf{coNP}}$ [EG95] in general. However, certain representations of Boolean functions such as Binary Decision Diagrams (BDD) [Ake78] allow for an efficient search over the prime implicants [CM92; MOM98].

As already briefly mentioned, the case $\delta = 1$ is often too strict, especially for high-dimensional problems as commonly found in modern machine learning. Let us illustrate this with the task of image classification as an example. In this case, often small regions of the input image can be manipulated in a way that changes a classifier prediction, e.g., through adversarial patches [Bro+17; Liu+18]. 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.

Thus, we extend the complexity analysis for $\delta < 1$, which adds a model counting component. Although model counting can be done efficiently for various classes of representations of Boolean functions, such as 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 will prove in Section 3.3.2. 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 $\Psi(x_1, \ldots, x_j = 0, \ldots, x_n)$ and $\Psi(x_1, \ldots, x_j = 1, \ldots, x_n)$. This is not true for δ-relevance.

Sufficient Explanations. Khosravi et al. introduced sufficient explanations for binary decision functions obtained from thresholding a continuous prediction model, e.g., a logistic regression classifier [Kho+19]. 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 $\Phi\colon \mathcal{X} \to [0,1]$ 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 binarization function (e.g., thresholding at 0.5), and $\mathcal{D}$ be a distribution on $\mathcal{X}$. A variable x_j of an input $\mathbf{x} \in \mathcal{X}$ is called a supporting variable, if

$$\begin{cases} \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\Phi(\mathbf{z}) \,\middle|\, \mathbf{z}_{\{j\}^c} = \mathbf{x}_{\{j\}^c}\right] \leq \Phi(\mathbf{x}) & \text{if } \theta(\Phi(\mathbf{x})) = 1, \\ \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\Phi(\mathbf{z}) \,\middle|\, \mathbf{z}_{\{j\}^c} = \mathbf{x}_{\{j\}^c}\right] > \Phi(\mathbf{x}) & \text{if } \theta(\Phi(\mathbf{x})) = 0. \end{cases}$$

In other words, randomizing x_j 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

$$\theta(\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\Phi(\mathbf{z}) \,|\, \mathbf{z}_S = \mathbf{x}_S\right]) = \theta(\Phi(\mathbf{x})).$$

For an already binary function Φ and $\mathcal{D} = \mathcal{U}(\{0,1\}^d)$, 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 Ribeiro, Singh, and Guestrin as local model-agnostic explanations [RSG18]. Given a generic function $\Phi\colon \mathcal{X} \to \mathcal{Y}$ from a domain $\mathcal{X}$ to a codomain $\mathcal{Y}$ (e.g., 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

$$A(\mathbf{x}) = 1 \qquad \text{and} \qquad \mathbb{P}_{\mathbf{z}\sim\mathcal{D}_\mathbf{x}}\left[\Phi(\mathbf{z}) = \Phi(\mathbf{x}) \,|\, A(\mathbf{z})\right] \geq \delta,$$

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

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

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 $\mathbf{x}$. Ribeiro, Singh, and Guestrin suggested to search for an anchor with the largest possible coverage, defined as $\text{coverage}(A) = \mathbb{P}_{\mathbf{z}\sim\mathcal{D}_{\mathbf{x}}}[A(\mathbf{z})]$. 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 $\mathcal{D}_{\mathbf{x}}$.

Shapley Values. Another concept for measuring the relevance or the *contribution* of individual variables to a collective are the Shapley values [Sha53] in cooperative game theory. Here, the variables are seen as players of a coalitional game, and the Shapley values describe a method to distribute the value achieved by a coalition of players to the individual players. This distribution fulfills a set of game theoretic properties that make it "fair".

Let $\nu\colon 2^{[n]} \to \mathbb{R}$ be a function that assigns a value to each subset of variables (coalition of players). It is called the *characteristic function* of the game. Then, the Shapley value of the j-th variable (j-th player) is defined as

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

which can be interpreted as the marginal contribution of the j-th variable to the value ν averaged over all possible coalitions. In general it is #P-hard to compute Shapley values [DP94]. However, in some cases efficient approximation algorithms exist [FWJ08].

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 randomizing the remaining variables. In [TK10] it was proposed to use

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

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

$$\begin{aligned} \mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z}) = \Psi(\mathbf{x}) \mid \mathbf{z}_S = \mathbf{x}_S] &= 1 - \frac{1}{2^{n-|S|}} \sum_{\substack{\mathbf{z}\in\{0,1\}^n \\ \mathbf{z}_S = \mathbf{x}_S}} |\Psi(\mathbf{z}) - \Psi(\mathbf{x})| \\ &= 1 - |\nu(S) + \mathbb{E}_{\mathbf{z}}[\Psi(\mathbf{z})] - \Psi(\mathbf{x})|, \end{aligned}$$

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

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 chapter is not to distribute the value of coalitions amongst the variables but to find (small) coalitions that are guaranteed to have a certain value.

3.2 Computational Complexity Analysis

Recall the first main theorem of this chapter, which shows that the δ-Relevant-Input problem is generally hard to solve for $\delta \in (0, 1)$.

Theorem 3.5. *For $\delta \in (0, 1)$ the δ-Relevant-Input problem is $\mathsf{NP}^{\mathsf{PP}}$-complete.*

The proof of Theorem 3.5 will be split into two parts. We will show that δ-Relevant-Input is $\mathsf{NP}^{\mathsf{PP}}$-hard in Section 3.2.1 and that it is contained in $\mathsf{NP}^{\mathsf{PP}}$ in Section 3.2.2.

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

We now give the first part of the proof of Theorem 3.5. 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 NP is the Boolean-satisfiability-problem [Coo71].

Definition 3.8. The Sat problem is defined as follows.

Given: A Boolean function $\Psi\colon \{0,1\}^n \to \{0,1\}$ in conjunctive normal form (CNF).

Decide: Does there exist $\mathbf{x} \in \{0,1\}^n$ such that $\Psi(\mathbf{x}) = 1$?

An extension of Sat, the majority-satisfiability-problem, is the canonical complete problem for PP [Gil74; Sim75].

Definition 3.9. The Maj-Sat problem is defined as follows.

Given: A Boolean function $\Psi\colon \{0,1\}^n \to \{0,1\}$ in conjunctive normal form (CNF).

Decide: Does $\mathbb{P}_{\mathbf{x}}[\Psi(\mathbf{x})] > \frac{1}{2}$ hold?

Finally, the canonical complete problem for $\mathsf{NP}^{\mathsf{PP}}$ is a further extension, called the exists-majority-satisfiability-problem. It was defined and proven $\mathsf{NP}^{\mathsf{PP}}$-complete in [LGM98].

Definition 3.10. The E-Maj-Sat problem is defined as follows.

Given: A Boolean function $\Psi\colon \{0,1\}^n \to \{0,1\}$ in conjunctive normal form (CNF) and $k \in [n]$.

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

Remark 3.11. Sat, Maj-Sat, and E-Maj-Sat are the prototypical complete problems for the complexity classes NP, PP, and $\mathsf{NP}^{\mathsf{PP}}$ respectively. While Sat asks only for the existence of a satisfying variable assignment Maj-Sat asks whether the majority of variable assignments is satisfying. Note that the Sat decision can also be rephrased in terms of probabilities by asking if $\mathbb{P}_{\mathbf{x}}[\Psi(\mathbf{x})] > 0$ holds. Finally, E-Maj-Sat asks for the existence of a fixed partial assignment to the first k variables such the majority of assignments to the remaining $(n-k)$ variables is satisfying. Maj-Sat can be seen as a special case of E-Maj-Sat with $k = 0$.

Three hurdles must be overcome if we want to reduce E-Maj-Sat to the δ-Relevant-Input problem.

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 randomize.
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 and give a chain of polynomial-time reductions

$$\text{E-Maj-Sat} \preceq_p \text{IP1} \preceq_p \text{IP2} \preceq_p \delta\text{-Relevant-Input} \tag{3.1}$$

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

Remark 3.12. Let Ψ_1 and Ψ_2 be two Boolean functions, not necessarily of different variables. Then

$$\begin{aligned} \mathbb{P}[\Psi_2] = 0 \quad &\Longrightarrow \quad \mathbb{P}[\Psi_1 \oplus \Psi_2] = \mathbb{P}[\Psi_1], \\ \mathbb{P}[\Psi_2] = 1 \quad &\Longrightarrow \quad \mathbb{P}[\Psi_1 \oplus \Psi_2] = 1 - \mathbb{P}[\Psi_1], \end{aligned}$$

and if Ψ_1 and Ψ_2 are independent, that is $\mathbb{P}[\Psi_1 \wedge \Psi_2] = \mathbb{P}[\Psi_1]\,\mathbb{P}[\Psi_2]$, also

$$\mathbb{P}[\Psi_2] = \frac{1}{2} \quad \Longrightarrow \quad \mathbb{P}[\Psi_1 \oplus \Psi_2] = \frac{1}{2}.$$

Lemma 3.13. *For $k \in \mathbb{N}$ let*

$$\text{EQ}\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}\colon (\mathbf{u},\mathbf{v}) \mapsto \bigwedge_{j=1}^{k} \neg(u_j \oplus v_j)$$

and

$$\text{FLIP}\colon \{0,1\}^k \times \{0,1\}^k \times \{0,1\} \to \{0,1\}\colon (\mathbf{u},\mathbf{v},t) \mapsto \left(\bigvee_{j=1}^{k} (u_j \oplus v_j)\right) \wedge t.$$

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

$$\mathbb{P}\left[A(\mathbf{u},\mathbf{v}) \wedge \mathrm{EQ}(\mathbf{u},\mathbf{v})\right] > 0, \quad \textit{and} \quad \mathbb{P}\left[A(\mathbf{u},\mathbf{v}) \wedge \neg\,\mathrm{EQ}(\mathbf{u},\mathbf{v})\right] > 0,$$

we have

$$\mathbb{P}\left[\Psi(\mathbf{u},\mathbf{r}) \oplus \mathrm{FLIP}(\mathbf{u},\mathbf{v},t) \,|\, A(\mathbf{u},\mathbf{v})\right] > \frac{1}{2} \iff \mathbb{P}\left[\Psi(\mathbf{u},\mathbf{r}) \,|\, \mathrm{EQ}(\mathbf{u},\mathbf{v}), A(\mathbf{u},\mathbf{v})\right] > \frac{1}{2}.$$

The condition EQ determines whether $\mathbf{u} = \mathbf{v}$ or not. As soon as there exists a j with $u_j \neq v_j$, $\mathrm{FLIP}(\mathbf{u},\mathbf{v},t)$ has the value of t, which is 1 with probability $\frac{1}{2}$. That means by modulo-adding FLIP 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 observe that FLIP can be rewritten as $\mathrm{FLIP}(\mathbf{u},\mathbf{v},t) = \neg\,\mathrm{EQ}(\mathbf{u},\mathbf{v}) \wedge t$ and therefore

$$\mathbb{P}\left[\mathrm{FLIP} \,|\, \mathrm{EQ}\right] = 0 \qquad \text{and} \qquad \mathbb{P}\left[\mathrm{FLIP} \,|\, \neg\,\mathrm{EQ}\right] = \frac{1}{2}.$$

Further, $\Psi \,|\, A$ and $\mathrm{FLIP} \,|\, A$ are conditionally independent given $\neg\,\mathrm{EQ}$ since in this case FLIP depends only on t. Thus, we obtain from Remark 3.12 that

$$\mathbb{P}\left[\Psi \oplus \mathrm{FLIP} \,|\, A, \mathrm{EQ}\right] = \mathbb{P}\left[\Psi \,|\, A, \mathrm{EQ}\right] \qquad \text{and} \qquad \mathbb{P}\left[\Psi \oplus \mathrm{FLIP} \,|\, A, \neg\,\mathrm{EQ}\right] = \frac{1}{2}.$$

Therefore,

$$\begin{aligned}
\mathbb{P}\left[\Psi \oplus \mathrm{FLIP} \,|\, A\right] &= \mathbb{P}\left[\Psi \oplus \mathrm{FLIP} \,|\, A, \mathrm{EQ}\right]\mathbb{P}\left[\mathrm{EQ}\right] + \mathbb{P}\left[\Psi \oplus \mathrm{FLIP} \,|\, A, \neg\,\mathrm{EQ}\right]\mathbb{P}\left[\neg\,\mathrm{EQ}\right] \\
&= \mathbb{P}\left[\Psi \,|\, A, \mathrm{EQ}\right]\mathbb{P}\left[\mathrm{EQ}\right] + \frac{1}{2}(1 - \mathbb{P}\left[\mathrm{EQ}\right]) \\
&= \frac{1}{2} + \left(\mathbb{P}\left[\Psi \,|\, A, \mathrm{EQ}\right] - \frac{1}{2}\right)\mathbb{P}\left[\mathrm{EQ}\right],
\end{aligned}$$

which finally implies that $\mathbb{P}\left[\Psi \oplus \mathrm{FLIP} \,|\, A\right] > \frac{1}{2}$ if and only if $\mathbb{P}\left[\Psi \,|\, A, \mathrm{EQ}\right] > \frac{1}{2}$. □

Fixing or Randomizing Variables

Let us now come to the first step of the reductive chain (3.1). In this, we translate the possibility of freely assigning the first k variables into the choice of fixing or randomizing variables from a given assignment. This choice is however still restricted to the first k variables.

Definition 3.14. We define the Intermediate Problem 1 (IP1) as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^n \to \{0,1\}$, a signal $\mathbf{x} \in \{0,1\}^n$, and $k \in [n]$.

Decide: Does there exist $S \subseteq [k]$ such that $\mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z}) \,|\, \mathbf{z}_S = \mathbf{x}_S\right] > \frac{1}{2}$?

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 3.15. *We have E-Maj-Sat $\leq_p$ IP1. In particular, IP1 is $\mathsf{NP}^{\mathsf{PP}}$-hard.*

Proof. Let $\{\Psi, k\}$ be an E-Maj-Sat instance. With a slight abuse of notation we denote by $\Psi\colon \{0,1\}^n \to \{0,1\}$ the Boolean circuit representation corresponding to the CNF formula of the E-Maj-Sat instance. From now on we will not distinguish between Ψ and the CNF formula that it represents. 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 n variables of Ψ into the first k variables and the remaining $n-k$ 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

- $\Psi'\colon \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{n-k} \times \{0,1\} \to \{0,1\}\colon (\mathbf{u}, \mathbf{v}, \mathbf{r}, t) \mapsto \Psi(\mathbf{u}, \mathbf{r}) \oplus \mathrm{FLIP}(\mathbf{u}, \mathbf{v}, t)$,
- $\mathbf{x}' = (\mathbf{0}_k, \mathbf{1}_k, \mathbf{0}_{n-k}, 0) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{n-k} \times \{0,1\}$,
- $k' = 2k$,

with FLIP defined as in Lemma 3.13. This is a polynomial time construction.

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 $\mathbb{P}_{\mathbf{r}}[\Psi(\mathbf{u}^*, \mathbf{r})] > \frac{1}{2}$. Now, we can choose $S' = \{\, j \in [k] \,:\, u_j^* = 0 \,\} \cup \{\, j \in [2k] \setminus [k] \,:\, u_{j-k}^* = 1 \,\} \subseteq [k'] = [2k]$. Let

$$A\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}\colon (\mathbf{u}, \mathbf{v}) \mapsto \left(\bigwedge_{j \in S' \cap [k]} \neg u_j \right) \wedge \left(\bigwedge_{j \in S' \cap ([2k] \setminus [k])} v_{j-k} \right)$$

and EQ as in Lemma 3.13. 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 $u_j^* = 0$ implies $u_j = 0$ and $u_j^* = 1$ implies $v_j = 1$ for all $j \in [k]$. 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

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

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

$$\begin{aligned} \mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] &= \mathbb{P}[\Psi'(\mathbf{u}, \mathbf{v}, \mathbf{r}, t) \,|\, A(\mathbf{u}, \mathbf{v})] \\ &= \mathbb{P}[\Psi(\mathbf{u}, \mathbf{r}) \oplus \mathrm{FLIP}(\mathbf{u}, \mathbf{v}, t) \,|\, A(\mathbf{u}, \mathbf{v})]\,. \end{aligned}$$

We use Lemma 3.13 together with $\mathbb{P}_{\mathbf{r}}[\Psi(\mathbf{u}^*, \mathbf{r})] > \frac{1}{2}$ to conclude $\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] > \frac{1}{2}$ 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 $\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \middle| \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] > \frac{1}{2}$. 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 $u_j^* = 0$ when $j \in S'$ and $u_j^* = 1$ when $j + k \in S'$. For that, we need two statements to be true. First, not both j and $j + k$ can be in S'. And second, if neither j nor $j + k$ are in S', then there is always the possibility of adding one of them to S' and still satisfy $\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \middle| \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] > \frac{1}{2}$.

To prove the first statement, assume toward a contradiction that there exists a $j \in [k]$ with $j \in S'$ and $j + k \in S'$. 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 $u_j = 0 \neq 1 = v_j$ and hence

$$\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\mathrm{FLIP}(\mathbf{u}, \mathbf{v}, t) \,|\, (\mathbf{u}, \mathbf{v})_{S'} = (\mathbf{u}', \mathbf{v}')_{S'}\right] = \mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[t \,|\, (\mathbf{u}, \mathbf{v})_{S'} = (\mathbf{u}', \mathbf{v}')_{S'}\right] = \frac{1}{2}.$$

Thus, Remark 3.12 would imply $\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \middle| \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] = \frac{1}{2}$, which contradicts the assumption that $\{\Psi', \mathbf{x}', k'\}$ is a *Yes*-instance for IP1.

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

$$\begin{aligned}
\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \,|\, A(\mathbf{u}, \mathbf{v})\right] = \; & \frac{1}{4}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 0, v_j = 0\right] \\
& + \frac{1}{4}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 0, v_j = 1\right] \\
& + \frac{1}{4}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 1, v_j = 0\right] \\
& + \frac{1}{4}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 1, v_j = 1\right].
\end{aligned}$$

For the second and third summand we have $u_j \neq v_j$, thus using Remark 3.12 again, we get

$$\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 0, v_j = 1\right] = \frac{1}{2} = \mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 1, v_j = 0\right],$$

and obtain

$$\begin{aligned}
\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \,|\, A(\mathbf{u}, \mathbf{v})\right] = \; & \frac{1}{4}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 0, v_j = 0\right] \\
& + \frac{1}{4}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 0, v_j = 1\right] \\
& + \frac{1}{4}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 0, v_j = 1\right] \\
& + \frac{1}{4}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 1, v_j = 1\right]
\end{aligned}$$

and therefore

$$\begin{aligned}
\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \,|\, A(\mathbf{u}, \mathbf{v})\right] = \; & \frac{1}{2}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), u_j = 0\right] \\
& + \frac{1}{2}\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u}, \mathbf{v}, t) \middle| A(\mathbf{u}, \mathbf{v}), v_j = 1\right].
\end{aligned}$$

Altogether, if $\mathbb{P}_{\mathbf{u},\mathbf{v},t}\left[\Psi'(\mathbf{u},\mathbf{v},t) \mid A(\mathbf{u},\mathbf{v})\right] > \frac{1}{2}$, then at least one of the additional conditions $u_j = 0$ or $v_j = 1$ must also yield a probability greater than $\frac{1}{2}$. This implies that if $\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \middle| \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] > \frac{1}{2}$ then either j or $j+k$ 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 $j \in [k]$ exactly one of the cases $j \in S'$ or $j+k \in S'$ occurs. Then, we can define $\mathbf{u}^* \in \{0,1\}^k$ as $u^*_j = 0$ if $j \in S'$ and $u^*_j = 1$ 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 3.13 and conclude from $\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \middle| \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] > \frac{1}{2}$ that $\mathbb{P}_{\mathbf{r}}\left[\Psi(\mathbf{u}^*,\mathbf{r})\right] > \frac{1}{2}$. This shows that $\{\Psi,k\}$ is a *Yes*-instance for E-Maj-Sat. □

Allowing for All Variables to Be Chosen

We continue with the second step of the reductive chain (3.1). Instead of choosing the set S from the first k variables we are free to chose it from all n variables but with cardinality at most k.

Definition 3.16. We define the Intermediate Problem 2 (IP2) as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^n \to \{0,1\}$, a signal $\mathbf{x} \in \{0,1\}^n$, and $k \in [n]$.

Decide: Does there exist $S \subseteq [n]$ with $|S| \leq k$ such that $\mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z}) \mid \mathbf{z}_S = \mathbf{x}_S\right] > \frac{1}{2}$?

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 3.17. *We have IP1 $\leq_p$ 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 n variables of Ψ into the first k variables and the remaining $n-k$ 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

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

$$\Psi'(\mathbf{u},\mathbf{v},\mathbf{r}_1,\mathbf{r}_2,\mathbf{r}_3) = \Psi(\mathbf{u},\mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3) \wedge \left(\bigwedge_{j=1}^{k}((u_j \oplus \neg x_j) \vee v_j)\right),$$

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

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

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 $\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z}) \,|\, \mathbf{z}_S = \mathbf{x}_S] > \frac{1}{2}$. Now, choose

$$S' = S \cup \{\, j \in [2k] \setminus [k] \,:\, j - k \notin S \,\}.$$

Then, $|S'| = |S| + (k - |S|) = k = k'$ and for each $j \in [k]$ exactly one of the cases $j \in S'$ or $j + k \in S'$ occurs. The former corresponds to fixing $u_j = x'_j = x_j$ and the latter to fixing $v_j = 1$. Therefore,

$$\mathbb{P}_{(\mathbf{u},\mathbf{v})}\left[\bigwedge_{j=1}^{k}((u_j \oplus \neg x_j) \vee v_j) \,\middle|\, (\mathbf{u},\mathbf{v})_{S'} = \mathbf{x}'_{S'}\right] = 1,$$

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

$$\begin{aligned}
\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] &= \mathbb{P}\left[\Psi'(\mathbf{u},\mathbf{v},\mathbf{r}_1,\mathbf{r}_2,\mathbf{r}_3) \,\middle|\, (\mathbf{u},\mathbf{v})_{S'} = \mathbf{x}'_{S'}\right] \\
&= \mathbb{P}\left[\Psi'(\mathbf{u},\mathbf{v},\mathbf{r}_1,\mathbf{r}_2,\mathbf{r}_3) \,\middle|\, \mathbf{u}_S = \mathbf{x}_S, \mathbf{v}_{[k]\setminus S} = \mathbf{1}_{k-|S|}\right] \\
&= \mathbb{P}[\Psi(\mathbf{u},\mathbf{r}) \,|\, \mathbf{u}_S = \mathbf{x}_S] \\
&= \mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z}) \,|\, \mathbf{z}_S = \mathbf{x}_S] > \frac{1}{2}. \qquad (3.2)
\end{aligned}$$

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 $S' \subseteq [2k + 3(n-k)]$ with $|S'| \leq k' = k$ and

$$\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] > \frac{1}{2}.$$

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

$$\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] \leq \mathbb{P}\left[\bigwedge_{j=1}^{k}((u_j \oplus \neg x_j) \vee v_j) \,\middle|\, (\mathbf{u},\mathbf{v})_{S'} = \mathbf{x}'_{S'}\right] = \left(\frac{3}{4}\right)^N,$$

and since $\left(\frac{3}{4}\right)^3 < \frac{1}{2}$ but $\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] > \frac{1}{2}$, 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 $\mathbf{z}'_{S'} = \mathbf{x}'_{S'}$ 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 $j \in S'$, we have $\mathbb{P}\left[(u_j \oplus \neg x_j) \vee v_j \,\middle|\, (\mathbf{u},\mathbf{v})_{S'} = \mathbf{x}'_{S'}\right] = 1$ and additionally having $j + k \in S'$ could not increase the probability of satisfying Ψ'. Hence, we can assume $j + k \notin S'$ in this case. Contrary, if $j \notin S'$, we have

$$\mathbb{P}\left[(u_j \oplus \neg x_j) \vee v_j \,\middle|\, (\mathbf{u},\mathbf{v})_{S'} = \mathbf{x}'_{S'}\right] = \frac{1}{2} + \frac{1}{2}\mathbb{P}\left[v_j \,\middle|\, (\mathbf{u},\mathbf{v})_{S'} = \mathbf{x}'_{S'}\right],$$

Table 3.1: **Changing Probability Thresholds.** Overview of four possibilities to change the probability threshold of a Boolean function Ψ by conjoining or disjoining it with an auxiliary function Π.

	$>$ to $\geq$	$\geq$ to $>$
raising probability threshold	Lemma 3.18	Lemma 3.22
lowering probability threshold	Lemma 3.19	Lemma 3.23

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

Altogether, without loss of generality, we can assume $S' \subseteq [2k]$, $|S'| = k$ and for each $j \in [k]$ exactly one of the cases $j \in S'$ or $j + k \in S'$ occurs. We now choose $S = S' \cap [k]$. Then, the rest of the proof proceeds exactly as in (3.2), and we conclude

$$\mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z}) \,|\, \mathbf{z}_S = \mathbf{x}_S\right] = \mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}'_{S'}) \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] > \frac{1}{2},$$

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

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. See Table 3.1 for an overview of the four possible ways to change the probability threshold of a Boolean function. We will make use of two of them in this section and the other two in the next section.

To raise the probability threshold, we make use of the following lemma.

Lemma 3.18. *Given $0 \leq \delta_1 < \delta_2 < 1$, for any $n \in \mathbb{N}$ there exists a monotone function $\Pi^{\delta_2}_{\delta_1,n} \colon \{0,1\}^d \to \{0,1\}$ with $d \in O(n^2)$ such that for all $\Psi \colon \{0,1\}^n \to \{0,1\}$ we have*

$$\mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z})\right] > \delta_1 \quad \Longleftrightarrow \quad \mathbb{P}_{(\mathbf{z},\mathbf{r})}\left[\Psi(\mathbf{z}) \vee \Pi^{\delta_2}_{\delta_1,n}(\mathbf{r})\right] \geq \delta_2.$$

The function $\Pi^{\delta_2}_{\delta_1,n}$ can be constructed in $O(d)$ time.

The constructive proof of Lemma 3.18 can be found in Appendix A.1. An analogous lemma is used to lower the probability threshold.

Lemma 3.19. *Given $0 < \delta_1 \leq \delta_2 \leq 1$, for any $n \in \mathbb{N}$ there exists a monotone function $\Pi^{\delta_2}_{\delta_1,n} \colon \{0,1\}^d \to \{0,1\}$ with $d \in O(n^2)$ such that for all $\Psi \colon \{0,1\}^n \to \{0,1\}$ we have*

$$\mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z})\right] > \delta_2 \quad \Longleftrightarrow \quad \mathbb{P}_{(\mathbf{z},\mathbf{r})}\left[\Psi(\mathbf{z}) \wedge \Pi^{\delta_2}_{\delta_1,n}(\mathbf{r})\right] \geq \delta_1.$$

The function $\Pi^{\delta_2}_{\delta_1,n}$ can be constructed in $O(d)$ time.

The constructive proof of Lemma 3.19 can be found in Appendix A.2.

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 3.20. *Given $0 < \delta < 1$, for any $n \in \mathbb{N}$ there exists $T_{\delta,n} \in \mathbb{N}$ and a monotone function $\Gamma_{\delta,n} \colon \{0,1\}^d \to \{0,1\}$ with $d + T_{\delta,n} \in O(n^2)$ such that for all $\Psi \colon \{0,1\}^n \to \{0,1\}$ we have*

$$\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z})] \geq \delta \iff \mathbb{P}_{(\mathbf{z},\mathbf{r},\mathbf{t})}\left[(\Psi(\mathbf{z}) \wedge \Gamma_{\delta,n}(\mathbf{r})) \vee \left(\bigwedge_{j=1}^{T} t_j\right)\right] \geq \delta,$$

for any $T \geq T_{\delta,n}$. The function $\Gamma_{\delta,n}$ can be constructed in $O(d)$ time.

The constructive proof of Lemma 3.20 can be found in Appendix A.3. Now, we are able to prove the following lemma.

Lemma 3.21. *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 consider the two cases $\delta \leq \frac{1}{2}$ and $\delta > \frac{1}{2}$ separately and start with the latter, that is $\delta \in (\frac{1}{2}, 1)$. 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.

For this, let $\Pi = \Pi^{\delta_2}_{\delta_1,n} \colon \{0,1\}^\ell \to \{0,1\}$ be as in Lemma 3.18 with $\delta_1 = \frac{1}{2}$ and $\delta_2 = \delta$. Further, let $\Gamma = \Gamma_{\delta,n+\ell} \colon \{0,1\}^m \to \{0,1\}$ and $T_{\delta,n+\ell}$ be defined according to Lemma 3.20 and set $d = T_{\delta,n+\ell} + k$. We define

- $\Psi' \colon \{0,1\}^n \times \{0,1\}^\ell \times \{0,1\}^m \times \{0,1\}^d \to \{0,1\}$
  $(\mathbf{z}, \mathbf{r}, \mathbf{s}, \mathbf{t}) \mapsto ((\Psi(\mathbf{z}) \vee \Pi(\mathbf{r})) \wedge \Gamma(\mathbf{s})) \vee (\bigwedge_{j=1}^{d} t_j)$
- $\mathbf{x}' = (\mathbf{x}, \mathbf{0}_\ell, \mathbf{0}_m, \mathbf{1}_d) \in \{0,1\}^n \times \{0,1\}^\ell \times \{0,1\}^m \times \{0,1\}^d$,
- $k' = k$.

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 $\bigwedge_{j=1}^{d} 1 = 1$.

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

$$\begin{aligned}
& \mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z}) \mid \mathbf{z}_S = \mathbf{x}_S] > \frac{1}{2} \\
\iff \quad & \mathbb{P}_{(\mathbf{z},\mathbf{r})}[\Psi(\mathbf{z}) \vee \Pi(\mathbf{r}) \mid \mathbf{z}_S = \mathbf{x}_S] \geq \delta \\
\iff \quad & \mathbb{P}_{(\mathbf{z},\mathbf{r},\mathbf{s},\mathbf{t})}\left[((\Psi(\mathbf{z}) \vee \Pi(\mathbf{r})) \wedge \Gamma(\mathbf{s})) \vee \left(\bigwedge_{j=1}^{d} t_j\right) \middle| \mathbf{z}_S = \mathbf{x}_S\right] \geq \delta \\
\iff \quad & \mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') = \Psi'(\mathbf{x}') \middle| \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] \geq \delta.
\end{aligned}$$

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 $\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') = \Psi'(\mathbf{x}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] \geq \delta$. Since Π and Γ are monotone and their initial input assignments according to $\mathbf{x}'$ 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 Π or Γ. Furthermore, at most $k' = k$ of the d variables in the additional conjunction from Lemma 3.20 can be included in S', which by the choice of d does not affect whether the overall probability threshold of δ is reached or not. Thus,

$$\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') = \Psi'(\mathbf{x}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] \geq \delta \quad \Longrightarrow \quad \mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') = \Psi'(\mathbf{x}') \,\middle|\, \mathbf{z}'_{S'\cap[n]} = \mathbf{x}'_{S'\cap[n]}\right] \geq \delta.$$

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

$$\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') = \Psi'(\mathbf{x}') \,\middle|\, \mathbf{z}'_{S} = \mathbf{x}'_{S}\right] \geq \delta \quad \Longleftrightarrow \quad \mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z}) \,|\, \mathbf{z}_S = \mathbf{x}_S\right] > \frac{1}{2},$$

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

The reduction for the other case $\delta \in (0, \frac{1}{2}]$ can be done analogously by using Lemma 3.19 instead of Lemma 3.18 and we omit the details for brevity. □

3.2.2 δ-Relevant-Input is Contained in $\mathsf{NP}^{\mathsf{PP}}$

We now come to the second part of the proof of Theorem 3.5. 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 3.18 and 3.19 (cf. Table 3.1), will be useful.

Lemma 3.22. *Given $0 \leq \delta_1 \leq \delta_2 < 1$, for any $n \in \mathbb{N}$ there exists a monotone function $\Pi^{\delta_2}_{\delta_1,n}\colon \{0,1\}^d \to \{0,1\}$ with $d \in O(n^2)$ such that for all $\Psi\colon \{0,1\}^n \to \{0,1\}$ we have*

$$\mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z})\right] \geq \delta_1 \quad \Longleftrightarrow \quad \mathbb{P}_{(\mathbf{z},\mathbf{r})}\left[\Psi(\mathbf{z}) \vee \Pi^{\delta_2}_{\delta_1,n}(\mathbf{r})\right] > \delta_2.$$

The function $\Pi^{\delta_2}_{\delta_1,n}$ can be constructed in $O(d)$ time.

The constructive proof of Lemma 3.22 can be found in Appendix A.1.

Lemma 3.23. *Given $0 < \delta_1 < \delta_2 \leq 1$, for any $n \in \mathbb{N}$ there exists a monotone function $\Pi^{\delta_2}_{\delta_1,n}\colon \{0,1\}^d \to \{0,1\}$ with $d \in O(n^2)$ such that for all $\Psi\colon \{0,1\}^n \to \{0,1\}$ we have*

$$\mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z})\right] \geq \delta_2 \quad \Longleftrightarrow \quad \mathbb{P}_{(\mathbf{z},\mathbf{r})}\left[\Psi(\mathbf{z}) \wedge \Pi^{\delta_2}_{\delta_1,n}(\mathbf{r})\right] > \delta_1.$$

The function $\Pi^{\delta_2}_{\delta_1,n}$ can be constructed in $O(d)$ time.

The constructive proof of Lemma 3.23 can be found in Appendix A.2.

Lemma 3.24. *For $\delta \in (0,1)$ the δ-Relevant-Input problem is contained in* $\mathsf{NP}^{\mathsf{PP}}$.

Proof. Again we consider the two cases $\delta \leq \frac{1}{2}$ and $\delta > \frac{1}{2}$ separately and start with the latter, that is $\delta \in (\frac{1}{2}, 1)$. Let $\{\Psi, \mathbf{x}, k\}$ be an instance of δ-Relevant-Input. It suffices to show that the decision problem whether a given set $S \subseteq [n]$ 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 $\Pi = \Pi^{\delta_2}_{\delta_1,n} \colon \{0,1\}^d \to \{0,1\}$ as in Lemma 3.23 for $\delta_1 = \frac{1}{2}$ and $\delta_2 = \delta$. Then,

$$\mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z}) \,|\, \mathbf{z}_S = \mathbf{x}_S\right] \geq \delta \quad \Longleftrightarrow \quad \mathbb{P}_{(\mathbf{z},\mathbf{r})}\left[\Psi(\mathbf{z}) \wedge \Pi(\mathbf{r}) \,|\, \mathbf{z}_S = \mathbf{x}_S\right] > \frac{1}{2}.$$

A probabilistic Turing machine can now draw a random assignment $(\mathbf{z}, \mathbf{r})$ conditioned on $\mathbf{z}_S = \mathbf{x}_S$ and evaluate $\Psi(\mathbf{z}) \wedge \Pi(\mathbf{r})$. 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 $S \subseteq [n]$ with $|S| \leq k$ and, using the oracle, check whether it is δ-relevant.

The other case $\delta \in (0, \frac{1}{2}]$ can be handled analogously by using Lemma 3.22 instead of Lemma 3.23 and we omit the details for brevity. □

3.3 Variations of the Problem

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.

The second variation concerns the optimization version of the problem. Here, we introduce a set size gap and relax the requirement to find the smallest δ-relevant set. Instead, for $k_1 < k_2$ we ask if a δ-relevant set of size k_1 exists or if all relevant sets must be of size at least k_2.

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 n can exist unless P = NP. Due to the connection between Boolean circuits and neural networks, that was already mentioned at the beginning of this chapter and illustrated in Figure 3.1, this inapproximability result shows theoretical limitations of interpretation methods for neural network decision. A detailed formal extension of the results to neural networks is given in Chapter 4.

3.3.1 Introducing a Probability Gap

As explained in the problem formulation, see Section 3.1, 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.

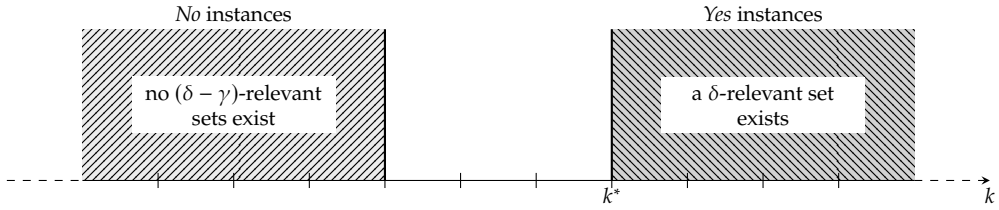


Figure 3.3: **A Problem Variant with Probability Gap.** Visualization of the γ-Gapped-δ-Relevant-Input problem for fixed Ψ and $\mathbf{x}$ and for various k. In the unmarked region in the center no δ-relevant set exists but $\widetilde{\delta}$-relevant sets could exist for any $\widetilde{\delta} < \delta$, 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 optimization problem Min-δ-Relevant-Input is the left boundary of the *Yes*-region.

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 Figure 3.3.

Definition 3.25. 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 $\Psi\colon \{0,1\}^n \to \{0,1\}$, a signal $\mathbf{x} \in \{0,1\}^n$, and $k \in [n]$.

Decide:

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

No: All $S \subseteq [n]$ with $|S| \leq k$ 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. Similar to the original problem formulation, we can also state an optimization 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. Figure 3.3). 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 optimization problem. However, problems of this type can be referred to as weak optimization problems [GLS88].

Definition 3.26. 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 $\Psi\colon \{0,1\}^n \to \{0,1\}$ and signal $\mathbf{x} \in \{0,1\}^n$.

Find: $k \in [n]$ such that

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

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

Remark 3.27. 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 3.1 provides a solution to the gapped problems for any $\gamma > 0$.

Lemma 3.28. *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 $S \subseteq [n]$ 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 $d = \left\lceil \frac{2\ln(3)}{\gamma^2} \right\rceil$ independent random binary vectors $\mathbf{b}^{(j)} \in \{0,1\}^{n-|S|}$ for $j \in [d]$ from the uniform distribution on $\{0,1\}^{n-|S|}$ and define $\mathbf{z}^{(j)} \in \{0,1\}^n$ as $\mathbf{z}^{(j)}_S = \mathbf{x}_S$ and $\mathbf{z}^{(j)}_{S^c} = \mathbf{b}^{(j)}$. Set

$$\xi = \frac{1}{d}\sum_{j=1}^{d} \xi_j, \quad \text{where} \quad \xi_j = \begin{cases} 1, & \text{if } \Psi(\mathbf{x}) = \Psi(\mathbf{z}^{(j)}) \\ 0, & \text{if } \Psi(\mathbf{x}) \neq \Psi(\mathbf{z}^{(j)}) \end{cases} \quad \text{for} \quad j = 1, \dots, d.$$

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

The random variables ξ_j are independently and identically Bernoulli(p) distributed variables with parameter

$$p = \mathbb{E}[\xi_j] = \mathbb{E}[\xi] = \mathbb{P}_{\mathbf{z}}\left[\Psi(\mathbf{z}_S) = \Psi(\mathbf{x}) \,|\, \mathbf{z}_S = \mathbf{x}_S\right].$$

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

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

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

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

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 $S \subseteq [n]$ with $|S| \leq k$ and, using the oracle, check if it is δ-relevant or not $(\delta-\gamma)$-relevant. □

3.3.2 Introducing a Set Size Gap

Even the gapped version of the minimization 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 optimization but a search problem. In order to give a meaning to the concept of approximation factors we use the following convention.

Definition 3.29. 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 3.26) with $\widetilde{k} \leq k \leq c\widetilde{k}$.

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

Definition 3.30. For $\delta \in (0, 1]$ and $\gamma \in [0, \delta)$ we define the Intermediate Problem 3 (IP3) as follows.

Given: A Boolean circuit $\Psi\colon \{0,1\}^n \to \{0,1\}$, a signal $\mathbf{x} \in \{0,1\}^n$, and $k_1, k_2 \in [n]$ with $k_1 \leq k_2$.

Decide:

Yes: There exists $S \subseteq [n]$ with $|S| \leq k_1$ such that S is δ-relevant for Ψ and $\mathbf{x}$.

No: All $S \subseteq [n]$ with $|S| \leq k_2$ are not $(\delta - \gamma)$-relevant for Ψ and $\mathbf{x}$.

Remark 3.31. The restriction of IP3 to instances with $k_1 = k_2$ is exactly the same as the γ-Gapped-δ-Relevant-Input problem. However, here we also allow the case $k_1 < k_2$ with a gap in the set sizes as illustrated in Figure 3.4.

Lemma 3.32. *For $\delta \in (0, 1)$ and $\gamma \in [0, 1)$ we have Sat $\preceq_p$ IP3. In particular, in this case IP3 is* NP*-hard.*

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

$$u^{(i)} = \bigwedge_{j=1}^{q} u_j^{(i)},$$

initially set to one. Fixing all $u_j^{(i)}$ sets $u^{(i)}$ to one, while randomizing all $u_j^{(i)}$ effectively sets $u^{(i)}$ to zero with high probability. If we now disjoin the resulting formula with a

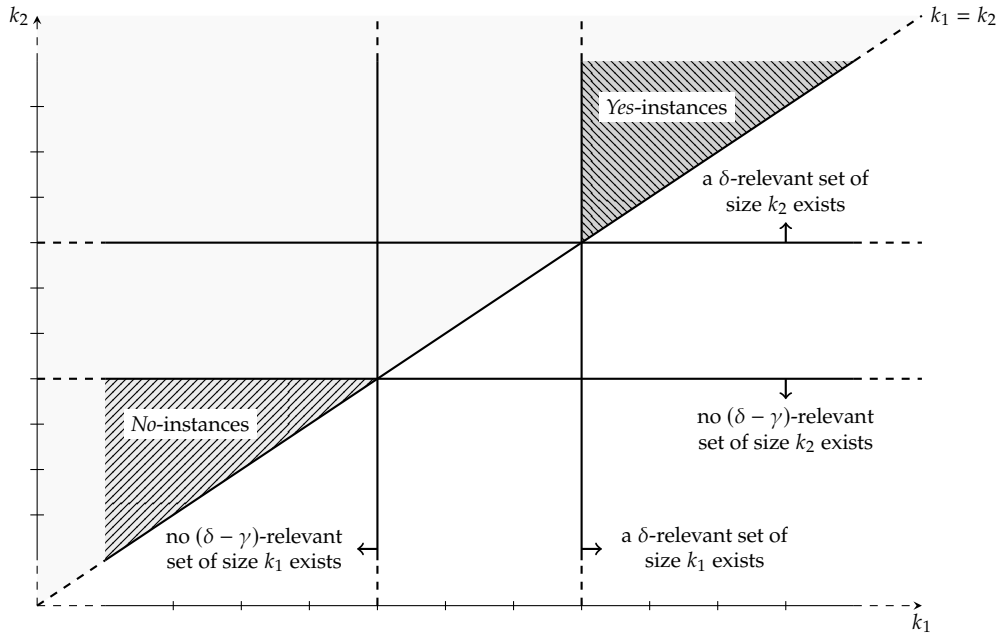


Figure 3.4: **A Problem Variant with Set Size Gap.** Visualization of the Intermediate Problem 3 for some fixed Ψ and $\mathbf{x}$ and for various k_1 and k_2. As before we do not expect an answer for this problem in the unmarked regions. The restriction to the diagonal $k_1 = k_2$ corresponds to the γ-Gapped-δ-Relevant-Input problem (cf. Figure 3.3).

polynomially large conjunction of additional independent variables, i.e.,

$$\Psi\left(\bigwedge_{j=1}^{q} u_j^{(1)}, \ldots, \bigwedge_{j=1}^{q} u_j^{(n)}\right) \vee \left(\bigwedge_{j=1}^{M} v_j\right),$$

initially also set to one, then any satisfying assignment for Ψ yields a δ-relevant set of size at most nq 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 nq results in the desired set size gap. We now make this argument formal.

Proof. Given a Sat instance in conjunctive normal form (CNF), let $\Psi\colon \{0,1\}^n \to \{0,1\}$ 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 $\{\Psi', \mathbf{x}', k_1', k_2'\}$ that is a *Yes*-instance for IP3 if and only if Ψ is a *Yes*-instance for Sat. Let

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

We set

- $k_1' = nq$,
- $k_2' \geq k_1'$ arbitrary but at most polynomial in n,

- $\Psi'\colon \{0,1\}^{n\times q} \times \{0,1\}^{k_2'+p} \to \{0,1\}$
  $(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)}, \mathbf{v}) \mapsto \Psi\left(\bigwedge_{j=1}^{q} u_j^{(1)}, \ldots, \bigwedge_{j=1}^{q} u_j^{(n)}\right) \vee \left(\bigwedge_{j=1}^{k_2'+p} v_j\right)$, with each $\mathbf{u}^{(i)} \in \{0,1\}^q$,
- $\mathbf{x}' = \mathbf{1}_{nq+k_2'+p}$.

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 $\mathbf{x} \in \{0,1\}^n$ with $\Psi(\mathbf{x}) = 1$. Let $S = \{\, j \in [n] \,:\, x_j = 1 \,\}$ and $S' = S \times [q]$. Then, $|S'| \leq k_1'$. Denote

$$A(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)}) = \bigwedge_{(i,j)\in S'} u_j^{(i)}.$$

Hence, S' is δ-relevant for Ψ' and $\mathbf{x}'$ if $\mathbb{P}\left[\Psi'(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)}, \mathbf{v}) \,\middle|\, A(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)})\right] \geq \delta$. We have

$$\begin{aligned}\mathbb{P}\left[\Psi'(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)}, \mathbf{v}) \,\middle|\, A(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)})\right] &\geq \mathbb{P}\left[\Psi\left(\bigwedge_{j=1}^{q} u_j^{(1)}, \ldots, \bigwedge_{j=1}^{q} u_j^{(n)}\right) \,\middle|\, A(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)})\right] \\ &\geq \mathbb{P}\left[\forall i \in [n] \,:\, \bigwedge_{j=1}^{q} u_j^{(i)} = x_i \,\middle|\, A(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)})\right].\end{aligned}$$

With a union bound and using the abbreviated notation $\mathbf{A} = A(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)})$, we obtain

$$\begin{aligned}\mathbb{P}\left[\forall i \in [n] \,:\, \bigwedge_{j=1}^{q} u_j^{(i)} = x_i \,\middle|\, \mathbf{A}\right] &= 1 - \mathbb{P}\left[\exists i \in [n] \,:\, \bigwedge_{j=1}^{q} u_j^{(i)} \neq x_i \,\middle|\, \mathbf{A}\right] \\ &= 1 - \mathbb{P}\left[\exists i \in S^c \,:\, \bigwedge_{j=1}^{q} u_j^{(i)}\right] \\ &\geq 1 - |S^c| 2^{-q} \\ &\geq \delta,\end{aligned}$$

which shows that $\{\Psi', \mathbf{x}', k_1', k_2'\}$ is a *Yes*-instance for IP3.

Sufficiency. Now, conversely, let Ψ be a *No*-instance for Sat. Then, for any subset $S' \subseteq [nq + k_2' + p]$ with $|S'| \leq k_2'$ we have

$$\begin{aligned}\mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') = \Psi'(\mathbf{x}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{x}'_{S'}\right] &= \mathbb{P}_{\mathbf{z}'}\left[\Psi'(\mathbf{z}') \,\middle|\, \mathbf{z}'_{S'} = \mathbf{1}_{|S'|}\right] \\ &= \mathbb{P}_{(\mathbf{u}^{(1)},\ldots,\mathbf{u}^{(n)},\mathbf{v})}\left[\bigwedge_{j=1}^{k_2'+p} v_j \,\middle|\, (\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)}, \mathbf{v})_{S'} = \mathbf{1}_{|S'|}\right] \\ &\leq 2^{-(k_2'+p-|S'|)} \\ &\leq 2^{-p} \\ &< \delta - \gamma.\end{aligned}$$

This shows that S' is not $(\delta - \gamma)$-relevant for Ψ' and $\mathbf{x}'$, hence $\{\Psi', \mathbf{x}', k_1', k_2'\}$ is a *No*-instance for IP3. □

Recall the second main theorem of this chapter which shows the inapproximability of the Min-γ-Gapped-δ-Relevant-Input problem.

Theorem 3.6. *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 $n^{1-\alpha}$ unless* P = NP.

The proof idea is to choose k_2' 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 of Theorem 3.6. We prove this by showing that the existence of an approximation algorithm for Min-γ-Gapped-δ-Relevant-Input with approximation factor $n^{1-\alpha}$ would allow us to decide IP3 for certain instances. These can be chosen as in the proof of Lemma 3.32, which in turn implies that we could decide Sat. This is only possible in polynomial time if P = NP.

Given a Sat instance as a CNF formula, let $\Psi\colon \{0,1\}^n \to \{0,1\}$ be a Boolean circuit representation of the CNF formula and $\{\Psi', \mathbf{x}', k_1', k_2'\}$ an equivalent IP3 instance as in the proof of Lemma 3.32. As before, we will not further distinguish between the CNF formula and the circuit Ψ representing it. We have seen that there is some freedom in the choice of k_2' as long as it satisfies $k_1' \leq k_2'$ and is at most polynomial in n. We will choose it in such a way, that any approximate solution k for Min-γ-Gapped-δ-Relevant-Input with approximation factor ${n'}^{1-\alpha}$ would allow us to decide $\{\Psi', \mathbf{x}', k_1', k_2'\}$ by checking whether $k < k_2'$ or $k > k_2'$. For this we set $k_2' = \left\lceil \max(2k_1'({k_1'}^{1-\alpha} + p^{1-\alpha}), (2k_1')^{\frac{1}{\alpha}} + 1) \right\rceil$ with $p = \left\lfloor \log_2\left(\frac{1}{\delta-\gamma}\right) \right\rfloor + 1$ as before. Recall that $k_1' = nq$ with $q = \left\lceil \log_2\left(\frac{n}{1-\delta}\right) \right\rceil$, so clearly k_2' is polynomial in n and $k_1' \leq k_2'$. Further, we have $k_2' > (2k_1')^{\frac{1}{\alpha}}$ so $1 - k_1' {k_2'}^{-\alpha} > \frac{1}{2}$ and therefore

$$k_2'(1 - k_1' {k_2'}^{-\alpha}) > \frac{k_2'}{2} \geq k_1'({k_1'}^{1-\alpha} + p^{1-\alpha}).$$

Now, let $n' = k_1' + k_2' + p$ denote the number of variables of Ψ'. By the subadditivity of the map $z \mapsto z^{1-\alpha}$ we finally obtain

$$k_1' {n'}^{1-\alpha} = k_1'(k_1' + k_2' + p)^{1-\alpha} \leq k_1' \left({k_1'}^{1-\alpha} + {k_2'}^{1-\alpha} + p^{1-\alpha} \right) < k_2'.$$

It remains to show that an IP3 instance with $k_2' > k_1' {n'}^{1-\alpha}$ can be decided by an approximation algorithm for Min-γ-Gapped-δ-Relevant-Input with approximation factor ${n'}^{1-\alpha}$. Assume such an algorithm exists and let k be an approximate solution. Then, there exists a true solution $\widetilde{k}$ with $\widetilde{k} \leq k \leq {n'}^{1-\alpha}\widetilde{k}$.

Firstly, assume that $\{\Psi', \mathbf{x}', k_1', k_2'\}$ is a *Yes*-instance for IP3. Then, there is a δ-relevant set of size k_1'. However, no set smaller than $\widetilde{k}$ can be δ-relevant. This implies $\widetilde{k} \leq k_1'$ and therefore $k \leq {n'}^{1-\alpha} k_1' < k_2'$.

Secondly, assume that $\{\Psi', \mathbf{x}', k_1', k_2'\}$ is a *No*-instance for IP3. Then, all sets of size at most k_2' are not $(\delta - \gamma)$-relevant. However, there exists a $(\delta - \gamma)$-relevant set of size $\widetilde{k}$. This implies $k \geq \widetilde{k} > k_2'$.

Altogether, checking whether $k < k_2'$ or $k > k_2'$ decides $\{\Psi', \mathbf{x}', k_1', k_2'\}$. □

Proof of Corollary 3.7. This is a direct consequence of Theorem 3.6 for the case $\gamma = 0$. □

3.4 Discussion

We want to briefly discuss the scope of our analysis in this chapter and the implications for algorithms that explain the predictions of classifiers. One could ask whether a solution set S to the Min-δ-Relevant-Input problem is in itself already a good explanation for a classifier prediction. We are not arguing that a solution set alone is enough to fully explain the decision of a classifier to humans. 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. Given a good explanation for a classification prediction, we want to be able to conclude: If we fix these input variables then the classification will remain unchanged with high probability.

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.

3.4.1 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 behavior 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 behavior 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 variables including the cat will get less relevant for the prediction "cat" if we add more variables covering the dog to it. In fact, this non-monotonicity property was essential for the constructions used in our proof of the inapproximability theorem.

3.4.2 Binary versus Continuous

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 (see Chapter 4) as well as Bayesian networks.[2]

Moreover, we only considered a binary partition into relevant and non-relevant variables analogous to the prime implicant explanation, even though many practical methods provide continuous relevance scores (in some cases even negative scores) [SWM17]. The concept of relevance scores is discussed in more detail in Chapter 4. Here, we want to briefly give two reasons for restricting our analysis to binary partitions until now.

Firstly, there is generally no agreed upon interpretation of what the continuous relevance scores produced by various algorithms mean. Therefore, we prefer to keep the clear meaning of a partition of the variables, in the same way as this is done for prime implicants.

Secondly, many prominent applications of relevance mappings rely on binarizations of the continuous relevance scores. In other words, the continuous scores are merely an intermediate algorithmic choice. 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.

3.4.3 Choice of Distribution

We restricted our analysis to the case of using the uniform distribution over the binary cube to randomize 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 recognizes 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 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.

3.5 Conclusion

There exists a wide variety of algorithms that aim to make modern machine learning methods interpretable. In turn, these algorithms themselves must be trusted. Thus, it is clearly important to define the exact problem that the algorithms try to solve and to gain insights about the quality of their solutions.

In this chapter we discussed that our probabilistic version of prime implicant explanations is a crucial part of the problem that practitioners want to solve when they design interpretation algorithms. We showed that the task of identifying the relevant components of an input assignment to the variables of a Boolean circuit is complete for the complexity class $\mathsf{NP}^{\mathsf{PP}}$ and thus, for example, as difficult as planing under uncertainty [DB90].

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

The chapter furthermore investigated whether it is possible to reduce the complexity of the problem at the cost of getting only approximate solutions. We relaxed the problem by introducing the promise of a probability gap that allows for efficient bounding of the fraction of satisfying assignments. Furthermore, we required that a solution set only approximates the optimal set up to any non-trivial approximation factor. Both these relaxations do not render this problem computationally feasible unless $\mathsf{P} = \mathsf{NP}$.

This makes practical guarantees for the interpretation of classifier decisions infeasible, as long as the classifiers are powerful enough to represent arbitrary logical functions. This includes the important case of ReLU neural networks (see Chapter 4 for details). Provable algorithmic guarantees would require further restrictions of the problem setting. However, the hardness instances we constructed 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 classifier functions and their inputs that depend on the actual data structures on which they have been trained. These, however, are not yet well enough understood. As long as this is the case, we have to rely on heuristic solutions that are thoroughly evaluated numerically.

In Chapter 4 we present our own heuristic algorithm for a continuous (non-discrete) variant of the δ-Relevant-Input problem and classifier functions with compositional (layered) structure (such as neural networks). For several image classification tasks we demonstrate numerically that our algorithm approximates small relevant sets better than other widely-used baseline methods.

4

Interpretability of Neural Network Classifiers

In this chapter we want to extend the concepts and results of Chapter 3 from Boolean functions to more general classifier functions. In fact, recall that the overall goal of the first part of this thesis is a study and analysis of the following questions regarding decisions made by neural network classifiers.

> *Q1: Is there a small part of the input variables that determines the classification decision with high probability?*
>
> *Q2: What is the smallest part of the input variables that determines the classification decision with high probability?*

In the previous chapter we have seen a connection between prime implicants and the explanation of decisions made by Boolean functions. The concept of prime implicants has also been extended from Boolean logic to abductive reasoning in first order logics [Mar91; Mar00] and to give explanations of more general classifier decisions [SCD18]. Recall that 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 deterministic requirement to produce the same classification for *all* completions is often too strict, especially for high-dimensional problems as commonly found in modern machine learning. Therefore, we introduced a relaxation of this notion, called δ-relevance, in Chapter 3. It can be seen as a probabilistic version of prime implicant explanations, which only requires that the classifier prediction remains unchanged with high probability. In order to analyze the interpretability of neural network classifiers we want to take this idea further and extend it from the discrete to the continuous setting.

4.1 The ϵ-Distortion-Input Problem

We start by extending the concept of δ-relevant sets of variables in Definition 3.1 from Boolean functions defined on a discrete domain to classifier functions defined on a continuous domain. The notion of δ-relevance relies on random completions of input variables that are not part of the relevant set. In the discrete setting we restricted our analysis to the uniform distribution $\mathcal{U}(\{0,1\}^n)$, i.e., all variables are considered independently and identically distributed. Let us first generalize this to other distributions as a preparation for the continuous setting.

Definition 4.1. For fixed $\mathbf{x} \in \{0,1\}^n$ or $\mathbf{x} \in [0,1]^n$, a probability distribution $\mathcal{V}$ on $\{0,1\}^n$ or $[0,1]^n$ respectively, and a set $S \subseteq [n]$, we call the random vector defined as

$$\mathbf{z}_S = \mathbf{x}_S \qquad \text{and} \qquad \mathbf{z}_{S^c} = \mathbf{e}_{S^c}, \quad \mathbf{e} \sim \mathcal{V},$$

an *obfuscation* of $\mathbf{x}$ with respect to $\mathcal{V}$ and S. We denote the resulting distribution as $\mathcal{V}_{S,\mathbf{x}}$ (abbreviated as $\mathcal{V}_S$ whenever the dependence on $\mathbf{x}$ is clear from context).

Remark 4.2. The distribution $\mathcal{V}_{S,\mathbf{x}}$ of the obfuscation $\mathbf{z}$ is obtained by marginalizing $\mathcal{V}$ over all variables in S and replacing them deterministically by $\mathbf{x}_S$.

Remark 4.3. Instead of marginalizing over variables in S one could also condition on them. If $\mathcal{V}$ has independently distributed variables, for example $\mathcal{V} = \mathcal{U}(\{0,1\}^n)$ or $\mathcal{V} = \mathcal{U}([0,1]^n)$, then $\mathcal{V}_{S,\mathbf{x}}$ has the same distribution as $\mathbf{z} \,|\, (\mathbf{z}_S = \mathbf{x}_S)$ with $\mathbf{z} \sim \mathcal{V}$. In other words, under the independence assumption there is no difference between conditioning on or marginalizing over variables in S. In the context of determining relevant variables it seems more natural to consider the marginalization approach for distributions with correlated variables, see [JMB20] and Section 4.5 for a discussion.

The formulation of δ-relevance in Definition 3.1 via a probability threshold for not changing the classification decision is too strict for classifier functions with a continuous domain and co-domain. Instead we use a reformulation based on expectation values.

Remark 4.4. The δ-relevance property of a set S of variables for a binary classifier $\Psi\colon \{0,1\}^n \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^n$ in Definition 3.1 can be rewritten as

$$\begin{aligned}
& \mathbb{P}_{\mathbf{z}\sim\mathcal{U}(\{0,1\}^n)}\left[\Psi(\mathbf{z}) = \Psi(\mathbf{x}) \,|\, \mathbf{z}_S = \mathbf{x}_S\right] \geq \delta \\
\iff \quad & \mathbb{P}_{\mathbf{z}\sim\mathcal{U}(\{0,1\}^n)}\left[\Psi(\mathbf{z}) \neq \Psi(\mathbf{x}) \,|\, \mathbf{z}_S = \mathbf{x}_S\right] \leq 1 - \delta \\
\iff \quad & \mathbb{E}_{\mathbf{z}\sim\mathcal{U}(\{0,1\}^n)}\left[|\Psi(\mathbf{z}) - \Psi(\mathbf{x})| \,|\, \mathbf{z}_S = \mathbf{x}_S\right] \leq 1 - \delta \\
\iff \quad & \mathbb{E}_{\mathbf{z}\sim\mathcal{U}_S}[|\Psi(\mathbf{z}) - \Psi(\mathbf{x})|] \leq 1 - \delta,
\end{aligned}$$

where Remark 4.3 was used in the last step.

This formulation using the expected difference of predictions is well-suited to be generalized to classifiers on a continuous domain. Also other distance measures can be considered instead of the absolute difference. In the following, the exact choice of the distance function will not be of importance as long as it satisfies two basic properties.

Assumption 4.5. For the remainder of this chapter let $\mathrm{dist}\colon [0,1] \times [0,1] \to \mathbb{R}_{\geq 0}$ be a measurable function satisfying the two properties

(P1) $\mathrm{dist}(\cdot,\cdot)$ is bounded on $[0,1]^2$ by a constant $M > 0$,

(P2) $\mathrm{dist}(0,0) = \mathrm{dist}(1,1) = 0$ and $\mathrm{dist}(0,1) = \mathrm{dist}(1,0) = 1$.

This can be guaranteed (if necessary by appropriate rescaling) for most common positive definite and symmetric distance measures, such as absolute or squared differences.

We are now ready to define the analogous concept to δ-relevant sets.

Definition 4.6. Let $\Phi\colon [0,1]^n \to [0,1]$, $\mathbf{x} \in [0,1]^n$, and $\mathcal{V}$ be a probability distribution on $[0,1]^n$. Then the *distortion* of a set $S \subseteq [n]$ with respect to Φ, $\mathbf{x}$, and $\mathcal{V}$ is defined as

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

We use the abbreviated notation $D(S)$ whenever Φ, $\mathbf{x}$, and $\mathcal{V}$ are clear from the context.

In other words, a set of variables S can be considered δ-relevant if it achieves a limited distortion $D(S) \leq 1 - \delta$.

Definition 4.7. Let $\Phi\colon [0,1]^n \to [0,1]$, $\mathbf{x} \in [0,1]^n$, and $\mathcal{V}$ be a probability distribution on $[0,1]^n$. Then the *rate* of a distortion limit $\epsilon \geq 0$ with respect to Φ, $\mathbf{x}$, and $\mathcal{V}$ is defined as

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

We use the abbreviated notation $R(\epsilon)$ whenever Φ, $\mathbf{x}$, and $\mathcal{V}$ are clear from the context.

The distortion limit ϵ takes the role of $1 - \delta$ from before. We will also refer to R as the *rate-distortion* function as it precisely describes the trade-off between distortion limits and the sizes of relevant sets (rates) that are necessary to achieve them.

The idea of formulating relevance in a rate-distortion viewpoint can also be motivated with the hypothetical setup illustrated in Figure 4.1. The terminology is borrowed from information theory where rate-distortion is used to analyze 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 fulfill: Given a distortion limit ϵ, the goal is to find a set S achieving the minimum in Definition 4.7, i.e., evaluating the rate-distortion function. This generalizes 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 fulfill 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 4.3.1 and 4.4.

Analogous to the δ-Relevant-Input problem in the binary case (Definition 3.3) we can define the corresponding decision problem for the continuous setting.

Definition 4.8. For $\epsilon \geq 0$, $\mathcal{F} \subseteq \{\Phi\colon [0,1]^n \to [0,1]\}$, and a probability distribution $\mathcal{V}$ on $[0,1]^n$, we define the $(\mathcal{F}, \mathcal{V})$-$\epsilon$-Distortion-Input problem as follows.

Given: A function $\Phi \in \mathcal{F}$, a signal $\mathbf{x} \in [0,1]^n$, and $k \in [n]$.

Decide: Does there exist $S \subseteq [n]$ with $|S| \leq k$ such that $D(S, \Phi, \mathbf{x}, \mathcal{V}) \leq \epsilon$?

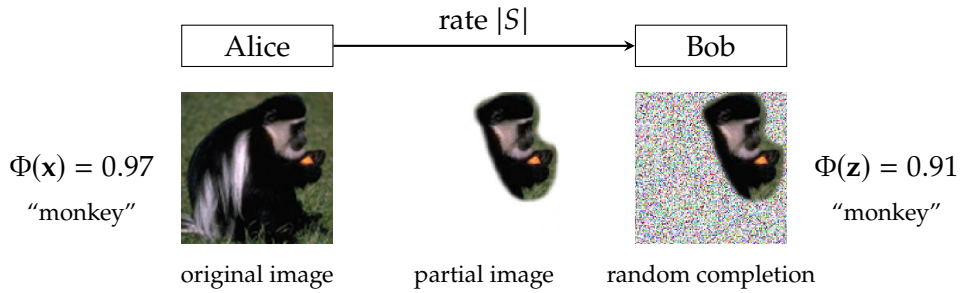


Figure 4.1: **Rate-Distortion Viewpoint.** We motivate the rate-distortion viewpoint from the following hypothetical scenario: two people, Alice and Bob, have access to the same neural network classifier. Alice classified an image as a "monkey" and wants to convey this to Bob. She is only allowed to 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.

The set $\mathcal{F}$ of "allowed" function takes the role of the Boolean circuits from before and will be chosen as a set of neural network functions in our complexity analysis. The associated minimization problem is defined in the usual way.

Definition 4.9. For $\epsilon \geq 0$, $\mathcal{F} \subseteq \{\Phi\colon [0,1]^n \to [0,1]\}$, and a probability distribution $\mathcal{V}$ on $[0,1]^n$, we define the Min-$(\mathcal{F},\mathcal{V})$-$\epsilon$-Distortion-Input problem as follows.

Given: A function $\Phi \in \mathcal{F}$ and a signal $\mathbf{x} \in \{0,1\}^n$.

Task: Find the minimal $k \in \mathbb{N}$ such that there exist $S \subseteq [n]$ with $|S| \leq k$ and so that $D(S, \Phi, \mathbf{x}, \mathcal{V}) \leq \epsilon$.

Remark 4.10. The $(\mathcal{F},\mathcal{V})$-$\epsilon$-Distortion-Input decision problem answers the question if the rate-distortion function $R(\epsilon, \Phi, \mathbf{x}, \mathcal{V})$ for a given Φ and $\mathbf{x}$ is at most k. The corresponding minimization problem Min-$(\mathcal{F},\mathcal{V})$-$\epsilon$-Distortion-Input is equivalent to evaluating $R(\epsilon, \Phi, \mathbf{x}, \mathcal{V})$ for a given Φ and $\mathbf{x}$.

4.2 Computational Complexity Analysis

The inapproximability of small relevant sets has been shown for binary functions, represented as Boolean circuits, and the uniform distribution $\mathcal{U}(\{0,1\}^n)$ on $\{0,1\}^n$ in Corollary 3.7. We will now generalize this result for classifiers on a continuous domain.

Theorem 4.11. *Let $\mathcal{V} = \mathcal{U}([0,1]^n)$, $\mathcal{F} = \mathcal{NN}^{n,*,1}_{*,\varrho,\varrho}$, and $\epsilon \in (0,1)$. Then for any $\alpha \in (0,1)$ there is no polynomial time approximation algorithm for the Min-$(\mathcal{F},\mathcal{V})$-$\epsilon$-Distortion-Input problem with an approximation factor $n^{1-\alpha}$ unless* P = NP.

Recall that $\mathcal{NN}^{n,*,1}_{*,\varrho,\varrho}$ is the set of all ReLU neural networks of finite depth with n input variables and a single output. For ease of presentation, we state and proof the result for

the uniform distribution. However, we want to remark that it is also valid for more general distributions, see Section 4.5 for a discussion. The proof strategy is based on the fact that neural networks with ReLU activations can efficiently represent Boolean circuits.

Definition 4.12. Let $m \in \mathbb{N}$, $\Psi\colon \{0,1\}^n \to \{0,1\}^m$ and $\Phi\colon [0,1]^n \to [0,1]^m$. Then Φ is said to *interpolate* Ψ if Φ restricted to $\{0,1\}^n$ is equal to Ψ.

Lemma 4.13. *Let $\Psi\colon \{0,1\}^n \to \{0,1\}$ be a Boolean circuit function. Then there exists a ReLU neural network $\Phi \in \mathcal{NN}^{n,*,1}_{*,\varrho,\varrho}$ that interpolates Ψ and has a size (depth and width) polynomial in the size of Ψ.*

This follows from results relating Boolean circuits to linear threshold circuits of comparable size [Par96] and linear threshold circuits to ReLU networks of comparable size [MB17]. See also Figure 3.1 for an example of a neural network representation of a Boolean circuit.

The idea for the proof of Theorem 4.11 can be summarized 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 $\mathcal{U}([0,1]^n)$ into the binary distribution $\mathcal{U}(\{0,1\}^n)$ 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 $D(S, \Phi, \mathbf{x}, \mathcal{U}([0,1]^n))$ and $D(S, \Psi, \mathbf{x}, \mathcal{U}(\{0,1\}^n))$ 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 3.1. Thus, it can be shown that for the right choice of η any approximation algorithm for Min-$(\mathcal{NN}^{n,*,1}_{*,\varrho,\varrho}, \mathcal{U}([0,1]^n))$-$\epsilon$-Distortion-Input would also be an approximation algorithm for the Min-γ-Gapped-δ-Relevant-Input problem with the same approximation factor. The inapproximability result of Theorem 3.6 thus carries over to the continuous setting.

For brevity, we introduce the notation

$$D^b_{\Phi,\mathbf{x}}(S) = D(S, \Phi, \mathbf{x}, \mathcal{U}(\{0,1\}^n)),$$
$$D^c_{\Phi,\mathbf{x}}(S) = D(S, \Phi, \mathbf{x}, \mathcal{U}([0,1]^n)),$$

for the distortions with respect to the uniform distribution on the binary and the continuous domain respectively. As observed in Remark 4.4 we can relate the binary distortion D^b to δ-relevance.

Lemma 4.14. *Let $\Psi\colon \{0,1\}^n \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^n$. Then for any $S \subseteq [n]$ we have*

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

Proof. This follows from a direct calculation analogous to Remark 4.4 above using the property (P2) of the distance function. □

For $0 < \eta \leq 1$ we set $\Phi_\eta(\mathbf{x}) = \varphi\left(\frac{1}{\eta}\left(\mathbf{x} - \frac{1-\eta}{2}\mathbf{1}_n\right)\right)$ with

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

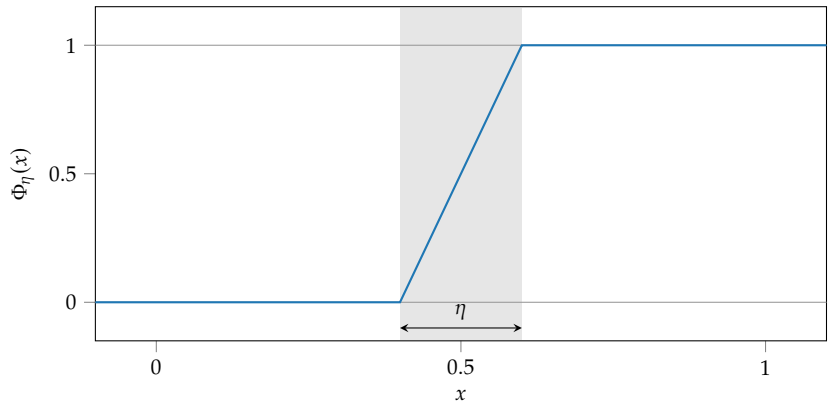


Figure 4.2: **Binarization of Inputs.** Illustration of the binarization function Φ_η for one component. The event E_S in the proof of Lemma 4.15 describes those $\mathbf{z} \in [0,1]^n$ for which at least one component not indexed by S falls into the gray marked region of width η.

and observe that Φ_η interpolates the identity on $\{0,1\}^n$ and can be realized by two ReLU layers of size $O(n)$, cf. Figure 4.2.

The following lemma relates the distortions D^b and D^c for a discrete function Ψ and a continuous map Φ interpolating it.

Lemma 4.15. *Let $\Psi\colon \{0,1\}^n \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^n$. Then for any $\Phi_0\colon [0,1]^n \to [0,1]$ interpolating Ψ, $S \subseteq [n]$, 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

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

Proof. The first part of the claim follows directly from the fact that Φ_0 and $\Phi = \Phi_0 \circ \Phi_\eta$ interpolate Ψ. For the second part we consider the event

$$E_S = \left\{ \mathbf{z} \in [0,1]^n \,:\, \exists j \in S^c \text{ such that } z_j \in \left(\frac{1-\eta}{2}, \frac{1+\eta}{2} \right) \right\},$$

and observe

$$\mathbb{P}_{\mathbf{z}\sim\mathcal{U}([0,1]^n)}\left[\mathbf{z} \in E_S\right] = 1 - (1-\eta)^{n-|S|}.$$

We introduce the abbreviated notation

$$\begin{aligned} A_S &= \mathbb{E}_{\mathbf{z}\sim\mathcal{U}([0,1]^n)}\left[\operatorname{dist}(\Phi(\mathbf{z}), \Phi(\mathbf{x})) \,|\, \mathbf{z}_S = \mathbf{x}_S, \mathbf{z} \in E_S\right] \\ B_S &= \mathbb{E}_{\mathbf{z}\sim\mathcal{U}([0,1]^n)}\left[\operatorname{dist}(\Phi(\mathbf{z}), \Phi(\mathbf{x})) \,|\, \mathbf{z}_S = \mathbf{x}_S, \mathbf{z} \notin E_S\right]. \end{aligned}$$

We can split the expectation value in the continuous distortion term as

$$\begin{aligned} D^c_{\Phi,\mathbf{x}}(S) &= \mathbb{E}_{\mathbf{z}\sim\mathcal{U}([0,1]^n)}\left[\operatorname{dist}(\Phi(\mathbf{z}), \Phi(\mathbf{x})) \,|\, \mathbf{z}_S = \mathbf{x}_S\right] \\ &= A_S \mathbb{P}_{\mathbf{z}\sim\mathcal{U}([0,1]^n)}\left[\mathbf{z} \in E_S\right] + B_S \mathbb{P}_{\mathbf{z}\sim\mathcal{U}([0,1]^n)}\left[\mathbf{z} \notin E_S\right] \\ &= A_S \left(1 - (1-\eta)^{n-|S|}\right) + B_S (1-\eta)^{n-|S|}. \end{aligned}$$

For the second term, we get

$$\begin{aligned} B_S &= \mathbb{E}_{\mathbf{z}\sim\mathcal{U}([0,1]^n)}\left[\operatorname{dist}(\Phi_0\circ\Phi_\eta(\mathbf{z}),\Phi_0\circ\Phi_\eta(\mathbf{x}))\,\middle|\,\mathbf{z}_S=\mathbf{x}_S,\mathbf{z}\notin E_S\right] \\ &= \mathbb{E}_{\mathbf{z}\sim\mathcal{U}(\{0,1\}^n)}\left[\operatorname{dist}(\Phi_0(\mathbf{z}),\Phi_0(\mathbf{x}))\,|\,\mathbf{z}_S=\mathbf{x}_S\right] \\ &= \mathbb{E}_{\mathbf{z}\sim\mathcal{U}(\{0,1\}^n)}\left[\operatorname{dist}(\Psi(\mathbf{z}),\Psi(\mathbf{x}))\,|\,\mathbf{z}_S=\mathbf{x}_S\right] \\ &= D^b_{\Psi,\mathbf{x}}(S), \end{aligned}$$

where the second equality follows from the choice of Φ_η and $\mathbf{z}\notin E_S$ and the third equality follows from the fact that Φ_0 interpolates Ψ. Thus, we conclude

$$\begin{aligned} D^c_{\Phi,\mathbf{x}}(S) &= A_S\left(1-(1-\eta)^{n-|S|}\right)+D^b_{\Psi,\mathbf{x}}(S)(1-\eta)^{n-|S|} \\ &= D^b_{\Psi,\mathbf{x}}(S)+\left(1-(1-\eta)^{n-|S|}\right)(A_S-B_S). \end{aligned}$$

Using Bernoulli's inequality and $0\le A_S,B_S\le M$, this finally results in

$$|D^c_{\Phi,\mathbf{x}}(S)-D^b_{\Psi,\mathbf{x}}(S)|\le M\left(1-(1-\eta)^{n-|S|}\right)\le M(n-|S|)\eta\le Mn\eta,$$

which finishes the proof. □

We now come to the proof of the main theorem of this section.

Proof of Theorem 4.11. 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. We will construct $\Phi\in\mathcal{F}$ so that the instance $\{\Phi,\mathbf{x}\}$ of Min-$(\mathcal{F},\mathcal{U}([0,1]^n))$-$\epsilon$-Distortion-Input has a solution that is also valid for the $\{\Psi,\mathbf{x}\}$ instance.

Let $\Phi_0\colon[0,1]^n\to[0,1]$ be a neural network that interpolates Ψ, set $\eta=\frac{\gamma}{2Mn}$ and $\Phi=\Phi_0\circ\Phi_\eta$. Recall that solving the Min-$(\mathcal{F},\mathcal{U}([0,1]^n))$-$\epsilon$-Distortion-Input instance is the same as evaluating $R(\epsilon,\Phi,\mathbf{x},\mathcal{U}([0,1]^n))$. We show that $R(\epsilon,\Phi,\mathbf{x},\mathcal{U}([0,1]^n))$ is a solution for the Min-γ-Gapped-δ-Relevant-Input problem instance $\{\Psi,\mathbf{x}\}$, i.e., it fulfills both conditions in Definition 3.26. To see this, let

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

and hence $|S^*|=R(\epsilon,\Phi,\mathbf{x},\mathcal{U}([0,1]^n))$. Lemma 4.15 yields

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

and together with Lemma 4.14 we get

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

showing that the first condition in Definition 3.26 is satisfied.

Similarly, for any S with $|S|<R(\epsilon,\Phi,\mathbf{x},\mathcal{U}([0,1]^n))$ we know $D(S,\Phi,\mathbf{x},\mathcal{U}([0,1]^n))>\epsilon$. Thus by Lemma 4.15

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

and again using Lemma 4.14 we obtain

$$\mathbb{P}_{\mathbf{z}\sim\mathcal{U}(\{0,1\}^n)}\left[\Psi(\mathbf{z}) = \Psi(\mathbf{x}) \mid \mathbf{z}_S = \mathbf{x}_S\right] < \delta,$$

showing that the second condition in Definition 3.26 is satisfied as well.

Hence, any algorithm approximating Min-$(\mathcal{F}, \mathcal{U}([0,1]^n))$-$\epsilon$-Distortion-Input 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 by Theorem 3.6 that achieving the factor $n^{1-\alpha}$ is NP-hard for any $\alpha \in (0,1)$, which completes the proof. □

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. Existing heuristics tend to produce continuous relevance scores instead of a strict partition into a relevant and a non-relevant subset of input components. We will discuss this in the next section and then present our own novel heuristic approach in Section 4.3.2.

4.3 Relaxation of the Problem

Until now we considered partitions of the input components of a classifier into a set S of relevant and its complement S^c of non-relevant ones. In contrast, many existing relevance mapping algorithms produce continuous relevance scores for each variable. These can be encoded by a vector $\mathbf{s} \in [0,1]^n$, where 0 means least relevant and 1 means most relevant.[1] It seems not immediately clear how the continuous scores relate to the partitions that were discussed in Sections 4.1 and 4.2. However, we argue that the exact numerical values of continuous relevance map are generally meaningless, in the sense that knowing the precise value s_j of the j-th variable has no specific meaning. Instead, it is the *ordering* of the input components according to their relevance scores that is of importance, i.e., the relations $s_i < s_j$ or $s_i > s_j$ between different variables. Finally, any ordering of variables can be associated to a partition by choosing to include the k most relevant variables in S and the remaining ones in S^c for some value of $k \in [n]$.

4.3.1 Relevance Scores and Orderings

Given an ordering of input components according to their relevance for a prediction (either explicitly produced by an interpretability method, see Section 4.3.3, or implicitly inferred from a score vector $\mathbf{s}$ produced by an interpretability method, see Section 4.3.2), we denote by $\pi\colon [n] \to [n] \in \mathrm{Sym}(n)$ the permutation that describes the ordering. This means that $\pi(k)$ is considered the k-th most relevant input component and $\pi([k])$ is the set of the k most relevant input components. Such an ordering can be seen as a greedy approach to solve one of the following two questions[2], related to our initial question $Q2$, for varying ϵ.

[1] Some methods also consider "negative" relevance scores for features speaking against a classification decision. In this case the scores are encoded in the range $[-1,1]$ instead of $[0,1]$.

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

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 [ZF14], 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 4.11 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. Similar to the rate function of a set of variables introduced in Definition 4.7 we can define the rate function of an ordering of variables.

Definition 4.16. Let $\Phi\colon [0,1]^n \to [0,1]$, $\mathbf{x} \in [0,1]^n$, and $\mathcal{V}$ be a probability distribution on $[0,1]^n$. For a permutation $\pi\colon [n] \to [n] \in \mathrm{Sym}(n)$ the *rate associated to* π of a distortion limit $\epsilon \geq 0$ with respect to Φ, $\mathbf{x}$, and $\mathcal{V}$ is defined as

$$R_\pi(\epsilon, \Phi, \mathbf{x}, \mathcal{V}) = \min\left\{k \in [n] \,:\, D(\pi([k]), \Phi, \mathbf{x}, \mathcal{V}) \leq \epsilon\right\}.$$

We use the abbreviated notation $R_\pi(\epsilon)$ whenever Φ, $\mathbf{x}$, and $\mathcal{V}$ are clear from the context.

Remark 4.17. Clearly $R(\epsilon) \leq R_\pi(\epsilon)$ holds for any ordering π and all $\epsilon \geq 0$. But we can evaluate orderings by how well they can approximate the optimal rate $R(\epsilon)$.

It would be desirable to obtain meaningful upper bounds on the approximation error between $R(\epsilon)$ and $R_\pi(\epsilon)$. 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 sections and then come back to the relevance orderings for comparing it to other established methods in Section 4.4.

4.3.2 Sparse Rate-Distortion Explanations

Finding the optimal partition into S and S^c for varying distortion limits $D(S) \leq \epsilon$ is a hard combinatorial optimization problem. As we have seen, component orderings can serve as greedy approximations. One could attempt to obtain such orderings either explicitly or implicitly from continuous relevance scores. We start with the latter, since most of the established interpretability methods for neural networks produce continuous relevance scores. Therefore, we make use of the following problem relaxation that was already briefly mentioned above. Instead of binary relevance decisions (*relevant* versus *non-relevant*) encoded by the set S, we allow for a continuous score for each component, encoded by a vector $\mathbf{s} \in [0,1]^n$, where 0 means least relevant and 1 means most relevant. We redefine the obfuscation of $\mathbf{x}$ with respect to $\mathbf{s}$ analogous to Definition 4.1.

Definition 4.18. For fixed $\mathbf{x} \in [0,1]^n$, a probability distribution $\mathcal{V}$ on $[0,1]^n$, and a score $\mathbf{s} \in [0,1]^n$, we call the random vector

$$\mathbf{z} = \mathbf{x} \odot \mathbf{s} + \mathbf{e} \odot (\mathbf{1}_n - \mathbf{s})$$

defined as the component-wise convex combination of $\mathbf{x}$ and a random vector $\mathbf{e} \sim \mathcal{V}$ an *obfuscation* of $\mathbf{x}$ with respect to $\mathcal{V}$ and $\mathbf{s}$. We denote the resulting distribution as $\mathcal{V}_{\mathbf{s},\mathbf{x}}$ (or simply abbreviated as $\mathcal{V}_\mathbf{s}$ whenever the dependence on $\mathbf{x}$ is clear from context).

This is a generalization of the obfuscation introduced in Definition 4.1 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 $\|\mathbf{s}\|_1 = \sum_{i=1}^n |s_i|$. Analogous to the distortion of a set S of variables introduced in Definition 4.6 we can define the distortion of a score vector $\mathbf{s}$.

Definition 4.19. Let $\Phi\colon [0,1]^n \to [0,1]$, $\mathbf{x} \in [0,1]^n$, and $\mathcal{V}$ be a probability distribution on $[0,1]^n$. Then the *distortion* of a score $\mathbf{s} \in [0,1]^n$ with respect to Φ, $\mathbf{x}$, and $\mathcal{V}$ is defined as

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

We use the abbreviated notation $D(\mathbf{s})$ whenever Φ, $\mathbf{x}$, and $\mathcal{V}$ are clear from the context.

Similarly we define the corresponding rate function.

Definition 4.20. Let $\Phi\colon [0,1]^n \to [0,1]$, $\mathbf{x} \in [0,1]^n$, and $\mathcal{V}$ be a probability distribution on $[0,1]^n$. Then the *rate* of a distortion limit $\epsilon \geq 0$ with respect to Φ, $\mathbf{x}$, and $\mathcal{V}$ is defined as

$$R(\epsilon, \Phi, \mathbf{x}, \mathcal{V}) = \min \left\{ \|\mathbf{s}\|_1 \,:\, \mathbf{s} \subseteq [0,1]^n, D(\mathbf{s}, \Phi, \mathbf{x}, \mathcal{V}) \leq \epsilon \right\}.$$

We use the abbreviated notation $R(\epsilon)$ whenever Φ, $\mathbf{x}$, and $\mathcal{V}$ are clear from the context.

We call the approach of obtaining relevance mappings by finding score vectors $\mathbf{s}$ of small rate with small distortion *rate-distortion-explanations (RDE)*. There are three variants to formulate this as an optimization problem.

We can use the Lagrangian formulation with a regularization parameter $\lambda > 0$

$$\begin{aligned} &\text{minimize} && D(\mathbf{s}) + \lambda \|\mathbf{s}\|_1 \\ &\text{subject to} && \mathbf{s} \in [0,1]^n \end{aligned} \tag{L-RDE}$$

or the rate-constrained formulation with a maximal rate $0 \leq k \leq n$

$$\begin{aligned} &\text{minimize} && D(\mathbf{s}) \\ &\text{subject to} && \|\mathbf{s}\|_1 \leq k, \\ &&& \mathbf{s} \in [0,1]^n \end{aligned} \tag{RC-RDE}$$

or the distortion-constrained formulation with a maximal distortion $0 \leq \epsilon \leq 1$

$$\begin{aligned} &\text{minimize} && \|\mathbf{s}\|_1 \\ &\text{subject to} && D(\mathbf{s}) \leq \epsilon, \\ &&& \mathbf{s} \in [0,1]^n. \end{aligned} \tag{DC-RDE}$$

The three formulations are of course connected, however the relation between λ, k, and ϵ that would lead to comparable solutions is far from trivial. Hence, depending on the application 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 a balance between rate and distortion is desirable. Also, one should keep in mind that the three formulation behave quite differently regarding their numerical optimization.

The Lagrangian formulation (L-RDE) results in a non-convex optimization problem with box constraints and can be solved by any gradient based algorithm capable of handling box constraints, such as Projected Gradient Descent (PGD) [NW06], see Algorithm 1, or L-BFGS-B [Byr+95].

The rate-constrained formulation results in a non-convex optimization problem with box constraints as well as an additional linear constraint. Again it can be solved, for exampled, via Projected Gradient methods, see Algorithm 2. This requires a non-trivial projection step onto the feasible region $C = \{\mathbf{s} \in [0,1]^n \,:\, \|\mathbf{s}\|_1 \leq k\}$ at every iteration. An alternative projection-free first-order method is the Frank-Wolfe (FW) algorithm [FW56] or Conditional Gradients method [LP66], see Algorithm 3. It replaces the projection step of PGD (which is essentially a quadratic sub-problem) with the evaluation of an often computationally cheaper Linear Minimization Oracle (LMO). We refer to Appendix B.1 for a more detailed discussion of the algorithms, projections, LMOs, and choices of hyper-parameters used in our experiments.

Finally, the distortion-constrained formulation yields an optimization problem with linear objective but non-convex constraint. Such general constraints are typically hard to handle. Possible approaches include primal log-barrier methods [NW06] or trust-region SQP interior point methods [BHN99].

The distortion-constrained version is most closely related to the rate function in Definition 4.20 and the Min-$(\mathcal{F},\mathcal{V})$-$\epsilon$-Distortion-Input problem of Section 4.2. However, we found that for high-dimensional problems, such as the STL-10 experiments, cf. Section 4.4.4, (DC-RDE) behaves numerically rather unstable compared to (L-RDE) and (RC-RDE). Therefore, we recommend to use (L-RDE) or (RC-RDE) and will only consider these two variants in our numerical experiments.

Algorithm 1 Projected Gradient Descent (PGD) for (L-RDE)

Input: initial guess $\mathbf{s}^0 \in [0,1]^n$, number of steps T, step sizes $\eta^t > 0$, regularization parameter $\lambda > 0$
Output: an (approximate) stationary point $\mathbf{s}^{\text{opt}}$ of (L-RDE)

for $t \leftarrow 1$ **to** T **do**
  $\mathbf{s} \leftarrow \mathbf{s}^{t-1} - \eta^t(\nabla D(\mathbf{s}^{t-1}) + \lambda \cdot \mathbf{1}_n)$ ▷ Gradient Descent step
  $\mathbf{s}^t \leftarrow \operatorname{argmin}_{\mathbf{v}\in[0,1]^n} \|\mathbf{s} - \mathbf{v}\|_2^2$ ▷ Projection step, see Appendix B.1.3.
end for
return $\mathbf{s}^T$

Algorithm 2 Projected Gradient Descent (PGD) for (RC-RDE)

Input: initial guess $\mathbf{s}^0 \in C = \{\mathbf{s} \in [0,1]^n : \|\mathbf{s}\|_1 \leq k\}$, number of steps T, step sizes $\eta^t > 0$
Output: an (approximate) stationary point $\mathbf{s}^{\text{opt}}$ of (RC-RDE)

for $t \leftarrow 1$ **to** T **do**
  $\mathbf{s} \leftarrow \mathbf{s}^{t-1} - \eta^t \nabla D(\mathbf{s}^{t-1})$ ▷ Gradient Descent step
  $\mathbf{s}^t \leftarrow \operatorname{argmin}_{\mathbf{v}\in C} \|\mathbf{s} - \mathbf{v}\|_2^2$ ▷ Projection step, see Algorithm 5 in Appendix B.1.3
end for
return $\mathbf{s}^T$

Algorithm 3 Frank-Wolfe (FW) algorithm for (RC-RDE)

Input: initial guess $\mathbf{s}^0 \in C = \{\mathbf{s} \in [0,1]^n : \|\mathbf{s}\|_1 \leq k\}$, number of steps T, step sizes $\eta^t \in [0,1]$
Output: an (approximate) stationary point $\mathbf{s}^{\text{opt}}$ of (RC-RDE)

for $t \leftarrow 1$ **to** T **do**
  $\mathbf{v}^t \leftarrow \operatorname{argmin}_{\mathbf{v}\in C} \langle \nabla D(\mathbf{s}^{t-1}), \mathbf{v} \rangle$ ▷ LMO, see Algorithm 6 in Appendix B.1.3
  $\mathbf{s}^t \leftarrow \mathbf{s}^{t-1} + \eta^t(\mathbf{v}^t - \mathbf{s}^{t-1})$
end for
return $\mathbf{s}^T$

4.3.3 Ordered Rate-Distortion Explanations

Instead of first obtaining relevance scores and afterwards retrieving a relevance ordering from them, one could find an optimal ordering directly by solving

$$\begin{aligned} &\text{minimize} && \frac{1}{n-1} \sum_{k \in [n-1]} D(\mathbf{\Pi} \mathbf{p}_k) \\ &\text{subject to} && \mathbf{\Pi} \in S_n, \end{aligned}$$

where S_n denotes the set of $(n \times n)$ permutation matrices and $\mathbf{p}_k = \sum_{j=1}^{k} \mathbf{e}_j$ is the vector of k-ones and $(n-k)$-zeros. Hence, $D(\mathbf{\Pi} \mathbf{p}_k)$ corresponds to the distortion of fixing the k most relevant features (according to $\mathbf{\Pi}$) [3] and the objective aims at minimizing the average distortion across all rates $k \in [n-1]$ simultaneously. [4] We relax this combinatorial problem (S_n is discrete) to

$$\begin{aligned} &\text{minimize} && \frac{1}{n-1} \sum_{k \in [n-1]} D(\mathbf{\Pi} \mathbf{p}_k) && \text{(Ord-RDE)} \\ &\text{subject to} && \mathbf{\Pi} \in B_n, \end{aligned}$$

where S_n is replaced with its convex hull, i.e., the Birkhoff polytope $B_n = \operatorname{conv}(S_n)$ of doubly stochastic $(n \times n)$ matrices. Similar to (RC-RDE), this results in an optimization problem with non-convex objective over a convex and compact feasible region, which can be solved with the Frank-Wolfe algorithm (or a stochastic version of it [HL16] if n is large and evaluating the complete sum in (Ord-RDE) is too expensive, see Algorithm 4). There is no exact projection method specific to the Birkhoff polytope, see Appendix B.1, hence we do not consider PGD for solving (Ord-RDE).

Algorithm 4 Stochastic Frank-Wolfe (SFW) algorithm for (Ord-RDE)

Input: initial guess $\mathbf{\Pi}^0 \in B_n$, number of steps T, step sizes $\eta^t \in [0,1]$, batch sizes $b^t \in [n-1]$, momentum factors $\rho^t \in [0,1]$
Output: an (approximate) stationary point $\mathbf{\Pi}^{\text{opt}}$ of (Ord-RDE)

$\mathbf{M}_0 \leftarrow \mathbf{0}_{n \times n}$
for $t \leftarrow 1$ **to** T **do**
  sample $k_1, \dots, k_{b^t}$ i.i.d. uniformly from $[n-1]$
  $\mathbf{G}^t \leftarrow \frac{1}{b^t} \sum_{j=1}^{b^t} \nabla D(\mathbf{\Pi}^{t-1} \mathbf{p}_{k_j}) \mathbf{p}_{k_j}^\top$ ▷ Gradient estimate
  $\mathbf{M}^t \leftarrow \rho^t \mathbf{M}^{t-1} + (1-\rho^t) \mathbf{G}^t$ ▷ Momentum update
  $\mathbf{V}^t \leftarrow \operatorname{argmin}_{\mathbf{V} \in B_n} \langle \mathbf{M}^t, \mathbf{V} \rangle$ ▷ Linear Minimization Oracle, see Appendix B.1
  $\mathbf{\Pi}^t \leftarrow \mathbf{\Pi}^{t-1} + \eta^t (\mathbf{V}^t - \mathbf{\Pi}^{t-1})$
end for
return $\mathbf{\Pi}^T$

A solution $\mathbf{\Pi}^{\text{opt}}$ to (Ord-RDE) is a convex combination of permutation matrices (the vertices of B_n). It can be used to obtain mappings for specific rates via $\mathbf{\Pi}^{\text{opt}} \mathbf{p}_k$, which we

[3] If $\pi \in \operatorname{Sym}(n)$ denotes the permutation represented by $\mathbf{\Pi}$, then $D(\mathbf{\Pi} \mathbf{p}_k)$ exactly corresponds to the term $D(\pi([k]))$ in Definition 4.16.
[4] For $k = n$ the distortion $D(\mathbf{\Pi} \mathbf{p}_n)$ will always be zero, hence this term can be omitted from the averaging.

interpret as a convex combination of the respective k most relevant components according to each permutation contributing to a convex decomposition of $\mathbf{\Pi}^{\text{opt}}$. From now on we refer to $\mathbf{\Pi}^{\text{opt}}\mathbf{p}_k$ with $k \in [n-1]$ as the single-rate mappings associated to $\mathbf{\Pi}^{\text{opt}}$.

One should note that, in contrast to (RC-RDE), a straightforward approach to solving (Ord-RDE) is not feasible for large-scale problems: optimizing over matrices in $\mathbb{R}^{n\times n}$ instead of vectors in $\mathbb{R}^n$ results in increased computational costs, both in terms of memory requirements and runtime (see also the descriptions of the LMOs in Appendix B.1). However, this might in part be remedied by a clever and more memory-efficient representation of iterates.[5]

Another possibility to overcome this limitation is to emulate a similar multi-rate strategy that relies solely on the rate-constrained formulation: we can separately solve (RC-RDE) for all $k \in \mathcal{K}$ for some $\mathcal{K} \subseteq [n-1]$ and combine these solutions, e.g., by averaging, to obtain relevance scores

$$\mathbf{s} = \frac{1}{|\mathcal{K}|} \sum_{k\in\mathcal{K}} \left\{ \begin{array}{ll} \operatorname{argmin} & D(\mathbf{s}) \\ \text{subject to} & \|\mathbf{s}\|_1 \leq k, \\ & \mathbf{s} \in [0,1]^n, \end{array} \right\} \tag{MR-RDE}$$

that take multiple rates into account. Consequently, this effectively yields an induced ordering of the variables according to their relevance across all rates (if a variable is contained in the solutions for many of the rates then it contributes more to the averaging of relevance maps over $\mathcal{K}$ and should be considered more relevant than a variable that is only contained in a few solutions, hence it should come earlier in the ordering).

In summary, we can interpret the different RDE variants in terms of their objective regarding the relevance ordering evaluation: solving a single (RC-RDE) problem aims at optimizing the rate-distortion function, i.e., achieving low distortion, at a single rate. This might lead to suboptimal results at other rates. In contrast, (Ord-RDE) aims at directly optimizing the relevance ordering and thus optimizes the distortion for all rates simultaneously on average. Finally, (MR-RDE) combines single-rate solutions spread across the range of considered rates and thus approximately also aims at optimizing the distortion everywhere by minimization at well-chosen attachment points.

4.3.4 Assumed Density Filtering

In order to solve any of the RDE problem variants by first-order methods we need an efficient way of evaluating $D(\mathbf{s})$ and its gradient. We will give a brief overview of one possibility here. A more detailed discussion is given in Chapter 5. For the remainder of this section we will only consider the squared distance $\operatorname{dist}(a,b) = (a-b)^2$, which satisfies both properties of Assumption 4.5.

[5] The t-th iterate $\mathbf{\Pi}_t$ of the Frank-Wolfe algorithm is a convex combination of an active set of at most t permutations (corresponding to vertices of B_n). Storing these together with the convex weights allows to effectively recover the iterate but reduces the memory requirement from $O(n^2)$ to $O(tn)$ (assuming $t < n$). For algorithm variants keeping track of an active set anyways, such as the Away-Step Frank-Wolfe algorithm, see Appendix B.1, there is no computational overhead. Similarly, a sparse matrix representation reduces the memory requirement to $O(\#\text{non-zero components})$. However, the bottleneck of our current implementation is the LMO evaluation. The gradients will still be dense matrices. In fact, they are sums of rank-1 matrices. We leave a study of the practical benefits of exploiting this structure for the LMO evaluation to future research.

Remark 4.21. For the squared distance $\mathrm{dist}(a,b) = (a-b)^2$ the distortion can be rewritten in a bias-variance decomposition

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

In this case the expected distortion is determined by the first and second moment of the output layer distribution of the neural network classifier. 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 n and the distribution $\mathcal{V}$ sampling might be infeasible. Thus, we focus on a second possibility, which takes the specific structure of Φ more into account.

From Definition 4.18 it is straight-forward to obtain the first and second moment

$$\begin{aligned}\mathbb{E}_{\mathbf{z}\sim\mathcal{V}_\mathbf{s}}\left[\mathbf{z}\right] &= \mathbf{x}\odot\mathbf{s} + \mathbb{E}_{\mathbf{e}\sim\mathcal{V}}\left[\mathbf{e}\right]\odot(\mathbf{1}_n-\mathbf{s})\\ \mathbb{V}_{\mathbf{z}\sim\mathcal{V}_\mathbf{s}}\left[\mathbf{z}\right] &= \mathrm{diag}(\mathbf{1}_n-\mathbf{s})\mathbb{V}_{\mathbf{e}\sim\mathcal{V}}\left[\mathbf{e}\right]\mathrm{diag}(\mathbf{1}_n-\mathbf{s})\end{aligned}$$

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 Remark 4.21 we utilize the layered structure of Φ and propagate the distribution of the neuron activations through the network. For this, we use an approximation method, 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]. 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 (with respect to KL-divergence). This amounts to matching the first two moments of the distribution [Min01]. A discussion of why it is unavoidable to use an approximate method as well as a more detailed description of the general ADF approach is given in Chapter 5. 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 approximate 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.,

$$\begin{aligned}\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}\left[\mathbf{W}\mathbf{z}+\mathbf{b}\right] &= \mathbf{W}\boldsymbol{\mu}+\mathbf{b},\\ \mathbb{V}_{\mathbf{z}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}\left[\mathbf{W}\mathbf{z}+\mathbf{b}\right] &= \mathbf{W}\boldsymbol{\Sigma}\mathbf{W}^\top,\end{aligned} \tag{4.1}$$

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

$$\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}\left[\varrho(\mathbf{z})\right] = \boldsymbol{\sigma}\odot f(\boldsymbol{\eta}) + \boldsymbol{\mu}\odot F(\boldsymbol{\eta}).$$

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 (4.1) to

$$\mathbb{V}^{\text{diag}}_{\mathbf{z}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}\left[\mathbf{W}\mathbf{z}+\mathbf{b}\right] = (\mathbf{W}\odot\mathbf{W})\boldsymbol{\sigma},$$

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

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

Or, we use an approximation for the full covariance matrix

$$\mathbb{V}_{\mathbf{z}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}\left[\varrho(\mathbf{z})\right] \approx \mathbf{N}\boldsymbol{\Sigma}\mathbf{N} \tag{4.2}$$

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 factorization $\mathbb{V}_{\mathbf{z}\sim\mathcal{V}_\mathbf{s}}[\mathbf{z}] \approx \mathbf{Q}\mathbf{Q}^\top$ at the input layer with $\mathbf{Q}\in\mathbb{R}^{n\times r}$ for $r \ll n$ (for example half of a truncated singular value decomposition), then the symmetric update (4.2) 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. More details on low-rank approximations of covariance matrices can be found in Appendix B.7.

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 automatic differentiation (backpropagation). We investigate the *diagonal* as well as the *low-rank* approximation to the covariance matrix in our numerical inquiry. However, for brevity we only report results for the diagonal approximation in the next section and leave a comparison to the low-rank approximation to Appendices B.2.1 and B.2.2.

4.4 Evaluating and Comparing Explanations

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 can not 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+15], Deep Taylor decompositions [MSM18], Sensitivity Analysis [SVZ13], SmoothGrad [Smi+17], Guided Backprop [Spr+15], SHAP [LL17], and LIME [RSG16]. In the following, we keep the description of the numerical experiments and the choice of all hyper-parameters brief and refer to Appendices B.3 to B.6 for all details. Here, we only show results for RDE with the diagonal ADF approximation for evaluating the distortion functional. A selection of corresponding results for low-rank approximations is presented in Appendix B.2. Further, also additional results for (RC-RDE) using variations of the vanilla Frank-Wolfe (FW) algorithm and Projected Gradient Descent (PGD) can be found there.

4.4.1 Invariance to Input Transformations

Kindermans et al. propose that relevance mapping methods should be invariant or covariant to simple transformations in the input domain, e.g., mean shifts or variable permutations, if the transformation can be compensated by the first network layer [Kin+17]. However, they observe 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. We present the result and proof for (L-RDE) but the same argument carries over to (RC-RDE),(DC-RDE),(MR-RDE), and (Ord-RDE).

Lemma 4.22. *Let $\mathbf{s}^* \in [0,1]^n$ be an L-RDE relevance score for Φ and $\mathbf{x}$ with respect to $\mathcal{V}$, i.e., a solution of (L-RDE). Further let $\mathbf{P} \in S_n$ be a permutation matrix, $\mathbf{\Lambda} \in (0,1]^{n\times n}$ a diagonal scaling matrix, $\mathbf{m} \in [0,1]^n$ a shift vector, and $\widetilde{\mathbf{x}} = h(\mathbf{x}) = \mathbf{P}\mathbf{\Lambda}\mathbf{x} + \mathbf{m}$ 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 $\mathbf{s} \mapsto \mathbf{P}\mathbf{s}$ is a bijective transform on $[0,1]^n$, it suffices to show that

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

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

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

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. □

4.4.2 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

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

that checks binary strings of length n for the existence of a block of k consecutive ones, similar to the example shown in Figure 3.2.

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 $\frac{1}{4}$.

We can construct a ReLU neural network $\Phi\colon [0,1]^n \to [0,1]$ that interpolates Ψ, see Appendix B.3 for details. Relying on the connection between the binary and continuous setting established in Section 4.2 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 $n = 16$ 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 4.3(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 4.3(b).

In this setting we compare (L-RDE) and (RC-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. The rate for (RC-RDE) is $k = 6$ and intentionally chosen larger than the optimal rate $k = 5$ that allows for zero distortion in order to see how the "excess" relevance is distributed across the signal. It also better corresponds to sparsity of the (L-RDE) solution that is indirectly adapted to a specific rate via the regularization parameter λ.

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 4.3.

4.4.3 An8flower Benchmark Dataset

Oramas, Wang, and Tuytelaars propose 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 color of their flower or stem. The dataset also provides binary masks marking the colored part of

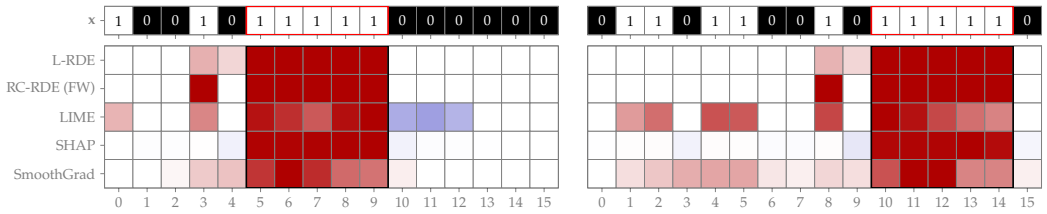


Figure 4.3: **Synthetic Binary Strings – Relevance Maps.** Relevance mappings generated by several methods for two binary strings. The left string (a) contains one block of five consecutive ones whereas the right string (b) contains one complete and one incomplete block. The colormap indicates positive 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. The rate constraint for RC-RDE is $k = 6$ ($n = 16$). Additional results for the diagonal and low-rank variants of RDE and for different FW variants for RC-RDE are shown in Figure B.1 in Appendix B.

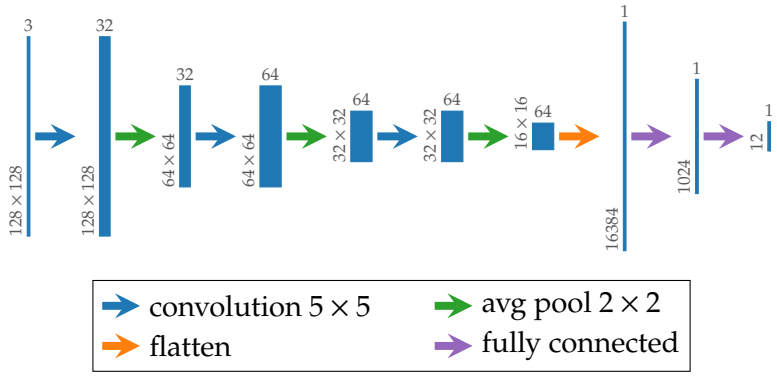


Figure 4.4: **An8flower – Network Architecture.** Illustration of the convolutional neural network architecture for the An8flower task. All convolutions and the first fully connected layer are followed by ReLU activations. The final fully connected layer is followed by a softmax activation.

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 classifier 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, as illustrated in Figure 4.4) 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 Figure 4.5. The mappings are calculated for the pre-softmax score of the class with the

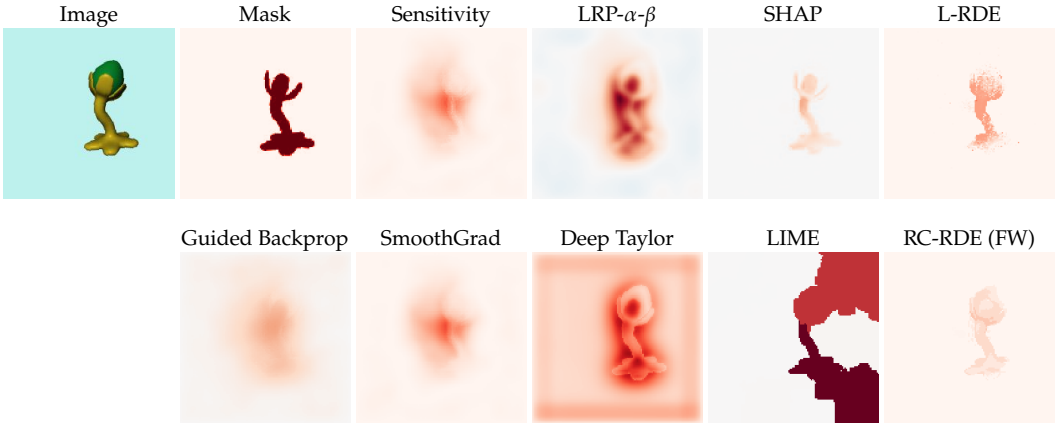


Figure 4.5: **An8flower – Relevance Maps.** Relevance mappings generated by several methods for an image from the An8flower benchmark dataset classified as *yellow stem* by our network. The colormap indicates positive relevances as red and negative relevances as blue. The rate constraint for RC-RDE is $k = 1000$ ($n = 49152$). Additional results for the diagonal and low-rank variants of RDE and for different FW variants for RC-RDE are shown in Figure B.2 in Appendix B.

Table 4.1: **An8flower – Correlation Comparison.** 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). The rate constraint for RC-RDE is $k = 1000$ ($n = 49152$). Additional results for the diagonal and low-rank variants of RDE and for different FW variants for RC-RDE are shown in Table B.1 in Appendix B.

	Sensitivity	SmoothGrad	SHAP	Guided Backprop	LRP-α-β	Deep Taylor	LIME	L-RDE	RC-RDE (FW)
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	**0.35 ± 0.20**
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	**0.25 ± 0.16**

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 summarized in Table 4.1. Both (L-RDE) and (RC-RDE) achieve the best results for both the Pearson correlation coefficient as well as the Jaccard index compared to the other baseline methods. Overall, (RC-RDE) performs best.

4.4.4 Relevance Ordering Test

Next, we generate relevance mappings for grayscale images of handwritten digits from the MNIST dataset [LeC+98] as well as color 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 4.3.1, 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

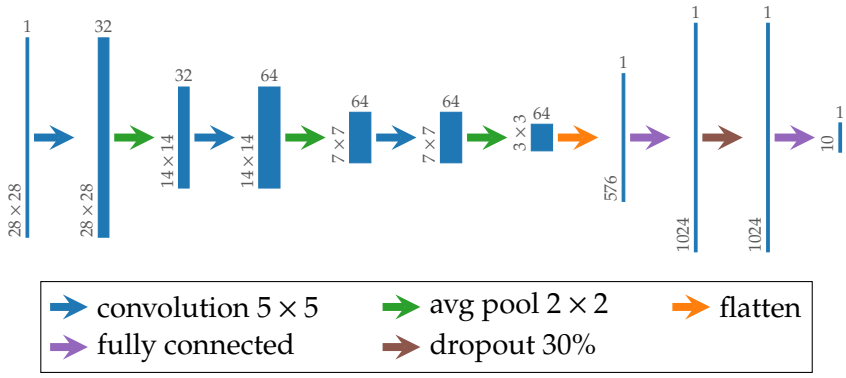


Figure 4.6: **MNIST – Network Architecture.** Illustration of the convolutional neural network architecture for the MNIST task. All convolutions and the first fully connected layer are followed by ReLU activations. The final fully connected layer is followed by a softmax activation.

original input, and observe the change in the classifier prediction. This is then averaged over multiple random input samples. A good relevance mapping will lead to a fast convergence to the classification prediction 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 4.3.1.

MNIST

We trained a convolutional neural network (three convolution layers each followed by average-pooling and finally two fully-connected layers and softmax output, as illustrated in Figure 4.6) 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 4.7. It shows (RC-RDE) mappings for FW and PGD at different selected rates k, as well as respective (MR-RDE) mappings with $\mathcal{K} = \{50, 100, 150, 200, 250, 300, 350, 400\}$ and an (Ord-RDE) mapping compared to the other baseline methods.[6] All mappings are calculated for the pre-softmax score of the class with the highest activation. All RDE variants generate similar results and highlight an area at the top as relevant, that distinguishes a six from, for example, the digits zero and eight. Both FW and PGD are robust across varying rates, in the sense that solutions for larger rates add additional features to the relevant set without significantly modifying the features that were already considered relevant at smaller rates. The (L-RDE) solution is most similar to the (RC-RDE) solutions at rate $k = 100$.

The relevance-ordering test results are shown in Figure 4.8 for two different performance measures (distortion and classification accuracy). All RDE methods result in a fast drop in the distortion (respectively a fast rise in the accuracy), indicating that the relevant features were correctly identified. They clearly outperform the comparison relevance mapping methods. The FW solutions perform slightly better than the PGD solutions.

[6] Images are 28×28 grayscale, hence $n = 28 \cdot 28 = 784$.

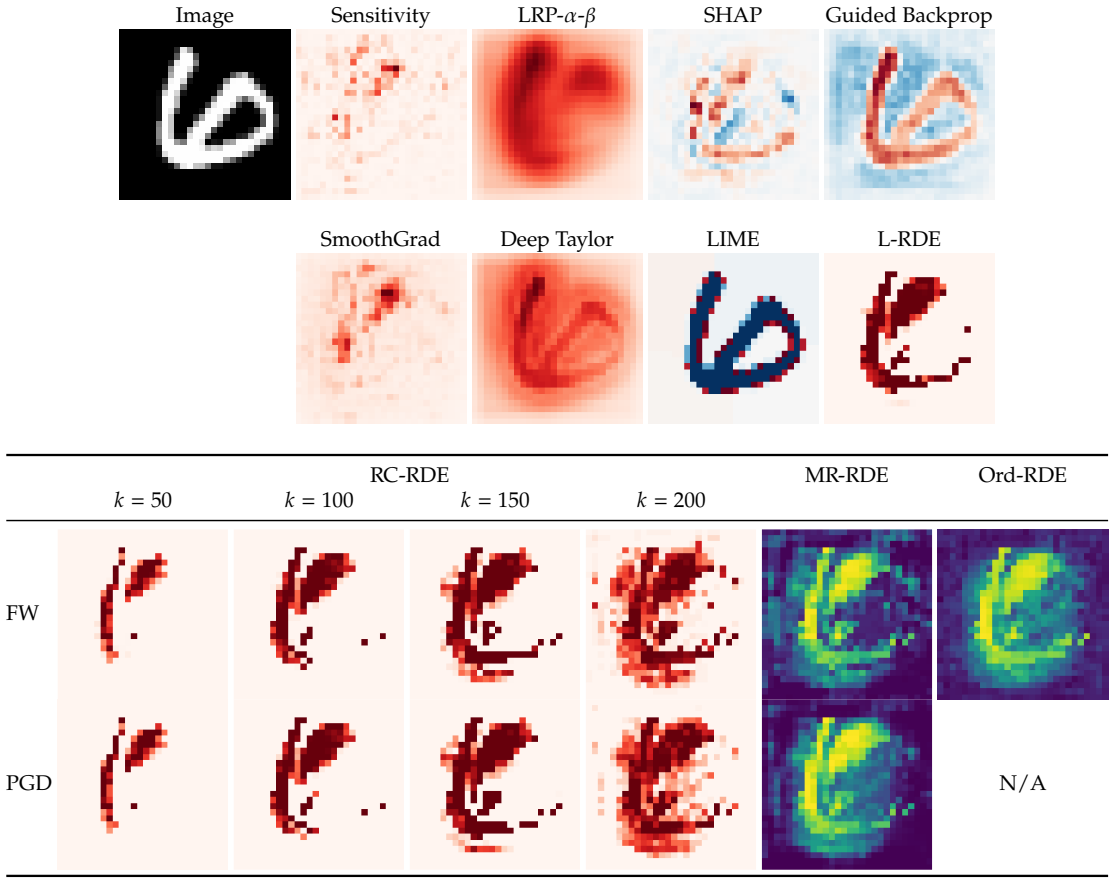


Figure 4.7: **MNIST – Relevance Maps.** Relevance mappings generated by several methods for an image from the MNIST dataset classified as *digit six* by our network. The colormap indicates positive relevance scores as red and negative relevances as blue. Multi-rate (MR-RDE) and ordering (Ord-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow). Additional results for different FW variants for RDE are shown in Figure B.3 in Appendix B.

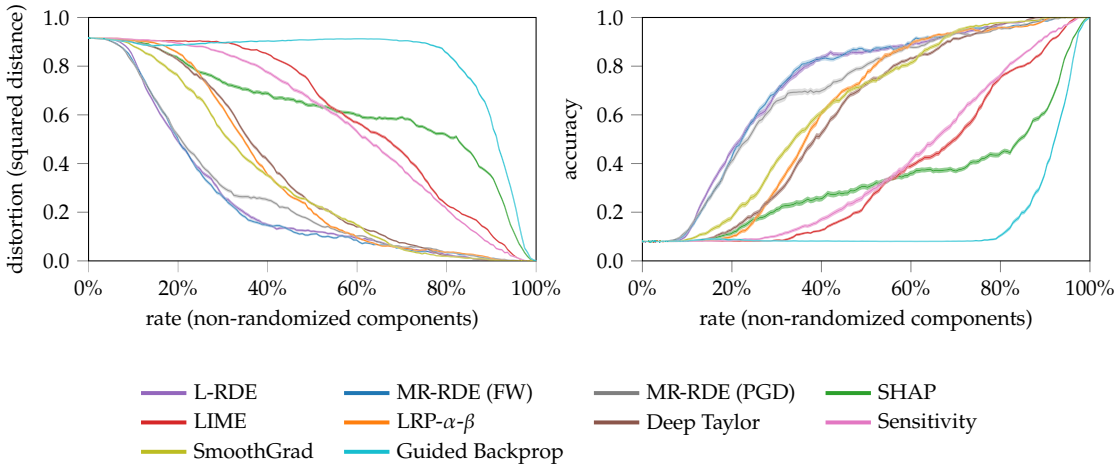


Figure 4.8: **MNIST – Ordering Comparison.** Relevance ordering test results of several methods for the MNIST dataset using squared distance (left) and classification accuracy (right) as performance measure. An average result over 50 images from the test set (5 random images per class) and 512 random input samples per image is shown (shaded regions mark ± standard deviation). Additional results for different FW variants for MR-RDE are shown in Figure B.4 in Appendix B.

STL-10

We use a VGG-16 network [SZ14], as illustrated in Figure 4.9, 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 two example images of a monkey and a horse are shown in Figures 4.10 and 4.11. They show (RC-RDE) mappings for FW and PGD at different selected rates k, as well as respective (MR-RDE) mappings with $\mathcal{K} = \{2000, 4000, 6000, 8000, 10000, 14000, 18000, 22000, 26000, 30000, 34000, 38000, 42000, 46000, 50000\}$ compared to the other baseline methods. [7] Due to the large size of problem instances, we do not show (Ord-RDE) solutions in this experiment. As before the mappings are calculated for the pre-softmax score of the class with the highest activation. All RDE methods generate similar results and highlight parts of the face and body of the monkey as relevant at small rates. Increasingly large parts of the body and tail of the monkey are added to the relevant set at higher rates. Similarly, parts of the head and front legs of the horse are marked relevant first and larger parts of its body get added at higher rates. As before, the results are robust across varying rates. Compared to MNIST, the difference between the FW and PGD solutions and between the Lagrangian and the rate-constrained formulations are more pronounced for STL-10. The sparsity of the (L-RDE) solution is most comparable to the (RC-RDE) solutions at rate $k = 2000$. However, it is less concentrated at a specific part of the body of the animals, especially for the horse image.

Figure 4.12 shows the relevance ordering test results for two different performance measures (distortion and classification accuracy). All RDE methods result in a fast drop in the distortion (respectively a fast rise in the accuracy), indicating that the relevant features were correctly identified. Again, they clearly outperform the other relevance

[7] Images are resized to 224×224 with three color channels, hence $n = 3 \cdot 224 \cdot 224 = 150528$. Mappings are visualized as a single channel heatmap that averages relevance scores across color channels.

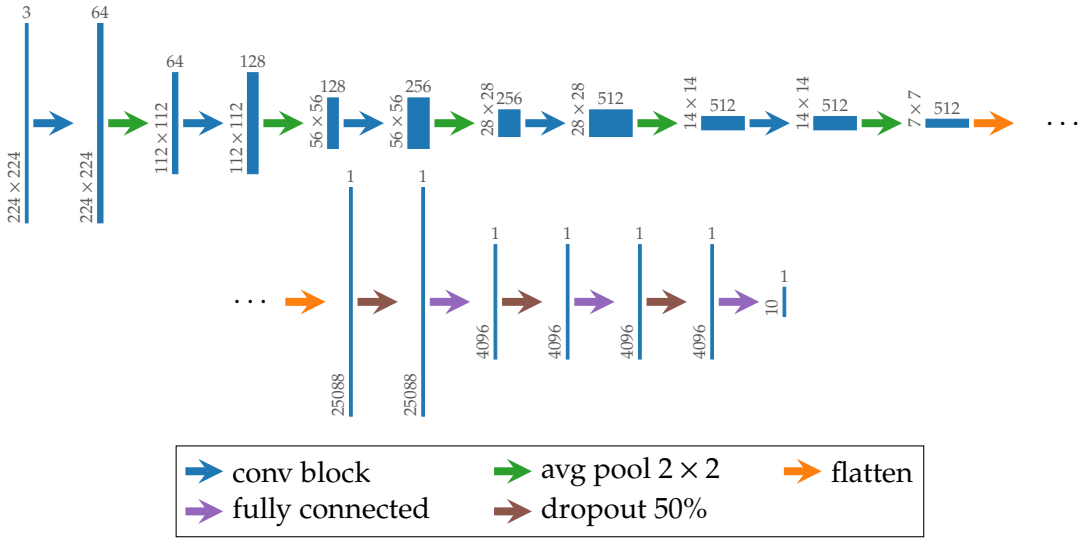


Figure 4.9: **STL10 – Network Architecture.** Illustration of the VGG-16 based convolutional neural network architecture for the STL-10 task. The first two convolutional blocks each consists of two 3×3 convolutions with ReLU activation. The last three convolutional blocks each consists of three 3×3 convolutions with ReLU activation. The first two fully connected layer are followed by ReLU activations. The final fully connected layer is followed by a softmax activation.

attribution methods, especially at the lower rates. However, we observe that RDE does not obtain the smallest distortions across all rates. The non-monotone behavior of the Frank-Wolfe (RC-RDE) curve for rates between 30% and 70% can be explained by the fact that the maximal considered rate in $\mathcal{K}$ corresponds to about 33% of the total number of components. Interestingly, PGD does not show the same behavior, which indicates that PGD is not able to solve the (RC-RDE) problems to the same extent as FW for low rates. The (L-RDE) solutions are (indirectly) adapted to only a single rate and show a similar non-monotone behavior. Unlike for MNIST, there is a considerable difference between (L-RDE) and (MR-RDE). The latter outperforms the former across all rates.

4.5 Discussion

We extended the concept of δ-relevance, a probabilistic version of prime implicants, for the explanation of classifier decisions from a binary to a continuous setting.

The resulting rate-distortion framework allows us to formulate a concrete objective that relevance maps should solve and to analyze the complexity of this problem. Furthermore, it enables us to compare different relevance maps in a quantitative numerical evaluation.

Our definition of the relevance of input components and our approach of generalizing the concept of prime implicants to the neural network setting is by no means the only possibility. We made some careful but maybe not always obvious choices, e.g., regarding the reference distributions $\mathcal{V}$ and $\mathcal{V}_S$. In the following we want to discuss these choices as well as point out some potential limitations and extensions of our analysis.

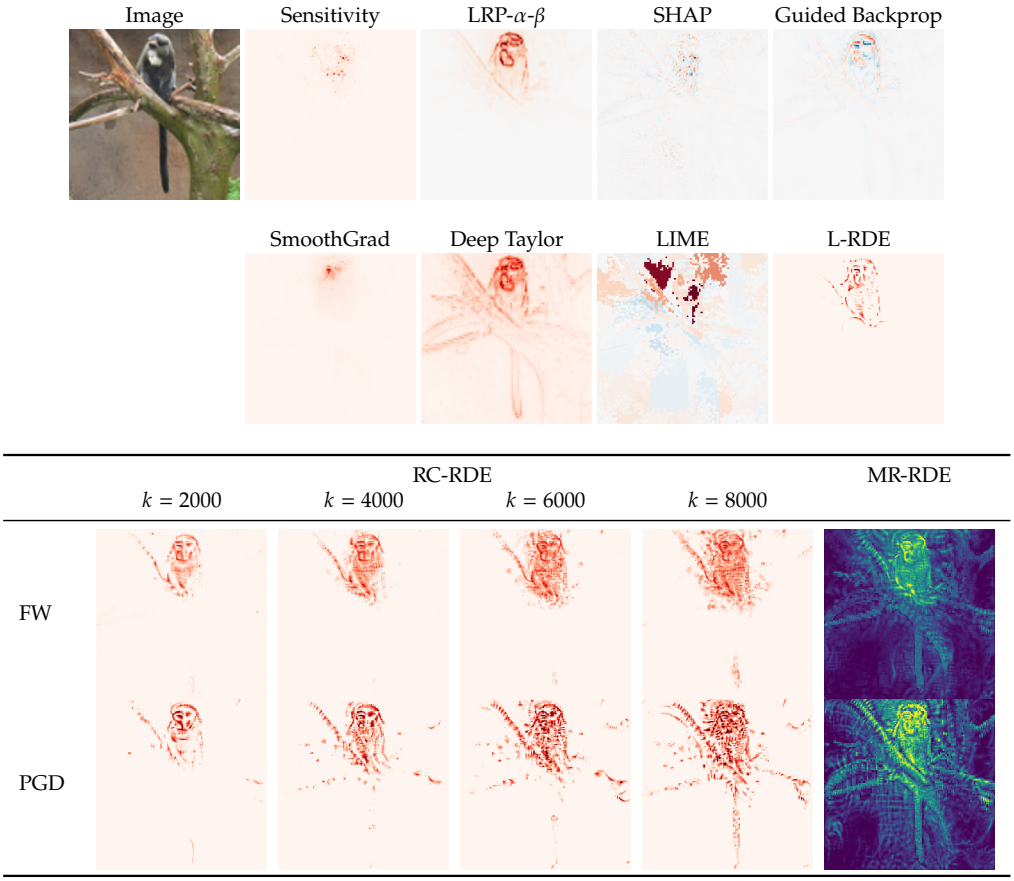


Figure 4.10: **STL10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *monkey* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow). Additional results for different FW variants for RC-RDE are shown in Figure B.10 in Appendix B.

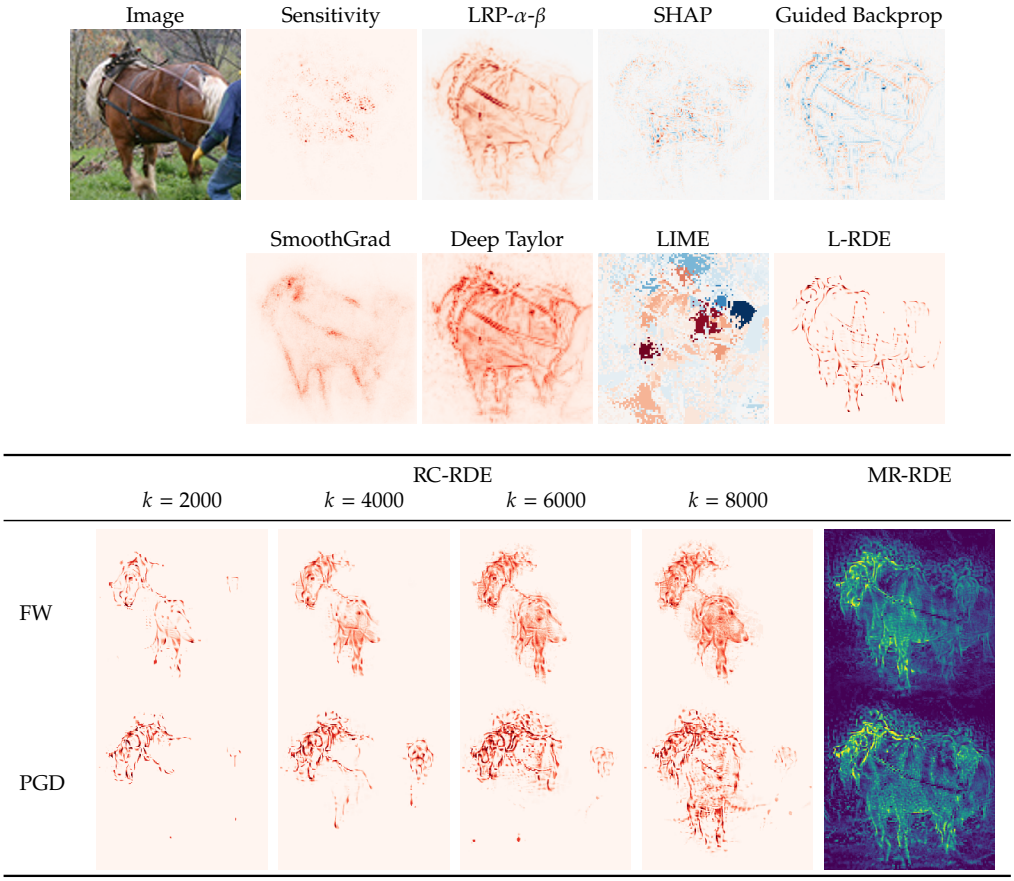


Figure 4.11: **STL10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *horse* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow). Additional results for different FW variants for RC-RDE are shown in Figure B.14 in Appendix B.

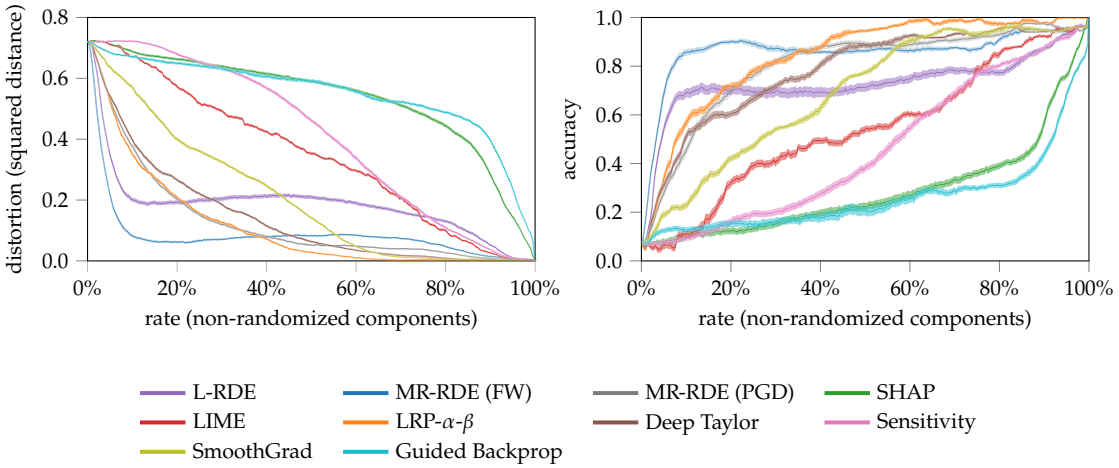


Figure 4.12: **STL10 – Ordering Comparison.** Relevance ordering test results of several methods for the STL-10 dataset using squared distance (left) and classification accuracy (right) as performance measure. An average result over 50 images from the test set (5 random images per class) and 64 random input samples per image is shown (shaded regions mark ± standard deviation). Additional results for different FW variants for MR-RDE are shown in Figure B.9 in Appendix B.

4.5.1 Sufficiency and Necessity of Finding Relevant Sets

Knowing the relevant set of input variables is not always sufficient to fully explain every aspect of a classifier decision. In particular, further information might be needed to completely reveal the workings of a classifier and to make it accessible to humans, rendering the classifier transparent. However, we believe that finding the relevant components is a necessary first step toward this goal. Any good explanation method should at least solve (a variation of) the relevant set problem as a subproblem. Thus, explaining neural network decisions in general is at least as hard as finding the relevant input components and our hardness result carries over to other, possibly more elaborate, notions of "explaining".

4.5.2 Non-Uniform Distributions

We use the uniform distribution $\mathcal{U}([0,1]^n)$ 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 $[0,1]^n$. We can choose μ as a product of any independent one-dimensional measures μ_j for the individual input components as long as for every $j \in [n]$ and $0 < \eta \leq 1$ there exist lower and upper thresholds $a_j, b_j \in [0,1]$ with $a_j < b_j$ such that

$$\mu_j([0,a_j]) = \mu_i([b_j,1]) \qquad \text{and} \qquad \mu_j([a_j,b_j]) \leq \eta.$$

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.

4.5.3 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 $\mathbf{z}$ to be deterministically given by $\mathbf{z}_S = \mathbf{x}_S$ on S and distributed according to $\mathbf{z}_{S^c} = \mathbf{e}_{S^c}$ with $\mathbf{e} \sim \mathcal{V}$ on the complement S^c. This means that the resulting distribution $\mathcal{V}_S$ of $\mathbf{z}$ corresponds to $\mathcal{V}$ marginalized over all components in S. One might be tempted to condition on the given components $\mathbf{x}_S$ instead of marginalizing. But this could actually be detrimental to uncover how the classifier operates [JMB20]. 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+16]. 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 recognize 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 reasoning of the network.

4.6 Conclusion

Our complexity result shows that no efficient algorithm can provably answer the questions *Q1* and *Q2*. Even more, we showed that no non-trivial error bounds for computing the size of a minimal set of relevant input components can be given, since the problem is not efficiently approximable. This holds as long as the considered neural networks can represent arbitrary logical functions, which is true for ReLU networks (even with bounded weights and depth).

Of course, this is a worst-case analysis. It is conceivable that the problem could be made feasible by introducing more subtle restrictions on the neural networks and the inputs that depend on the actual data structures on which the networks have been trained. However, these are not yet well enough understood. As long as this is the case, we have to rely on heuristic solution methods. In fact, many non-linear optimization problems are NP-hard in general and yet performed successfully on a regular basis. In these situations a thorough numerical evaluation of the solution methods is crucial, as no performance guarantees can be proven.

In this spirit, we have proposed one solution strategy for the relevance mapping problem and demonstrated that it performs better than existing alternatives in a wide range of quantitative comparison tests.

5

The Necessity of Using Approximate ADF

An important theoretical as well as computational challenge is calculating the distribution of outputs of a neural network from a given distribution of inputs. This task is a crucial ingredient for uncertainty quantification, explainable AI and Bayesian learning. In particular, it is a key component of our heuristic solution strategy to the relevance mapping problem in Chapter 4.

More precisely, the task is to describe the distribution μ_{out} of

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

where Φ 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

$$\mu_{\text{out}} = \Phi_* \mu_{\text{in}} \tag{5.1}$$

of μ_{in} with respect to the transformation Φ. 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 (ADF) [GR18], see also Section 4.3.4, and in the case of Bayesian neural networks the more general framework of Expectation Propagation (EP) [JNV14; SHM14] which contains ADF as a special case. Both methods were originally developed for general large-scale Bayesian inference problems [Min01; BK98].

Recall that ADF makes use of the layered feed-forward structure of neural networks: If we write $\Phi = \Phi_L \circ \Phi_{L-1} \circ \cdots \circ \Phi_2 \circ \Phi_1$ as a composition of its layers Φ_j, then (5.1) decomposes as

$$\begin{aligned} \mu_0 &= \mu_{\text{in}}, \\ \mu_j &= (\Phi_j)_* \mu_{j-1} \quad \text{for} \quad j = 1, \dots, L, \\ \mu_{\text{out}} &= \mu_L. \end{aligned} \tag{5.2}$$

The task can then be described in terms of investigating how the distributions are propagated through individual layers of the neural network. 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 iteratively obtaining μ_{out}. Such a family, if expressible 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 Gaussian distributions, or more generally exponential families. While the Gaussian family is invariant under purely affine linear layers, this is not the case for non-linear activations. Hence, the method relies on a projection step back

onto the Gaussian family after each network layer. In general, for any chosen family of probability distributions, (5.2) is replaced by

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

where $\mathrm{proj}(\cdot)$ is a suitable projection. In this case only $\mu_{\mathrm{out}} \approx \Phi_* \mu_{\mathrm{in}}$ 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 realized in a meaningful way—a fact that, to the best of our knowledge, remained unproven. In this chapter we will fill this gap.

We give a complete characterization 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 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. As a result, we provide a theoretical underpinning for statistical inference schemes such as ADF for neural networks.

Applications of ADF for Networks

Calculating output distributions of neural networks is a basic and central problem to make neural networks more applicable. Good performance on test data alone is not sufficient to ensure reliability. High-stakes applications such as, e.g., medical imaging or autonomous driving, make questions regarding uncertainty quantification (UQ), robustness, and explainability of network decisions inescapable [Oal+20; BBK19; Zha+18; Hol+17; MKG18; KC17].

Uncertainty Quantification. Studying the propagation of input uncertainties through an already trained (deterministic) network is used to analyze the reliability of their predictions and their sensitivity to changes in the data. ADF is used toward this goal, e.g., for sigmoid networks for automatic speech recognition [AN11; AAT14] and ReLU networks for image classification and optical flow [GR18].

Explainable Artificial Intelligence. So far, neural networks are mostly considered opaque "black-box" models. Recent approaches to make them human-interpretable have focused on the crucial subtask of finding the most relevant input variables for a given prediction. A probabilistic formulation of this task for classification networks was introduced in Chapters 3 and 4, see also [FV17]. Our heuristic solution strategy for deep ReLU networks in Section 4.3.2 is based on ADF with Gaussian distributions.

Bayesian Networks. Instead of point estimates, Bayesian neural networks (BNN) learn a probability distribution over the network weights through Bayesian inference. BNNs provide an inbuilt regularization by choosing appropriate priors. They naturally lead to network compression when using sparsity-promoting priors or encoding weights of high variance with less precision. ADF with Gaussian distributions has been employed for the fast calculation of posteriors during the training of BNNs with sigmoidal [OW98] and ReLU [HA15; SCC17] activations. Wu et al. use a related moment-propagation approach for deterministic variational inference in BNNs with ReLU or Heaviside activations [Wu+19]. Another approach was taken by Wang and Manning and relies on layer-wise Gaussian approximations to yield a fast alternative to dropout regularized training [WM13].

5.1 Characterization of Invariant Families of Distributions

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

Recall that $\mathcal{B}(\mathbb{R}^n)$ denotes the Borel σ-Algebra on $\mathbb{R}^n$ and $\mathcal{D}(\mathbb{R}^n)$ is the set of Radon probability measures on $\mathbb{R}^n$. We equip $\mathcal{D}(\mathbb{R}^n)$ with the Prokhorov metric [Pro56]

$$\begin{aligned} d_P(\mu,\nu) = \inf\{\epsilon > 0 \,:\, &\mu(B) \leq \nu(B^\epsilon) + \epsilon \text{ and} \\ &\nu(B) \leq \mu(B^\epsilon) + \epsilon \text{ for any } B \in \mathcal{B}(\mathbb{R}^n)\}, \end{aligned}$$

where $B^\epsilon = \{\, \mathbf{x} \in \mathbb{R}^n \,:\, \exists \mathbf{y} \in B \text{ with } \|\mathbf{x}-\mathbf{y}\| < \epsilon \,\}$ is the open ϵ-neighborhood of a Borel set $B \in \mathcal{B}(\mathbb{R}^n)$.[1]

Definition 5.1. Let $\Omega \subseteq \mathbb{R}^d$ and $p\colon \Omega \to \mathcal{D}(\mathbb{R}^n)$. We call p a d-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 parametrized in multiple ways and even the number d 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 parametrization p. All the following results are stated in terms of this parametrization.

Example 5.2 (2-dimensional Gaussian Family)**.** Consider the parameter space

$$\Omega = \mathbb{R}^2 \times \{(\sigma_1,\sigma_2,\sigma_3) \in \mathbb{R}^3 \,:\, \sigma_1,\sigma_2 \geq 0 \text{ and } \sigma_1\sigma_2 - \sigma_3^2 \geq 0\} \subseteq \mathbb{R}^5,$$

and

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

[1] The original definition by Prokhorov only considers closed sets B, however this results in the same metric.

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 5.3. Let $\Omega \subseteq \mathbb{R}^d$ and $p\colon \Omega \to \mathcal{D}(\mathbb{R}^n)$ be a family of probability distributions and $\Phi\colon \mathbb{R}^n \to \mathbb{R}^n$ any measurable function. Then the family is called Φ-*invariant*, if for any $\theta \in \Omega$ the pushforward of the measure $p(\theta)$ under Φ is again in the family, i.e.,

$$p(\omega) = \Phi_* p(\theta)$$

for some $\omega \in \Omega$. For a collection

$$\mathcal{F} \subseteq \{\, \Phi\colon \mathbb{R}^n \to \mathbb{R}^n \;:\; \Phi \text{ measurable} \,\}$$

of measurable functions it is called $\mathcal{F}$*-invariant*, if it is Φ-invariant for all $\Phi \in \mathcal{F}$.

We are interested in the special case of invariance with respect to ReLU layers of neural networks.

Definition 5.4. Let $\Omega \subseteq \mathbb{R}^d$ and $p\colon \Omega \to \mathcal{D}(\mathbb{R}^n)$ be a family of distributions. The family is called *ReLU-invariant*, if it is $\mathcal{F}$-invariant for the collection

$$\mathcal{F} = \mathcal{NN}^n_{1,\varrho} = \{\, \Phi\colon \mathbb{R}^n \to \mathbb{R}^n : \mathbf{x} \mapsto \varrho(\mathbf{W}\mathbf{x} + \mathbf{b}) \;:\; \mathbf{W} \in \mathbb{R}^{n\times n}, \mathbf{b} \in \mathbb{R}^n \,\},$$

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

We observe that if a family p is Φ_1-invariant and Φ_2-invariant for two functions Φ_1 and Φ_2 such that the composition $\Phi_2 \circ \Phi_1$ is well-defined, then it is also $(\Phi_2 \circ \Phi_1)$-invariant. In particular, a ReLU-invariant family is invariant for $\mathcal{NN}^n_{L,\varrho}$, i.e., for all ReLU networks of any depths L.

5.1.1 The Two Main Characterization Results

We can now state the first main theorem of this chapter.

Theorem 5.5. *Let $\Omega \subseteq \mathbb{R}^d$ and $p\colon \Omega \to \mathcal{D}(\mathbb{R}^n)$ be a continuous and ReLU-invariant d-parameter family of probability distributions. Then at least one of the following restrictions has to hold:*

R1. Restricted Dimension: $n = 1$,

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 characterize 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 parametrization the only ReLU-invariant distributions in dimensions $n \geq 2$ 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 assumption on the parametrization. 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 parametrization 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 5.5 by providing a kind of reverse statement showing that the three restrictions are individually necessary.

Theorem 5.6. *The following three statements hold independently of each other. There exists a continuous ReLU-invariant d-parameter family of probability distributions that...*

(i) ...fulfills restriction (R1) but avoids restrictions (R2) and (R3) from Theorem 5.5.

(ii) ...fulfills restriction (R2) but avoids restrictions (R1) and (R3) from Theorem 5.5.

(iii) ...fulfills restriction (R3) but avoids restrictions (R1) and (R2) from Theorem 5.5.

The proof for Theorem 5.5 will be given in Section 5.2 and the proof for the reverse result Theorem 5.6 in Section 5.3.

5.2 Proof of the First Characterization Result

The main idea for proving Theorem 5.5 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 parametrized 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 visualized in Figure 5.1. 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 5.5. Firstly, restricting the allowed neural network layers so that transformations to the arcs are

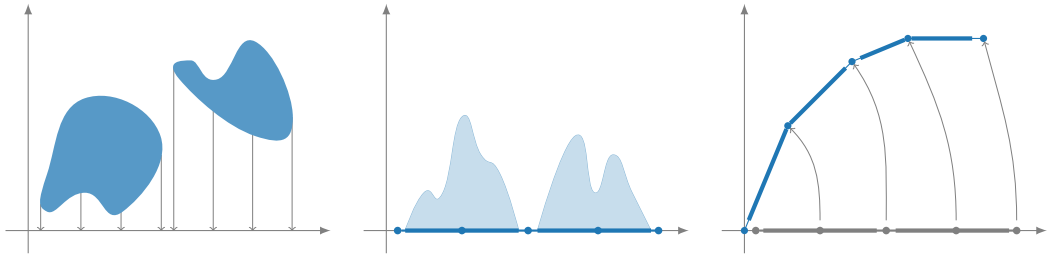


Figure 5.1: **Transformation of Distributions.** The main steps of transforming a generic probability distribution to a distribution supported on a polygonal chain. The original distribution (left) is projected onto a single dimension and partitioned into intervals (center) that each contain a sufficiently 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.

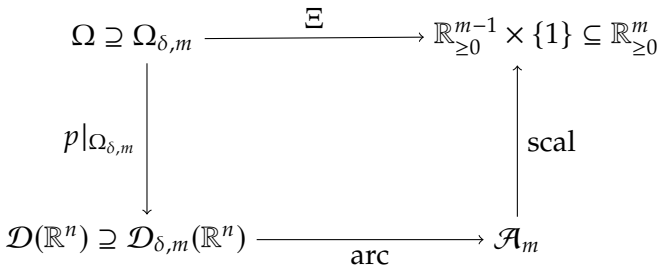


Figure 5.2: **Distribution and Arc Spaces.** Schematic overview of the spaces and functions involved in our proof.

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 parametrization 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 parametrization domain to distributions supported on polygonal arcs, then to the respective arcs, and finally to segment lengths is schematically shown in Figure 5.2. We will give exact definitions of all involved spaces and mappings below.

5.2.1 Details of the Proof

We will now present the main steps of the proof of Theorem 5.5. The proofs of the individual steps will be deferred in order to not distract from the main idea. We will assume toward a contradiction that all three restrictions in Theorem 5.5 are not satisfied.

Assumption 5.7. Let $n \geq 2$, $\Omega \subseteq \mathbb{R}^d$, and $p\colon \Omega \to \mathcal{D}(\mathbb{R}^n)$ be a locally Lipschitz continuous and ReLU-invariant d-parameter family of probability distributions such that there exists a $\theta \in \Omega$ for which $\operatorname{supp}(p(\theta))$ 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,

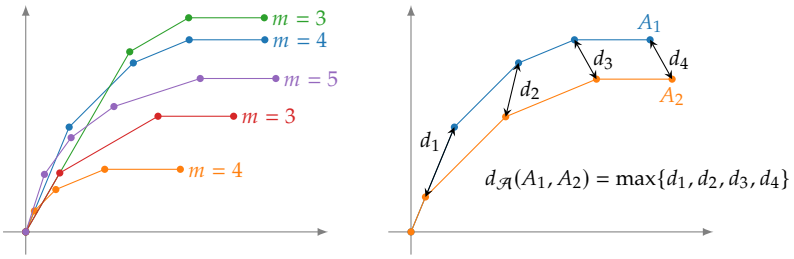


Figure 5.3: **Standard Arcs.** Examples of several standard m-arcs for varying m (left). Standardized arcs are always contained within the non-negative orthant and end on a horizontal line segment. The distance between two m-arcs is the maximal Euclidean distance of corresponding vertices (right).

which we call arcs.

Definition 5.8. An *m-arc* is a subset $A \subseteq \mathbb{R}^2$ defined as a polygonal chain, i.e., a connected piecewise linear curve,

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

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

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

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

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

It will turn out to be useful to remove some ambiguity from the set of m-arcs by standardizing their behavior at the start and end vertices. Some examples can be seen in Figure 5.3.

Definition 5.9. An m-arc A with vertices $\{\mathbf{v}_0, \dots, \mathbf{v}_m\}$ and length scales $\{r_1, \dots, r_m\}$ is called *standardized* (or a *standard m-arc*), if it starts at the origin and has a normalized last line segment, i.e., if $\mathbf{v}_0 = \mathbf{0}_2$ and $r_m = 1$. The set of all standard m-arcs is denoted $\mathcal{A}_m$ and equipped with the metric

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

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 5.10. 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 $n > 2$, a probability measure $\mu \in \mathcal{D}(\mathbb{R}^n)$ 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.

Remark 5.11. There is no particular significance to choosing $\operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\}$ in the definition of arc-supported measures. This is done for ease of notation, but any other two-dimensional subspace would work as well.

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

Lemma 5.12. *Let $\mu \in \mathcal{D}_{\delta,m}(\mathbb{R}^n)$ be δ-distributed on a standard m-arc, then this arc is unique.*

Proof. Toward a contradiction assume that μ is δ-distributed on $A_1, A_2 \in \mathcal{A}_m$ and $A_1 \neq A_2$. Let $\{\ell_1^j, \dots, \ell_m^j\} = \operatorname{segm}(A_j)$ and $\{r_1^j, \dots, r_m^j\} = \operatorname{scal}(A_j)$ 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

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

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

This allows us to define the function that maps an arc-supported measure $\mu \in \mathcal{D}_{\delta,m}(\mathbb{R}^n)$ to its unique induced m-arc $A_\mu \in \mathcal{A}_m$. We denote this as $\operatorname{arc}\colon \mathcal{D}_{\delta,m}(\mathbb{R}^n) \to \mathcal{A}_m \colon \mu \to A_\mu$. For a d-parameter family $p\colon \Omega \to \mathcal{D}(\mathbb{R}^n)$ we denote by

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

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

We are now ready to start discussing the main steps of the proof of Theorem 5.5. It relies on the following three results.

Lemma 5.13. *Under Assumption 5.7 and for any $m \in \mathbb{N}$ there exists a $\delta > 0$ such that the map $\Xi\colon \Omega_{\delta,m} \subseteq \mathbb{R}^d \to \mathbb{R}_+^{m-1} \times \{1\}$ 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 Section 5.2.2. 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. This can be done in four constructive steps:

(i) projecting the measure onto a one-dimensional subspace,

(ii) partitioning its support along that line into segments,

(iii) appropriately scaling the line segments,

(iv) bending the line into arc shape by repeated rotations and projections onto the non-negative orthant.

Finally, using the scaling factors to identify arcs with $\mathbb{R}_+^{m-1} \times \{1\}$ yields the claim.

Lemma 5.14. *Under Assumption 5.7 and for any $m \in \mathbb{N}$ let $\delta > 0$, $\Omega_{\delta,m} \subseteq \mathbb{R}^d$, and $\Xi\colon \Omega_{\delta,m} \subseteq \mathbb{R}^d \to \mathbb{R}_+^{m-1} \times \{1\}$ be as in Lemma 5.13. Then Ξ is locally Lipschitz continuous.*

The proof for this is given in Section 5.2.3. 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 5.15. *Let $\Omega \subseteq \mathbb{R}^d$ and $\Xi\colon \Omega \to \mathbb{R}^m$ be locally Lipschitz continuous. If the image $\Xi(\Omega)$ is a Borel set with non-empty interior in $\mathbb{R}^m$, then $m \leq d$. In particular, if Ξ maps surjectively onto $\mathbb{R}^m$ or $\mathbb{R}_+^m$, then $m \leq d$.*

The full proof of this is given in Appendix C.1. It essentially uses the fact that locally Lipschitz continuous maps cannot increase the Hausdorff dimension. Since the Hausdorff dimension of Ω is at most d and the Hausdorff dimension of $\Xi(\Omega)$ is m, the claim follows.

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

Proof of Theorem 5.5. Toward a contradiction let Assumption 5.7 hold. Let $m > d + 1$ be arbitrary. By Lemma 5.13 there exists a $\delta > 0$ so that the domain $\Omega_{\delta,m}$ is non-empty and $\Xi\colon \Omega_{\delta,m} \to \mathbb{R}_+^{m-1} \times \{1\}$ is surjective. By Lemma 5.14 Ξ is also locally Lipschitz continuous. Since $\mathbb{R}_+^{m-1} \times \{1\}$ is isometric to $\mathbb{R}_+^{m-1}$ and $m - 1 > d$ this contradicts Lemma 5.15. Hence, one of the assumptions in Assumption 5.7 cannot hold, which proves Theorem 5.5. □

5.2.2 Surjectivity of Ξ (Proof of Lemma 5.13)

For the remainder of the section let Assumption 5.7 hold. We will prove Lemma 5.13 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 d-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 neighborhood 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 5.16. *For any $q, r \in \mathbb{N}$ let $\mu \in \mathcal{D}(\mathbb{R}^q)$ and let $\Phi\colon \mathbb{R}^q \to \mathbb{R}^r$ be continuous. Then*

$$\operatorname{supp}(\Phi_*(\mu)) = \overline{\Phi(\operatorname{supp}(\mu))},$$

where $\Phi_*(\mu) \in \mathcal{D}(\mathbb{R}^r)$ is the pushforward of μ under Φ.

Proof. We begin by showing $\Phi(\operatorname{supp}(\mu)) \subseteq \operatorname{supp}(\Phi_*(\mu))$. Let $\mathbf{t} \in \Phi(\operatorname{supp}(\mu))$ and $U \subseteq \mathbb{R}^r$ be any open neighborhood of $\mathbf{t}$. There exists $\mathbf{x} \in \operatorname{supp}(\mu)$ so that $\mathbf{t} = \Phi(\mathbf{x})$. Since $\mathbf{t} \in U$, we get $\mathbf{x} \in \Phi^{-1}(U)$. Thus $\Phi^{-1}(U)$ is an open neighborhood of $\mathbf{x}$ and since $\mathbf{x} \in \operatorname{supp}(\Phi)$ we get $\Phi_*(\mu)(U) = \mu(\Phi^{-1}(U)) > 0$. Since U was an arbitrary neighborhood of $\mathbf{t}$ this shows $\mathbf{t} \in \operatorname{supp}(\Phi_*(\mu))$. This implies $\Phi(\operatorname{supp}(\mu)) \subseteq \operatorname{supp}(\Phi_*(\mu))$. But $\operatorname{supp}(\Phi_*(\mu))$ is closed, so we can even conclude $\overline{\Phi(\operatorname{supp}(\mu))} \subseteq \operatorname{supp}(\Phi_*(\mu))$.

We will show the converse inclusion by showing $\overline{\Phi(\operatorname{supp}(\mu))}^c \subseteq \operatorname{supp}(\Phi_*(\mu))^c$. For this let $\mathbf{t} \in \overline{\Phi(\operatorname{supp}(\mu))}^c$. Then there exists an open neighborhood U of $\mathbf{t}$ such that $U \cap \overline{\Phi(\operatorname{supp}(\mu))} = \emptyset$, which implies $\Phi^{-1}(U) \cap \operatorname{supp}(\mu) = \emptyset$. But this is only possible if $\Phi_*(\mu)(U) = \mu(\Phi^{-1}(U)) = 0$ and therefore $\mathbf{t} \in \operatorname{supp}(\Phi_*(\mu))^c$. □

To simplify notations in the following, we begin the transformation by establishing a standardized situation that takes places exclusively in the two-dimensional subspace $\operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\}$ of $\mathbb{R}^n$. 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 $\operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\}$.

Projection to a One-Dimensional Subspace

We use a projection to a one-dimensional space that preserves the infinite support.

Lemma 5.17. *Let $\mu \in \mathcal{D}(\mathbb{R}^n)$ such that $\operatorname{supp}(\mu)$ is not finite. Then there exists $\Phi \in \mathcal{NN}^n_{1,\varrho}$ such that $\Phi(\mathbb{R}^n) \subseteq \operatorname{span}\{\mathbf{e}_1\}$ and $\operatorname{supp}(\Phi_*(\mu))$ is infinite.*

Proof. The map $\Phi\colon \mathbb{R}^n \to \mathbb{R}^n$ has the form $\mathbf{x} \mapsto \varrho(\mathbf{W}\mathbf{x} + \mathbf{b})$ for some $\mathbf{W} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. Consider the coordinate projections $\operatorname{proj}_i\colon \mathbb{R}^n \to \mathbb{R}\colon \mathbf{x} \mapsto \mathbf{e}_i^\top \mathbf{x}$. Clearly we have

$$\operatorname{supp}(\mu) \subseteq \operatorname{proj}_1(\operatorname{supp}(\mu)) \times \cdots \times \operatorname{proj}_n(\operatorname{supp}(\mu)).$$

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 $\operatorname{proj}_j(\operatorname{supp}(\mu)) \cap \mathbb{R}_{\geq 0}$ is infinite, we set $\sigma = +1$, otherwise $\operatorname{proj}_j(\operatorname{supp}(\mu)) \cap \mathbb{R}_{\leq 0}$ must be infinite and we set $\sigma = -1$. Now choosing

$$\mathbf{W} = \begin{bmatrix} \sigma\mathbf{e}_j & \mathbf{0}_n & \dots & \mathbf{0}_n \end{bmatrix}^\top \quad \text{and} \quad \mathbf{b} = \mathbf{0}_n,$$

clearly yields $\Phi(\mathbb{R}^n) \subseteq \operatorname{span}\{\mathbf{e}_1\}$. Further, by Lemma 5.16 we have

$$\begin{aligned}\operatorname{supp}(\Phi_*(\mu)) &= \overline{\Phi(\operatorname{supp}(\mu))} \\ &= \overline{\varrho(\sigma \operatorname{proj}_j(\operatorname{supp}(\mu)))} \times \{0\} \times \cdots \times \{0\} \\ &= \begin{cases} \overline{\operatorname{proj}_j(\operatorname{supp}(\mu)) \cap \mathbb{R}_{\geq 0}} \times \{0\} \times \cdots \times \{0\}, & \text{if } \sigma = +1 \\ \overline{-\operatorname{proj}_j(\operatorname{supp}(\mu)) \cap \mathbb{R}_{\geq 0}} \times \{0\} \times \cdots \times \{0\}, & \text{if } \sigma = -1 \end{cases},\end{aligned}$$

which by the choice of σ is infinite. □

Partitioning into Line Segments

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.

Lemma 5.18. *Let $\mu \in \mathcal{D}(\mathbb{R}^n)$ such that $\operatorname{supp}(\mu)$ is not finite. Then for any $m \in \mathbb{N}$ there exist points $0 = b_0 < \cdots < b_m < \infty$ and $\Phi \in \mathcal{NN}^n_{3,\varrho}$ such that $\Phi(\mathbb{R}^n) \subseteq \operatorname{span}\{\mathbf{e}_1\} \cap [0, b_m]^n$ and $\operatorname{supp}(\Phi_*(\mu)) \cap \big((b_{j-1}, b_j) \times \{0\} \times \cdots \times \{0\}\big) \neq \emptyset$.*

Proof. By Lemma 5.17 there is $\Phi_1 \in \mathcal{NN}^n_{1,\varrho}$ such that $\Phi(\mathbb{R}^n) \subseteq \operatorname{span}\{\mathbf{e}_1\}$ and $\operatorname{supp}(\Phi_*(\mu))$ is infinite. Hence, there exist $0 < s_1 < \cdots < s_m < \infty$ such that $(s_j, 0, \ldots, 0) \in \operatorname{supp}(\Phi_*(\mu))$ for all $j = 1, \ldots, m$. Setting $b_0 = 0$, and

$$b_j = \frac{s_j + s_{j+1}}{2} \quad \text{for} \quad j = 1, \ldots, m-1,$$

and finally $b_m = s_m + 1$ we get $s_j \in (b_{j-1}, b_j)$ for $j = 1, \ldots, m$. Now choosing $\Phi_2 \in \mathcal{NN}^n_{2,\varrho}$ as

$$\Phi_2 \colon \mathbb{R}^n \to \mathbb{R}^n \colon \mathbf{x} \mapsto \varrho\left(-\varrho\left(-\mathbf{x} + \begin{bmatrix} b_m \\ \mathbf{0}_{n-1} \end{bmatrix}\right) + \begin{bmatrix} b_m \\ \mathbf{0}_{n-1} \end{bmatrix}\right)$$

and $\Phi = \Phi_2 \circ \Phi_1 \in \mathcal{NN}^n_{3,\varrho}$ yields the claim. □

Remark 5.19. The support of the transformed measures is one-dimensional after the initial projection and partitioning steps. Further, 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 an n-dimensional ReLU layer

$$\mathbb{R}^n \to \mathbb{R}^n \colon \mathbf{x} \mapsto \varrho\left(\begin{bmatrix} \mathbf{W} & \mathbf{0}_{2\times(n-2)} \\ \mathbf{0}_{(n-2)\times 2} & \mathbf{0}_{(n-2)\times(n-2)} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{b} \\ \mathbf{0}_{(n-2)\times 1} \end{bmatrix}\right),$$

only operating on $\operatorname{span}\{\mathbf{e}_1, \mathbf{e}_2\} \subseteq \mathbb{R}^n$ by appropriately padding with zeros. To simplify the notation, we leave out these zero paddings and just assume $n = 2$. Altogether, without loss of generality we can from now on assume (for b_j chosen as in Lemma 5.18) 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 $j = 1, \ldots, m$.

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 5.20. *For $m \in \mathbb{N}$, $m \geq 2$ let*

$$0 = a_0 < a_1 < \cdots < a_m < \infty \quad \text{and} \quad 0 = b_0 < b_1 < \cdots < b_m < \infty$$

be two collections of points in $\mathbb{R}_{\geq 0}$. Then there exists a ReLU network $F \in \mathcal{NN}^2_{m+1,\varrho}$ that 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 $\mathbb{R}_{\geq 0} \times \{0\}$ is a piecewise linear map with breakpoints (possibly) at $b_0, \dots, b_m$.

Proof. We denote the layers of F as $F_j \in \mathcal{NN}^2_{1,\varrho}$, that is $F = F_m \circ \cdots \circ F_0$. The proof consists of two parts. First, we iteratively construct a collection of piecewise linear and strictly increasing functions $\{f_j \colon \mathbb{R} \to \mathbb{R}\}_{j=0,\dots,m}$ such that

$$(f_j \circ \cdots \circ f_0)(b_i) = a_i \quad \text{for all} \quad i = 0, \dots, j, \tag{5.3}$$

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 (5.3) is satisfied for $j = 0$. Now assuming we have already constructed strictly increasing and piecewise linear functions $f_0, \dots, f_j$ satisfying (5.3) 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

$$f_{j+1}(x) = \begin{cases} x, & x \leq a_j, \\ x + \frac{a_{j+1} - b_{j+1}^j}{b_{j+1}^j - a_j}(x - a_j), & x > a_j. \end{cases}$$

We can also express f_{j+1} using ϱ, i.e.,

$$f_{j+1}(x) = x + \frac{a_{j+1} - b_{j+1}^j}{b_{j+1}^j - a_j} \varrho(x - a_j). \tag{5.4}$$

Then f_{j+1} maps b^j_{j+1} 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 Figure 5.4(left). Hence clearly (5.3) 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

$$1 + \frac{a_{j+1} - b^j_{j+1}}{b^j_{j+1} - a_j} = \frac{a_{j+1} - a_j}{b^j_{j+1} - a_j} > 0$$

on the region $x \geq a_j$. 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, see Figure 5.4(right).

Next we derive the ReLU layers $\{F_j\}_j$ from the functions $\{f_j\}_j$. For $j = 0, \ldots, m-1$ we denote

$$w_{j+1} = \frac{a_{j+1} - b^j_{j+1}}{b^j_{j+1} - a_j}$$

and rewrite (5.4) as

$$f_{j+1}(x) = \begin{bmatrix} 1 & w_{j+1} \end{bmatrix} \varrho \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ -a_j \end{bmatrix} \right) \quad \text{for} \quad x \geq 0.$$

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 $x \geq 0$, 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 $j = 1, \ldots, m-1$. 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

$$F_m(\mathbf{x}) \mapsto \varrho \left(\begin{bmatrix} 1 & w_m \\ 0 & 0 \end{bmatrix} \mathbf{x} \right)$$

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 $F = F_m \circ F_{m-1} \circ \cdots \circ F_1 \circ F_0$. Clearly all $F_j \in \mathcal{NN}^2_{1,\varrho}$, hence $F \in \mathcal{NN}^2_{m+1,\varrho}$. □

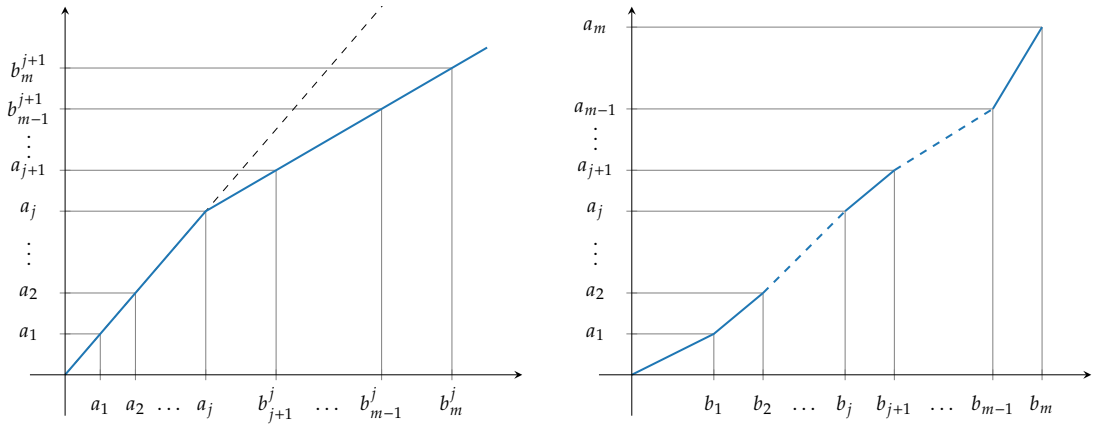


Figure 5.4: **Construction of Arcs – Scaling.** Illustration of the construction in the proof of Lemma 5.20. The function $f_{j+1}(x)$ transforms the points $0 = a_0 = b_0^j, a_1 = b_1^j, \dots, a_j = b_j^j$ and $b_{j+1}^j, \dots, b_m^j$ (left). The region $x \leq a_j$ is left unchanged and the region $x \geq a_j$ 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).

Bending the Arc

Let us next consider how to "bend" a one dimensional subspace into an m-arc. This iterative procedure is illustrated in Figure 5.5. It starts with all vertices in a straight line on the x_2-axis. First, the polygonal chain is rotated by $-\alpha_m$ ("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 activation 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 $\cos(\alpha_m)$.

Lemma 5.21. *For $m \in \mathbb{N}$, $m \geq 2$ let $0 = a_0 < a_1 < \cdots < a_m < \infty$ be a collection of points in $\mathbb{R}_{\geq 0}$. Then there exists a ReLU network $G \in \mathcal{NN}^2_{m+1,\varrho}$ such that its restriction $G|_{[a_0,a_m]\times\{0\}}$ parametrizes the unique m-arc A starting in the origin with scaling factors $r_i = \cos^{m-i}(\alpha_m)(a_i - a_{i-1})$ for $i = 1, \dots, m$.*

Proof. Let the rotation matrix of angle α be denoted

$$R_\alpha = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

and define

$$\mathbf{d}_j = \begin{bmatrix} \sin(\alpha_m)\cos(\alpha_m)^j a_{m-j} \\ 0 \end{bmatrix}, \quad \text{for} \quad j = 1, \dots, m,$$

as well as

$$G_0 \colon \mathbb{R}^2 \to \mathbb{R}^2 \colon \mathbf{x} \mapsto \varrho\left(R_{\frac{\pi}{2}}\mathbf{x}\right)$$

and

$$G_j\colon \mathbb{R}^2 \to \mathbb{R}^2\colon \mathbf{x} \mapsto \varrho\left(R_{-\alpha_m}\mathbf{x} - \mathbf{d}_j\right) \quad \text{for} \quad j = 1, \dots, m,$$

and finally $G = G_m \circ \cdots \circ G_0$. Clearly all $G_j \in \mathcal{NN}^2_{1,\varrho}$, hence $G \in \mathcal{NN}^2_{m+1,\varrho}$.

We need to show that

$$\mathbf{v}_i = G\left(\begin{bmatrix} a_i \\ 0 \end{bmatrix}\right) \quad \text{and} \quad \ell_i = G([a_{i-1}, a_i] \times \{0\})$$

are the vertices and line segments of the m-arc A respectively. For this, it remains to check that G is a piecewise linear function in its first component with breakpoints a_i, and that $\mathbf{v}_0 = \mathbf{0}_2$ as well as

$$\mathbf{v}_i = \mathbf{v}_{i-1} + \cos^{m-i}(\alpha_m)(a_i - a_{i-1}) \begin{bmatrix} \sin(i\alpha_m) \\ \cos(i\alpha_m) \end{bmatrix}, \quad \text{for} \quad i = 1, \dots, m. \tag{5.5}$$

We denote

$$\mathbf{v}_i^j = (G_j \circ \cdots \circ G_0)\left(\begin{bmatrix} a_i \\ 0 \end{bmatrix}\right) \quad \text{for} \quad i, j = 0, \dots, m.$$

By the choice of α_m we know $\sin(\alpha_m) > 0$ and $\cos(\alpha_m) > 0$. Hence, the first component of the bias vectors $\mathbf{d}_j$ satisfy $\sin(\alpha_m)\cos(\alpha_m)^j a_{m-j} \geq 0$ for $j = 1, \dots, m$. Therefore, $\mathbf{v}_0^j = \mathbf{0}_2$ for all j and in particular $\mathbf{v}_0 = \mathbf{v}_0^m = \mathbf{0}_2$. Further, we will show inductively over j that for any i, j we have

$$\mathbf{v}_i^j - \mathbf{v}_{i-1}^j = \begin{cases} \cos(\alpha_m)^j (a_i - a_{i-1}) \begin{bmatrix} 0 \\ 1 \end{bmatrix}, & i \leq m - j, \\ \cos(\alpha_m)^{m-i} (a_i - a_{i-1}) \begin{bmatrix} \sin((i+j-m)\alpha_m) \\ \cos((i+j-m)\alpha_m) \end{bmatrix}, & i > m - j. \end{cases} \tag{5.6}$$

Then (5.5) follows from (5.6) 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 (5.6).

Next, let us assume that (5.6) holds for some $j < m$. For any i we denote $\mathbf{w}_i^j = R_{-\alpha_m}\mathbf{v}_i^j - \mathbf{d}_j$ and therefore $\mathbf{v}_i^{j+1} = G_{j+1}(\mathbf{v}_i^j) = \varrho(\mathbf{w}_i^j)$ and

$$\mathbf{w}_i^j - \mathbf{w}_{i-1}^j = \begin{cases} \cos(\alpha_m)^j (a_i - a_{i-1}) \begin{bmatrix} \sin(\alpha_m) \\ \cos(\alpha_m) \end{bmatrix}, & i \leq m - j, \\ \cos(\alpha_m)^{m-i} (a_i - a_{i-1}) \begin{bmatrix} \sin((i+j+1-m)\alpha_m) \\ \cos((i+j+1-m)\alpha_m) \end{bmatrix}, & i > m - j. \end{cases}$$

We also have $\sin((i+j+1-m)\alpha_m) > 0$ and $\cos((i+j+1-m)\alpha_m) \geq 0$ for $i > m-j$ by the choice of α_m and therefore $\mathbf{w}_m^j \geq \mathbf{w}_{m-1}^j \geq \cdots \geq \mathbf{w}_1^j \geq \mathbf{w}_0^j = -\mathbf{d}_j$ component-wise, where even all inequalities are strict in the first component. We immediately see that all $\mathbf{w}_i^j$ lie in the upper half-plane. Further,

$$\begin{aligned}
\mathbf{w}_{m-j}^j &= \mathbf{w}_0^j + \sum_{i=1}^{m-j} (\mathbf{w}_i^j - \mathbf{w}_{i-1}^j) \\
&= -\mathbf{d}_j + \sum_{i=1}^{m-j} \cos(\alpha_m)^j (a_i - a_{i-1}) \begin{bmatrix} \sin(\alpha_m) \\ \cos(\alpha_m) \end{bmatrix} \\
&= -\begin{bmatrix} \sin(\alpha_m)\cos(\alpha_m)^j a_{m-j} \\ 0 \end{bmatrix} + \cos(\alpha_m)^j a_{m-j} \begin{bmatrix} \sin(\alpha_m) \\ \cos(\alpha_m) \end{bmatrix} \\
&= \cos(\alpha_m)^{j+1} a_{m-j} \begin{bmatrix} 0 \\ 1 \end{bmatrix},
\end{aligned}$$

from which we can conclude that

$$(\mathbf{w}_i^j)_1 \geq 0, \quad \text{for} \quad i = 1, \ldots, m \tag{5.7}$$

$$(\mathbf{w}_i^j)_2 < 0, \quad \text{for} \quad i = 1, \ldots, m-j-1, \tag{5.8}$$

$$(\mathbf{w}_i^j)_2 \geq 0, \quad \text{for} \quad i = m-j, \ldots, m. \tag{5.9}$$

Now $\mathbf{v}_i^{j+1} = \varrho(\mathbf{w}_i^j)$ together with (5.7) to (5.9) implies

$$\begin{aligned}
(\mathbf{v}_i^{j+1})_1 &= (\mathbf{w}_i^j)_1, \quad \text{for} \quad i = 1, \ldots, m, \\
(\mathbf{v}_i^{j+1})_2 &= 0, \quad \text{for} \quad i = 1, \ldots, m-j-1, \\
(\mathbf{v}_i^{j+1})_2 &= (\mathbf{w}_i^j)_2, \quad \text{for} \quad i = m-j, \ldots, m,
\end{aligned}$$

and therefore

$$\mathbf{v}_i^{j+1} - \mathbf{v}_{i-1}^{j+1} = \begin{cases} \cos(\alpha_m)^{j+1}(a_i - a_{i-1}) \begin{bmatrix} 0 \\ 1 \end{bmatrix}, & i \leq m-(j+1) \\ \cos(\alpha_m)^{m-i}(a_i - a_{i-1}) \begin{bmatrix} \sin((i+j+1-m)\alpha_m) \\ \cos((i+j+1-m)\alpha_m) \end{bmatrix}, & i > m-(j+1), \end{cases}$$

which shows that (5.6) holds for $j+1$ and concludes the inductive step.

Each G_{j+1} adds only one new breakpoint to the overall function, corresponding to the point $\mathbf{w}_{m-j}^j$ 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. □

Combining the Pieces

We are now ready to given the main proof of this section.

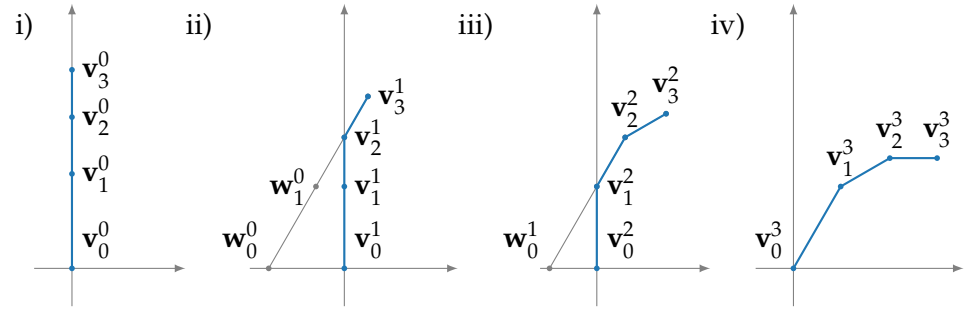


Figure 5.5: **Construction of Arcs – Bending.** Illustration of the "bending" of a 3-arc in 4 steps. Step (i) shows G_0 (90° rotation) applied to $[a_m, a_0] \times \{0\}$. Steps (ii) and (iii) show the image of $G_1 \circ G_0$ and $G_2 \circ G_1 \circ G_0$ respectively (blue) as well as the corresponding transform just before the last ReLU activation (gray). Step (iv) shows the final arc arising as the image of $G = G_3 \circ G_2 \circ G_1 \circ G_0$.

Proof of Lemma 5.13. Let $m \in \mathbb{N}$ be arbitrary. By assumption there exists a measure $\mu \in \{p(\theta)\}_{\theta\in\Omega}$ with infinite support. By using Lemmas 5.17 and 5.18 and Remark 5.19 we can without loss of generality assume $n = 2$ and that $\operatorname{supp}(\mu)$ is compact and contained in the non-negative part of $\operatorname{span}\{\mathbf{e}_1\}$. More precisely, $\operatorname{supp}(\mu) = S \times \{0\} \subseteq [0, b] \times \{0\}$ for a set $S \subseteq \mathbb{R}_{\geq 0}$ and some $b > 0$ and there are 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 $a_j = a_{j-1} + r_j \cos^{j-m}(\alpha_m)$ for $j = 1, \ldots, m$. By Lemma 5.20 there is a ReLU network $F \in \mathcal{NN}^n_{m+1,\varrho}$ 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 5.21 another ReLU network $G \in \mathcal{NN}^n_{m+1,\varrho}$ satisfying

$$G\left(\begin{bmatrix} a_i \\ 0 \end{bmatrix}\right) = \mathbf{v}_j \quad \text{for} \quad j = 1, \ldots, m.$$

Altogether $\Phi = G \circ F \in \mathcal{NN}^n_{2m+2,\varrho}$ is a ReLU network that transforms $\mathbf{b}_j = \begin{bmatrix} b_i & 0 \end{bmatrix}^\top$ to the vertex $\mathbf{v}_j$ and $[b_{j-1}, b_j] \times \{0\}$ to the line segment ℓ_j. By Lemma 5.16 we have

$$\operatorname{supp}(\Phi_*\mu) = \overline{\Phi(\operatorname{supp}(\mu))} \subseteq \overline{\Phi([b_0, b_m] \times \{0\})} \subseteq A.$$

Also, for each j we get

$$(\Phi_*\mu)(\ell_j) = \mu(\Phi^{-1}(\ell_j)) \geq \mu((b_{j-1}, b_j) \times \{0\}) \geq \delta$$

and therefore $\Phi_*\mu \in \mathcal{D}_{\delta,m}(\mathbb{R}^2)$ and $\operatorname{arc}(\Phi_*\mu) = A$. Using the ReLU-invariance, we know that $\Phi_*\mu \in \{p(\theta)\}_{\theta\in\Omega_{\delta,m}} \subseteq \{p(\theta)\}_{\theta\in\Omega}$. 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. □

5.2.3 Local Lipschitz Continuity of Ξ (Proof of Lemma 5.14)

We will now show the local Lipschitz continuity of the map $\Xi\colon \Omega_{\delta,m} \to \mathbb{R}_+^m$ by showing it for each of its three composite parts $p|_{\Omega_{\delta,m}}\colon \Omega_{\delta,m} \to \mathcal{D}_{\delta,m}(\mathbb{R}^n)$, $\mathrm{arc}\colon \mathcal{D}_{\delta,m}(\mathbb{R}^n) \to \mathcal{A}_m$, and $\mathrm{scal}\colon \mathcal{A}_m \to \mathbb{R}_+^m$.

Firstly, $p|_{\Omega_{\delta,m}}\colon \Omega_{\delta,m} \to \mathcal{D}_{\delta,m}(\mathbb{R}^n)$ is locally Lipschitz continuous by Assumption 5.7, so there is nothing to show.

Secondly, the local Lipschitz continuity of $\mathrm{arc}\colon \mathcal{D}_{\delta,m}(\mathbb{R}^n) \to \mathcal{A}_m$ can be derived from geometric observations.

Lemma 5.22. *The map* $\mathrm{arc}\colon (\mathcal{D}_{\delta,m}(\mathbb{R}^n), d_P) \to (\mathcal{A}_m, d_{\mathcal{A}})\colon \mu \mapsto A_\mu$, *is locally Lipschitz continuous with Lipschitz constant* $\frac{2\sqrt{2}}{\sin(\alpha_m)}$.

Proof. Let $\mu_0 \in \mathcal{D}_{\delta,m}(\mathbb{R}^n)$ be arbitrary. We will show Lipschitz continuity on the open ball of radius $\frac{\delta}{2}$ around μ_0. For this let $\mu_1, \mu_2 \in B_{\frac{\delta}{2}}(\mu_0) \cap \mathcal{D}_{\delta,m}(\mathbb{R}^n)$ and therefore $d_P(\mu_1, \mu_2) \leq \delta$. We know that μ_1 and μ_2 are δ-supported on unique standard m-arcs $A_1 = \mathrm{arc}(\mu_1) = A_{\mu_1} \in \mathcal{A}_m$ and $A_2 = \mathrm{arc}(\mu_2) = A_{\mu_2} \in \mathcal{A}_m$ and by definition these lie in the same two-dimensional subspace $\mathrm{span}\{\mathbf{e}_1, \mathbf{e}_2\}$ of $\mathbb{R}^n$. So without loss of generality we carry out the remaining proof in $\mathbb{R}^2$. Let $(\mathbf{v}_0^1, \ldots, \mathbf{v}_m^1) = \mathrm{vert}(A_1)$ and $(\mathbf{v}_0^2, \ldots, \mathbf{v}_m^2) = \mathrm{vert}(A_2)$ be the vertices and $(\ell_1^1, \ldots, \ell_m^1) = \mathrm{segm}(A_1)$ and $(\ell_1^2, \ldots, \ell_m^2) = \mathrm{segm}(A_2)$ the line segments of A_1 and A_2 respectively. We denote the affine hulls of individual line segments as $h_i^j = \mathrm{aff}(\ell_i^j)$ and further denote the Euclidean distance of two corresponding affine hulls by $d_i = \mathrm{dist}(h_i^1, h_i^2)$. An example for this can be seen in Figure 5.6.

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 visualized in Figure 5.7. Firstly, if $\mathbf{v}_i^1 = \mathbf{v}_i^2$ (cf. Figure 5.7, 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 5.7, top right) we obtain

$$\|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2 \leq \frac{d_{i+1}}{\sin(\alpha_m)}.$$

Thirdly, if $\mathbf{v}_i^1 \neq \mathbf{v}_i^2$ but $h_{i+1}^1 = h_{i+1}^2$ (cf. Figure 5.7, bottom left) we similarly get

$$\|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2 \leq \frac{d_i}{\sin(\alpha_m)}.$$

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 5.7, bottom right). We obtain

$$\|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2 \leq \frac{\sqrt{2}(d_i + d_{i+1})}{\sin(\alpha_m)},$$

using the parallelogram identity. Thus, it remains to bound all the distances d_i.

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 $\mathrm{dist}(\mathbf{0}_2, h_i^1) > \mathrm{dist}(\mathbf{0}_2, h_i^2)$ and vice versa if $\mathrm{dist}(\mathbf{0}_2, h_i^1) < \mathrm{dist}(\mathbf{0}_2, h_i^2)$. Without loss of generality assume that ℓ_i^1 is the outer line

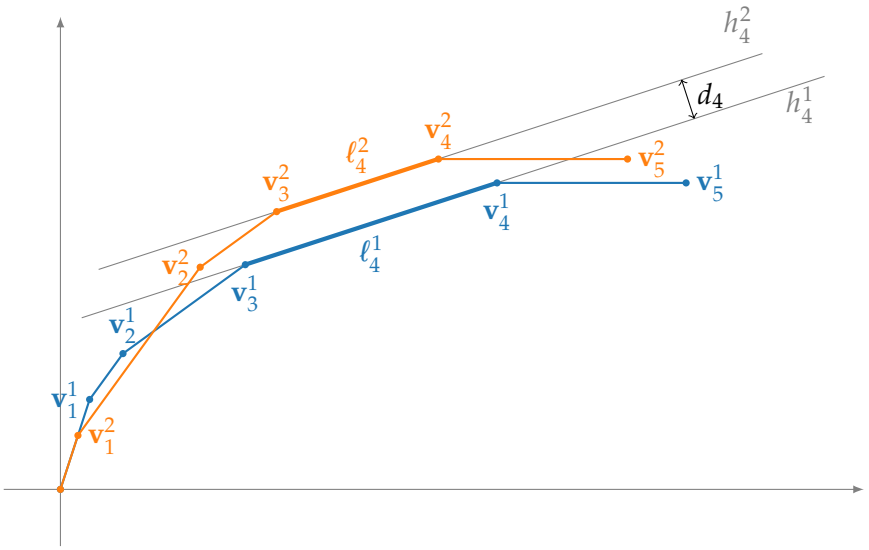


Figure 5.6: **Outer and Inner Arcs.** Example of two standard m-arcs A_1 (blue) and A_2 (orange) for $m = 5$. Supporting affine subspaces h_4^1 and h_4^2 and their distance d_4 are shown exemplarily for the fourth line segments. At this segment A_2 is the outer arc and A_1 is the inner arc.

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

$$d_{\mathcal{A}}(A_1, A_2) = \max_i \|\mathbf{v}_i^1 - \mathbf{v}_i^2\|_2 \leq \frac{2\sqrt{2}}{\sin(\alpha_m)} d_P(\mu_1, \mu_2). \qquad \square$$

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 5.23. *The map* $\operatorname{scal}\colon (\mathcal{A}_m, d_{\mathcal{A}}) \to (\mathbb{R}_{\geq 0}^m, \|\cdot\|_\infty)\colon A \mapsto (r_1, \ldots, r_m)$ *is Lipschitz continuous with Lipschitz constant* 2.

Proof. Let $A_1, A_2 \in \mathcal{A}_m$ be arbitrary standard m-arcs with vertices $(\mathbf{v}_0^1, \ldots, \mathbf{v}_m^1) = \operatorname{vert}(A_1)$ and $(\mathbf{v}_0^2, \ldots, \mathbf{v}_m^2) = \operatorname{vert}(A_2)$ and scaling factors $(r_1^1, \ldots, r_m^1) = \operatorname{scal}(A_1)$ and $(r_1^2, \ldots, r_m^2) = \operatorname{scal}(A_2)$ respectively. Since for all $i = 1, \ldots, m$ we have

$$r_i^1 \leq r_i^2 + \operatorname{dist}(\mathbf{v}_{i-1}^1, \mathbf{v}_{i-1}^2) + \operatorname{dist}(\mathbf{v}_i^1, \mathbf{v}_i^2)$$

and vice versa

$$r_i^2 \leq r_i^1 + \operatorname{dist}(\mathbf{v}_{i-1}^1, \mathbf{v}_{i-1}^2) + \operatorname{dist}(\mathbf{v}_i^1, \mathbf{v}_i^2),$$

we obtain

$$\|\operatorname{scal}(A_1) - \operatorname{scal}(A_2)\|_\infty = \max_{i=1,\ldots,m} |r_i^1 - r_i^2| \leq 2 d_{\mathcal{A}}(A_1, A_2). \qquad \square$$

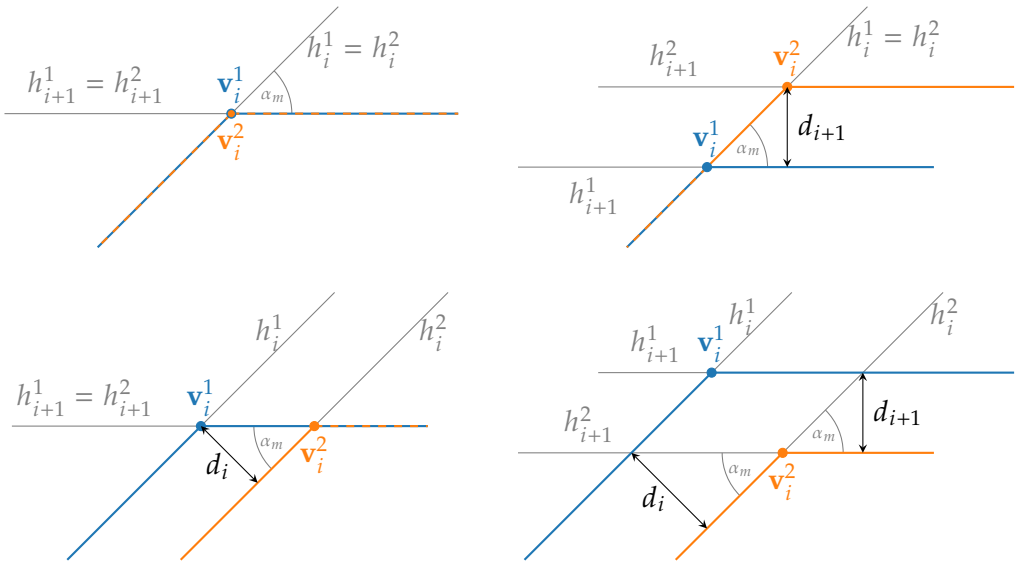


Figure 5.7: **Arc Arrangements.** Four possible arrangements of corresponding vertices $\mathbf{v}^1_i$ and $\mathbf{v}^2_i$ of two arcs. In the first case both vertices are equal (top left). In the second and third case the vertices are different but either the affine subspaces h^1_i and h^2_i or h^1_{i+1} and h^2_{i+1} coincide (top right and bottom left). In the fourth case all affine subspaces differ and their intersections form a parallelogram with $\mathbf{v}^1_i$ and $\mathbf{v}^2_i$ at opposite corners (bottom right).

Altogether we can combine the pieces and prove the local Lipschitz continuity of $\Xi\colon \Omega_{\delta,m} \to \mathbb{R}^m_+$.

Proof of Lemma 5.14. The claim follows directly from Assumption 5.7 and Lemmas 5.22 and 5.23 since the composition of locally Lipschitz continuous functions is again locally Lipschitz continuous. □

5.3 Proof of the Second Characterization Result

We will now come to the proof of Theorem 5.6 and show that exploiting any of the three restrictions in Theorem 5.5 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.

5.3.1 Families of Distributions in One Dimension

The first part of Theorem 5.6 corresponds to restriction (R1). We explicitly construct a family of ReLU-invariant probability distributions on $\mathbb{R}$ with a locally Lipschitz continuous parametrization 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 Figure 5.8, 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 parametrization.

We will follow a similar idea to obtain ReLU-invariant families in higher dimension when proving part (iii) of Theorem 5.6 in Section 5.3.3. 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 generalize our construction to higher dimensions this comes at the cost of losing the local Lipschitz continuity of the parametrization map.

We proceed in three steps. We first analyze the set of one-dimensional ReLU networks and show that it can be completely described by only six parameters. We then use these to parametrize all ReLU neural networks as functions in $C(K, \mathbb{R})$ on some compact domain $K \subseteq \mathbb{R}$ via a so called *realization* 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 $\Phi \in C(K, \mathbb{R})$ to the pushforward of a fixed *prototype* measure $\mu_0 \in \mathcal{D}(\mathbb{R})$ under Φ 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 now discuss each of the steps in more detail.

ReLU Neural Networks in One Dimension

Let us first show that any one dimensional ReLU network can be rewritten as a three layer network. We observe that $\mathcal{NN}^n_{1,\varrho} \subseteq \mathcal{NN}^n_{2,\varrho} \subseteq \cdots \subseteq \mathcal{NN}^n_{L,\varrho} \subseteq \mathcal{NN}^n_{L+1,\varrho} \subseteq \dots$ holds for any $n \in \mathbb{N}$ since "unused" layers can be set to act as the identity map (on $\mathbb{R}^n_{\geq 0}$). For $n \geq 2$ these inclusions are strict, however for $n = 1$ equality holds starting from depth $L = 3$.

Lemma 5.24. *For $L \in \mathbb{N}$, $L \geq 3$ we have $\mathcal{NN}^1_{L,\varrho} = \mathcal{NN}^1_{3,\varrho}$.*

Proof. The inclusion $\mathcal{NN}^1_{L,\varrho} \supseteq \mathcal{NN}^1_{3,\varrho}$ is clear. We will show the reverse inclusion. For this, let $\Phi \in \mathcal{NN}^1_{L,\varrho}$ and for $i = 1, \dots, L$ denote by $w_i, b_i \in \mathbb{R}$ the weight and bias of the i-th layer $\Phi_i(x) = \varrho(w_i x + b_i)$. If $w_i = 0$ for some layer, then Φ is a constant function taking a non-negative value, hence clearly $\Phi \in \mathcal{NN}^1_{3,\varrho}$. So from now on assume $w_i \neq 0$ for all layers. Denote by

$$C_i = \{\, x \in \mathbb{R} : w_i x + b_i \geq 0 \,\} \quad \text{for} \quad i = 1, \dots, L$$

the domain on which the i-th layer is affine linear with slope w_i. It is constant on the complement C_i^c. Further, let

$$D_i = (\Phi_{i-1} \circ \cdots \circ \Phi_1)^{-1}(C_i) \quad \text{for} \quad i = 2, \dots, L$$

and $D_1 = C_1$. Each C_i is a convex and closed subset of $\mathbb{R}$ and $\Phi_{i-1} \circ \cdots \circ \Phi_1$ is continuous and monotone as a composition of continuous and monotone functions. Hence, also all D_i are closed and convex. In $\mathbb{R}$ these are exactly the (possibly unbounded) closed intervals or the empty set.

Let now $U \subseteq \mathbb{R} \setminus \bigcap_{i=1}^L D_i$ be an arbitrary connected set. We will show that $\Phi|_U$ is constant. To see this, observe that $U = \bigcup_{i=1}^L U \cap D_i^c$. The complement D_i^c of the closed set D_i is open, hence $U \cap D_i^c$ is relatively open in U. By the choice of D_i we know $(\Phi_i \circ \cdots \circ \Phi_1)(U \cap D_i^c) \equiv 0$,

hence $\Phi|_{U\cap D_i^c}$ is constant. Since Φ is continuous and U is connected this implies that $\Phi|_U$ is constant.

From this we conclude that Φ is piecewise constant except for the (possibly empty) closed and convex region $\bigcap_{i=1}^{L} D_i$ where all layers are non constant and act affine linearly. On this region Φ is affine linear with slope $\prod_{i=1}^{L} w_i$.

Together with the fact that Φ is continuous, monotone and lower bounded by zero, this leaves only few different cases that can occur and that determine the overall shape of the function Φ, see Figure 5.8.

Bounded Case: Φ is bounded and has two constant and one affine linear region:

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

for some $c_1, c_2 \geq 0$ and $a_1 < a_2$. This includes the degenerate case where Φ is constant by choosing $c_1 = c_2$.

Unbounded Case: Φ is unbounded and has one constant and one affine linear region:

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

for some $c \geq 0$, $a \in \mathbb{R}$, and $w \neq 0$.

It remains to find the weights $\widetilde{w}_1, \widetilde{w}_2, \widetilde{w}_3$ and biases $\widetilde{b}_1, \widetilde{b}_2, \widetilde{b}_3$ of a three layer network $\widetilde{\Phi} \in \mathcal{NN}^1_{3,\varrho}$ that represents the same function as Φ. In the first case of a bounded function we choose

$$\begin{aligned} \widetilde{w}_1 &= 1, & \widetilde{b}_1 &= -a_1, \\ \widetilde{w}_2 &= -\frac{|c_2-c_1|}{a_2-a_1}, & \widetilde{b}_2 &= |c_2-c_1|, \\ \widetilde{w}_3 &= -\operatorname{sign}(c_2-c_1), & \widetilde{b}_3 &= c_2, \end{aligned}$$

and in the second case of an unbounded function we choose

$$\begin{aligned} \widetilde{w}_1 &= \operatorname{sign}(w), & \widetilde{b}_1 &= -\operatorname{sign}(w)a, \\ \widetilde{w}_2 &= |w|, & \widetilde{b}_2 &= c, \\ \widetilde{w}_3 &= 1, & \widetilde{b}_3 &= 0. \end{aligned}$$

The equality $\Phi = \widetilde{\Phi}$ follows by straightforward calculations. □

Neural Network Parametrization

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

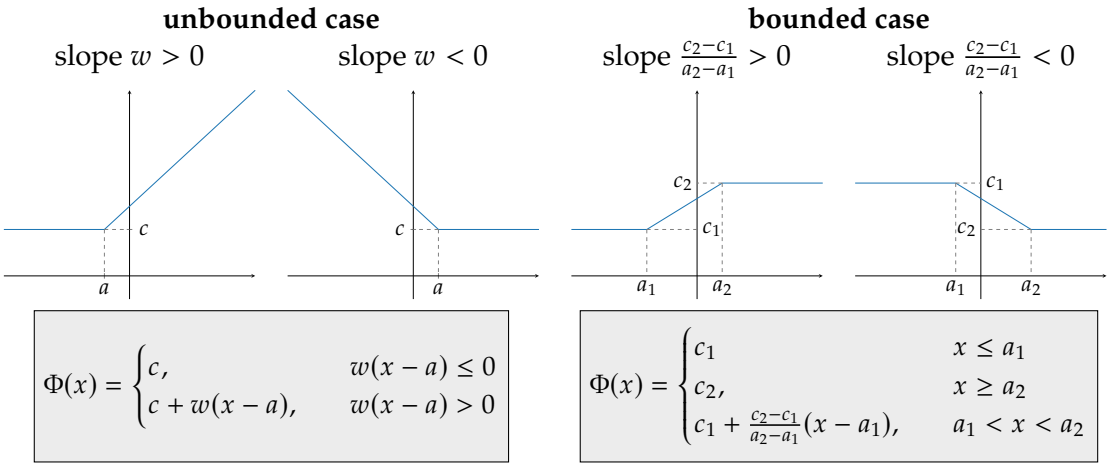


Figure 5.8: **Networks in One Dimension.** ReLU networks in dimension $n = 1$ can only take one of five distinct forms. We show the four non-constant cases. This includes two unbounded cases (left) and two bounded cases (right).

and biases. The mapping from weights and biases to the corresponding neural network function is referred to as the *realization* map. In [PRV18] the continuity of the realization 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 $C(K, \mathbb{R}^n)$ on a compact domain $K \subseteq \mathbb{R}^n$. If the activation function is Lipschitz continuous (which again is the case for the ReLU activation) also the realization 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 parametrized 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 realization of these weights and biases as $\Phi[\theta] \colon K \to \mathbb{R}$. Then the realization map is

$$R \colon \mathbb{R}^6 \to C(K, \mathbb{R}) \colon \theta \mapsto \Phi[\theta].$$

From [PRV18, Proposition 5.1] we obtain the following result.

Lemma 5.25. *The realization map $R \colon \mathbb{R}^6 \to C(K, \mathbb{R})$ 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

$$Q \colon C(K, \mathbb{R}) \to \mathcal{D}(\mathbb{R}) \colon \Phi \mapsto \Phi_* \mu_0.$$

The restriction of Q to the subset of ReLU neural network functions will be used to generate the invariant family of distributions.

Lemma 5.26. *The pushforward function $Q\colon C(K,\mathbb{R}) \to \mathcal{D}(\mathbb{R})$ is Lipschitz continuous.*

Proof. Let $\Phi_1, \Phi_2 \in C(K,\mathbb{R})$ and $B \in \mathcal{B}(\mathbb{R})$. Assuming $\Phi_1^{-1}(B) \neq \emptyset$, for any $x \in \Phi_1^{-1}(B)$ we have $\Phi_1(x) \in B$ and $|\Phi_1(x) - \Phi_2(x)| \leq \|\Phi_1 - \Phi_2\|_\infty$. Consequently, for any $\epsilon > \|\Phi_1 - \Phi_2\|_\infty$ we get $\Phi_2(x) \in B^\epsilon$ and therefore $x \in \Phi_2^{-1}(B^\epsilon)$. Thus $\Phi_1^{-1}(B) \subseteq \Phi_2^{-1}(B^\epsilon)$. Conversely, if $\Phi_1^{-1}(B) = \emptyset$ then the same inclusion trivially holds. Analogously $\Phi_2^{-1}(B) \subseteq \Phi_1^{-1}(B^\epsilon)$ can be shown for any $\epsilon > \|\Phi_1 - \Phi_2\|_\infty$. This directly implies

$$\begin{aligned} d_P(Q(\Phi_1), Q(\Phi_2)) &= d_P((\Phi_1)_*\mu_0, (\Phi_2)_*\mu_0) \\ &= \inf\{\,\epsilon > 0 : (\Phi_1)_*\mu_0(B) \leq (\Phi_2)_*\mu_0(B^\epsilon) + \epsilon \text{ and} \\ &\qquad (\Phi_2)_*\mu_0(B) \leq (\Phi_1)_*\mu_0(B^\epsilon) + \epsilon \text{ for any } B \in \mathcal{B}(\mathbb{R})\,\} \\ &= \inf\{\,\epsilon > 0 : \mu_0(\Phi_1^{-1}(B)) \leq \mu_0(\Phi_2^{-1}(B^\epsilon)) + \epsilon \text{ and} \\ &\qquad \mu_0(\Phi_2^{-1}(B)) \leq \mu_0(\Phi_1^{-1}(B^\epsilon)) + \epsilon \text{ for any } B \in \mathcal{B}(\mathbb{R})\,\} \\ &\leq \|\Phi_1 - \Phi_2\|_\infty. \end{aligned}$$

□

ReLU-Invariance

We can now define the family of distributions as $p = Q \circ R$. The local Lipschitz continuity of p is clear from the previous steps. It remains to establish the ReLU-invariance.

Lemma 5.27. *The six-parameter family of distributions $p\colon \mathbb{R}^6 \to \mathcal{D}(\mathbb{R})\colon \theta \mapsto (Q \circ R)(\theta)$ 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 $\Phi \in \mathcal{NN}^1_{1,\varrho}$ be any ReLU layer. We need to show that there exists a $\omega \in \mathbb{R}^6$ such that $p(\omega) = \Phi_* p(\theta)$.

By construction $p(\theta) = (Q \circ R)(\theta) = (R(\theta))_*\mu_0 = (\Phi[\theta])_*\mu_0$, where $\Phi[\theta] = R(\theta)$ is a ReLU neural network. Clearly also $\Phi \circ \Phi[\theta]$ is a ReLU neural network, and since by Lemma 5.24 any ReLU network can be rewritten as a three-layer network, there exists $\omega \in \mathbb{R}^6$ such that $R(\omega) = \Phi[\omega] = \Phi \circ \Phi[\theta]$. Finally,

$$p(\omega) = (\Phi \circ \Phi[\theta])_*\mu_0 = \Phi_*(\Phi[\theta])_*\mu_0 = \Phi_* p(\theta)$$

yields the claim. □

5.3.2 Families of Distributions With Finite Support

In this section we will prove the second part of Theorem 5.6 and show that ReLU-invariant families of probability distributions exist in which all distributions are finitely supported. A finitely supported Radon probability measure $\mu \in \mathcal{D}(\mathbb{R}^n)$ can be expressed as a mixture of finitely many Dirac measures,

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

for some $N \in \mathbb{N}$, $\mathbf{a}_i \in \mathbb{R}^n$, and $0 \leq c_i \leq 1$ with $\sum_{i=0}^{N} c_i = 1$. For any measurable function $\Phi\colon \mathbb{R}^n \to \mathbb{R}^n$ the pushforward of such a mixture is simply the mixture of the individual pushforwards, that is

$$\Phi_*(\mu) = \sum_{i=1}^{N} c_i \Phi_*(\delta_{\mathbf{a}_i}).$$

The pushforward of a Dirac measure is again a Dirac measure, more precisely we simply have $\Phi_*(\delta_{\mathbf{a}_i}) = \delta_{\Phi(\mathbf{a}_i)}$, and therefore

$$\Phi_*(\mu) = \sum_{i=1}^{N} c_i \delta_{\Phi(\mathbf{a}_i)}.$$

In particular, for a ReLU layer $\Phi \in \mathcal{NN}^n_{1,\varrho}$ with $\Phi(\mathbf{x}) = \varrho(\mathbf{W}\mathbf{x} + \mathbf{b})$ we get

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

Altogether, this shows that for any $N \in \mathbb{N}$ the set

$$\mathcal{D}_N = \{\, \mu \in \mathcal{D}(\mathbb{R}^n) \,:\, |\operatorname{supp}(\mu)| \leq N \,\}$$

is ReLU-invariant in any dimension $n \in \mathbb{N}$. It can be described by $d = N(n+1)$ parameters by simply specifying all the locations $\mathbf{a}_i$ and mixture coefficients c_i. More precisely, let $\Delta_N = \{\, \mathbf{c} \in [0,1]^N \,:\, \sum_i c_i = 1 \,\}$ and

$$\Omega_N = \underbrace{\mathbb{R}^n \times \cdots \times \mathbb{R}^n}_{N \text{ times}} \times \Delta_N \subseteq \mathbb{R}^d$$

and

$$p_N\colon \Omega_N \to \mathcal{D}(\mathbb{R}^n)\colon (\mathbf{a}_1, \ldots, \mathbf{a}_N, \mathbf{c}) \mapsto \sum_{i=1}^{N} c_i \delta_{\mathbf{a}_i}.$$

Then p_N is Lipschitz continuous and parametrizes $\mathcal{D}_N$.

5.3.3 Families of Distributions Without Local Lipschitz Continuity

In this section we will prove the third part of Theorem 5.6. If we omit the local Lipschitz continuity of the parametrization we can construct a ReLU-invariant one-parameter family of distributions in $\mathbb{R}^n$ for any dimension $n \in \mathbb{N}$. We proceed similar to the ideas in Section 5.3.1, 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.

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 main idea is quite straightforward and analogous to Section 5.3.1: 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 n-dimensional setting is to show that this leads to a continuous parametrization.

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

$$\mathbb{R}^\infty = \{\, (x_1, x_2, \ldots) \in \mathbb{R}^\omega \,:\, x_i \neq 0 \text{ for only finitely many } i \,\}$$

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 parametrize ReLU neural networks as functions in $C(K, \mathbb{R}^n)$ using a realization map $R\colon \Omega_\infty \to C(K, \mathbb{R}^n)$. Finally, we use a pushforward map $Q\colon C(K, \mathbb{R}^n) \to \mathcal{D}(\mathbb{R}^n)$ mapping neural network functions $\Phi \in C(K, \mathbb{R}^n)$ to the pushforward of a fixed *prototype* measure $\mu_0 \in \mathcal{D}(\mathbb{R}^n)$ under Φ 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 $p\colon \Omega \to \mathcal{D}(\mathbb{R}^n)$ 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 5.28. *There exists a continuous and surjective function* $\Gamma\colon \mathbb{R} \to \mathbb{R}^\infty$.

The construction of Γ is deferred to Appendix C.2.

Neural Network Parametrization

We extend the realization map from Section 5.3.1 to arbitrary dimensions and arbitrary numbers of layers. Further we will show that the results in [PRV18] concerning continuous network parametrizations can be extended to networks of arbitrary depth.

A ReLU neural network in n dimensions is characterized by the number of its layers as well as $n^2 + n$ 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 parametrize all such neural networks with the set

$$\Omega_\infty = \{\, (L, x_1, x_2, \ldots) \in \mathbb{R}^\infty \,:\, L \in \mathbb{N} \text{ and } x_i = 0 \text{ for all } i > L(n^2 + n) \,\} \subseteq \mathbb{R}^\infty.$$

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 $\Omega_L = \{L\} \times \mathbb{R}^{L(n^2+n)} \times \{0\}^\infty \subseteq \Omega_\infty$ and observe that

$$\Omega_\infty = \bigcup_{L \in \mathbb{N}} \Omega_L.$$

In fact this is the partition of Ω_∞ into its connected components and we will use Ω_L to parametrize $\mathcal{NN}^n_{L,\varrho}$.

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 n^2 and n 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 $\mathbf{W}_i \in \mathbb{R}^{n\times n}$ and $\mathbf{b}_i \in \mathbb{R}^n$. As before let $K \subseteq \mathbb{R}^n$ be any non-empty compact domain and denote the function that is the ReLU neural network realization of the weights $\{\mathbf{W}_1, \dots, \mathbf{W}_L\}$ and biases $\{\mathbf{b}_1, \dots, \mathbf{b}_L\}$ as $\Phi[\theta]\colon K \to \mathbb{R}^n$. Then we can define the extended realization map

$$R\colon \Omega_\infty \to C(K, \mathbb{R}^n)\colon \theta \mapsto \Phi[\theta].$$

Lemma 5.29. *The extended realization map $R\colon \Omega_\infty \to C(K, \mathbb{R}^n)$ is continuous.*

Proof. To show the continuity of $R\colon \Omega_\infty \to C(K, \mathbb{R}^n)$ 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 realization map is continuous from Ω_L to $C(K, \mathbb{R}^n)$. □

Pushforward Mapping

We extend our pushforward map from Section 5.3.1 to arbitrary dimension. As before let $\mu_0 \in \mathcal{D}(\mathbb{R}^n)$, the prototype, be any probability measure on $\mathbb{R}^n$ with $\operatorname{supp}(\mu_0) \subseteq K$. We define the pushforward function analogous to before as

$$Q\colon C(K, \mathbb{R}^n) \to \mathcal{D}(\mathbb{R}^n)\colon \Phi \mapsto \Phi_* \mu_0.$$

The restriction of Q to the subset of ReLU neural network functions will be used to generate the invariant family of distributions.

Lemma 5.30. *The pushforward function $Q\colon C(K, \mathbb{R}^n) \to \mathcal{D}(\mathbb{R}^n)$ is continuous.*

The proof works exactly as in the one-dimensional case in Section 5.3.1.

ReLU-Invariance

We can now define the one-parameter family of distributions as $p = Q \circ R \circ \Gamma$ 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 network functions in n dimensions. The continuity of p is clear from the three previous steps. It remains to establish the ReLU-invariance.

Lemma 5.31. *The one-parameter family of distributions $p\colon \Omega \to \mathcal{D}(\mathbb{R}^n)\colon \theta \mapsto (Q \circ R \circ \Gamma)(\theta)$ is ReLU-invariant.*

Proof. Let $\theta \in \Omega$ be arbitrary and $p(\theta) \in \mathcal{D}(\mathbb{R}^n)$ the corresponding probability measure. Further, let $\Phi \in \mathcal{NN}^n_{1,\varrho}$ be any ReLU layer. We need to show that there exists a $\omega \in \Omega$ such that $p(\omega) = \Phi_* p(\theta)$.

By construction we have $p(\theta) = (Q \circ R \circ \Gamma)(\theta) = (R(\Gamma(\theta)))_* \mu_0 = \widetilde{\Phi}_* \mu_0$, where $\widetilde{\Phi} = R(\Gamma(\theta))$ is a ReLU neural network. Clearly also $\Phi \circ \widetilde{\Phi}$ is a ReLU neural network, so there exists $\omega \in \Omega$ such that $R(\Gamma(\omega)) = \Phi \circ \widetilde{\Phi}$. Finally,

$$p(\omega) = (\Phi \circ \widetilde{\Phi})_* \mu_0 = \Phi_* \widetilde{\Phi}_* \mu_0 = \Phi_* p(\theta)$$

yields the claim. □

5.4 Discussion

We have limited our analysis in this chapter to the class of functions defined as layers of ReLU neural networks, as this is generally an extremely important class of functions and also the one for which ADF was used within the context of this thesis, cf. Section 4.3.4. Similarly, one could ask the same question for other function classes.

Different Activations. The constructive steps in the proof of Lemma 5.13 (surjectivity of Ξ) make explicit use of the fact that the ReLU is a continuous, piecewise linear function. Alternative continuous, piecewise linear activations, for example the leaky ReLU [MHN13] with a nonzero left-side slope, should allow for analogous constructions. Different commonly used activation functions are the hyperbolic tangent function, the logistic function, or the Heaviside function. Since the Heaviside function is not continuous and has a discrete and finite range, it transforms any distribution into a distribution with finite support. So clearly also in this case the only invariant distributions can be sampling distributions. For smooth activation functions, such as hyperbolic tangent and the logistic function, we are not aware of any characterization 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.

Restricted Weights and Biases. A second variation on the class of functions would be to put restrictions on the weight matrices or bias vectors. These 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 more detail in Appendix C.3. Interestingly, allowing only a finite or even countable set of weights and biases results in a possible parametrization of invariant distributions that circumvent the three restrictions (R1)–(R3) in Theorem 5.5. This is because the resulting function collection $\mathcal{F}$ is then countable as well, in contrast to the continuum of functions we considered until now. To construct an invariant family in this case, we can start with a prototype measure, analogously to Section 5.3.1 and Section 5.3.3. This gets mapped to a countable set of push-forward measures by

functions in $\mathcal{F}$. These can be locally Lipschitz-continuously parametrized making use of interpolation functions based on tree-like graphs, see Figure C.1 in Appendix C.3. These parametrizations, however, are purely of theoretical interest, since they rely on calculating and interpolating infinitely many distributions and are thus not of practical relevance.

Approximate Invariance. Finally, we want to comment that exact invariance might not be necessary for numerical computations in many application. Instead a notion of approximate invariance could be sufficient if the resulting errors can be reasonably bounded for every network layer. So far, to the best of our knowledge, no distribution families are known for which meaningful bounds exist. Whether this can indeed be realized is left for future investigations.

5.5 Conclusion

This chapter gives a comprehensive characterization 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 approximation schemes such as Assumed Density Filtering (ADF) or Expectation Propagation (EP) for applications involving, e.g., uncertainty quantification or interpretability methods for neural networks and Bayesian networks.

II

Reconstruction Tasks

6

Robustness of Reconstruction Methods

After discussing various questions connected to the interpretability of neural network classifier functions in the first part of this thesis we now turn toward the problem of reconstructing signals from incomplete measurements. In this chapter we consider another important aspect of neural network functions that was first observed in the classification setting, namely a potential lack of robustness, and examine whether it carries over to the setting of reconstruction problems.

The existence of so called *adversarial examples* or *adversarial perturbations* for classification neural networks is a well documented phenomenon in machine learning research. The networks are susceptible to small perturbation of the input signals that, although mostly unnoticeable to the naked eye, can have large effects on the networks output and completely change the predicted class. Starting with the initial observations of Szegedy et al. [Sze+14] there has been a substantial amount of work devoted to *adversarial attacks*, that is the finding of these perturbations. Most commonly, the "attacker" exploits gradient information to cross the discontinuous decision boundary of the network, while trying to stay as close as possible to the original unperturbed input. In response to this, an almost equally large research effort has been made to develop *defense strategies* that aim at making the trained networks more robust and less susceptible to adversarial attacks. For example, *adversarial training* exposes the network to perturbed inputs already during the training phase [GSS15]. This ongoing back and forth between novel attack and defense strategies led to the very active research field of *adversarial machine learning*. Until now, even the most elaborate defenses could eventually be defeated by more powerful attacks [MSG20]. It is not hard to imagine that this phenomenon undermines the reliable use of neural networks in sensitive applications and can pose a serious security threat, e.g., if a stop sign is misclassified in autonomous driving.

Recently, similar attack strategies have been investigated also in the context of learned reconstruction methods for inverse problems. Recall that we consider linear and finite-dimensional inverse problems:

$$\left\{\begin{array}{l} \textit{Given a linear forward operator } \mathbf{A} \in \mathbb{R}^{m\times n} \\ \textit{and corrupted measurements } \mathbf{y} = \mathbf{A}\mathbf{x}_0 + \mathbf{e} \\ \textit{with } \|\mathbf{e}\|_2 \leq \eta, \textit{ reconstruct the signal } \mathbf{x}_0. \end{array}\right\} \qquad \text{(IP, restated)}$$

Initial works present evidence that learned reconstruction schemes for (IP) based on neural networks might be unstable as well [Ant+20; Got+20; Hua+18]. In particular, Antun et al. observe that using deep learning for inverse problems comes at the cost of instabilities, in the sense that „[...] certain tiny, almost undetectable perturbations, both in the image and sampling domain, may result in severe artifacts in the reconstruction [...]" [Ant+20]. Given that the indirect measurements are typically corrupted by noise, these findings raise doubts regarding the usage of deep learning in sensitive fields such as medical imaging.

It seems peculiar that well-performing reconstruction neural network should be unstable. Indeed, the setting of classification tasks is fundamentally different from solving inverse problems. The former is an inherently discontinuous task: neural networks map high-dimensional signals to a low-dimensional discrete output space (the class labels). This necessarily results in sharp decision boundaries that can be crossed by adding perturbations to the input signals. In contrast, the latter is a regression task, requiring to map measurements to a high-dimensional signal space. In many cases it can be shown that this task is robustly solvable by classical model-based reconstruction methods [FR13], in the sense that error bounds of the form

$$\|\mathbf{x}_0 - \mathrm{Rec}(\mathbf{y})\|_2 \leq C \cdot \eta \qquad \text{(1.2, restated)}$$

hold, which scale linearly with the strength of of the noise. Such a bound requires that the reconstruction error approaches zero as the strength of the noise becomes arbitrarily small. The *accuracy* of a reconstruction method usually refers to the noise-free limit $\eta \to 0$. This case is evaluated and discussed in Chapter 7, while we generally consider *noisy* inverse problems in this chapter.

It is unclear why training neural networks for solving (IP) should lead to accurate but unstable methods, especially since model-based iterative algorithms often serve as inspirations for designing the network architectures. In the absence of theoretical recovery guarantees for learned methods we believe that a sound empirical verification of robustness is crucial. This chapter is dedicated to an extensive empirical study regarding the robustness of data-driven solution methods for inverse problems. A key component is a quantitative comparison to a classical reconstruction method for which a recovery guarantee as in (1.2) can be proven in certain situations.

Related Work

Initiated by Szegedy et al. [Sze+14], the study of the vulnerability of deep neural networks to adversarial examples has emerged as an active research area.[1] The vast majority of existing articles is concerned with classification and related tasks, such as image segmentation [AMT20]. On the other hand, only few works have explicitly addressed the adversarial robustness of learned solvers for inverse problems.

To the best of our knowledge, Huang et al. have made the first effort to transfer adversarial attacks to neural-network-based reconstruction methods [Hua+18]. They demonstrate that a distortion of the network inputs may result in the loss of small features in image signals. However, their initial findings are restricted to the specific problem of limited angle computed tomography, where the robust recovery of certain parts of the image is provably impossible [Qui93]. Moreover, the proposed perturbation model is non-standard and does not correspond to noise in the measurements.

More recently, the topic was brought to attention by the inspiring article of Antun et al. [Ant+20]. Their numerical experiments show instabilities of existing deep neural networks with respect to adversarial noise, out-of-distribution features, and changes in the number of measurements. An important difference to our work is that adversarial noise is only computed for learned schemes. We believe that a comparative "attack" of a

[1] An online collection compiled by Carlini [Car22] reports almost 5000 works related to adversarial examples (as of March 2022). For a systematic overview, we refer to the recent reviews [Yua+19; MSG20; Ort+20].

classical benchmark method is crucial for a fair assessment of robustness. Furthermore, the results of [Ant+20] are reported qualitatively by visualizing reconstructed images, as it is common in adversarial machine learning. We argue that the mathematical setup of the inverse problem (IP) calls for a quantitative error analysis that is in line with the bound of (1.2). Finally, the training stage of the networks in [Ant+20] does not seem to account for noise, which we have identified as a potential source of instability, see Section 6.3.1. Note that our study also analyzes the FBPConvNet architecture [Jin+17], a relative of AUTOMAP [Zhu+18], and an iterative scheme similar to DeepMRI [Sch+17]. Nevertheless, a one-to-one comparison to [Ant+20] is subtle due to task-specific architectures and data processing. A follow-up work of [Ant+20] presents a theoretical characterization of instabilities in terms of the kernel of the forward operator [Got+20]. Our results provide empirical evidence that the considered deep-learning-based schemes could be kernel aware (cf. Section 6.3.4).

As a countermeasure to the outcome of [Ant+20], Raj, Bresler, and Li suggest a sophisticated defense strategy resulting in robust networks [RBL20]. This work also addresses shortcomings of the attack strategy in [Ant+20], see Section 6.1.4 for details. In line with our findings, Kobler et al. propose the data-driven *total deep variation* regularizer and demonstrate its adversarial robustness for image denoising [Kob+20].

Finally, in another line of research, [SAH20] conducts a theoretical error analysis of a family of mappings (referred to as *RegNets*) that post-process classical regularization methods. Under the assumption of Lipschitz-continuous networks, convergence rates in the spirit of (1.2) are derived for the limit $\eta \to 0$. However, due to their asymptotic nature, such results do not directly address the adversarial perturbation scenarios of this chapter, where a whole range of noise levels η is analyzed. Apart from that, the convergence rates in [SAH20] linearly depend on the global Lipschitz constants, which are hard to compute and control in practice [Sze+14; VS18; HCC18]. Our simulations reveal that *pointwise* Lipschitz constants for common reconstruction networks are well-behaved, regardless of possibly large global constants.

6.1 Methods and Preliminaries

In this section, we briefly introduce the considered reconstruction schemes for solving the inverse problem (IP). This includes a representative selection of neural-network-based methods and total-variation minimization as a classical benchmark. Furthermore, our attack strategy to analyze their adversarial robustness is presented.

6.1.1 Neural Network Architectures

In the past five years, numerous deep-learning-based approaches for solving inverse problems have been developed, see [Arr+19; Ong+20] for overviews. This chapter focuses on a selection of widely used *end-to-end network schemes* that define an explicit reconstruction map from $\mathbb{R}^m$ to $\mathbb{R}^n$, see also Figure 6.1.

The first considered method is a *post-processing network*:

$$\mathrm{UNet}\colon \mathbb{R}^m \to \mathbb{R}^n,\ \mathbf{y} \mapsto [\mathbf{U} \circ \mathbf{A}^\ddagger](\mathbf{y}).$$

It employs the U-Net architecture $\mathbf{U}\colon \mathbb{R}^n \to \mathbb{R}^n$ [RFB15], illustrated in Figure 6.2, as a

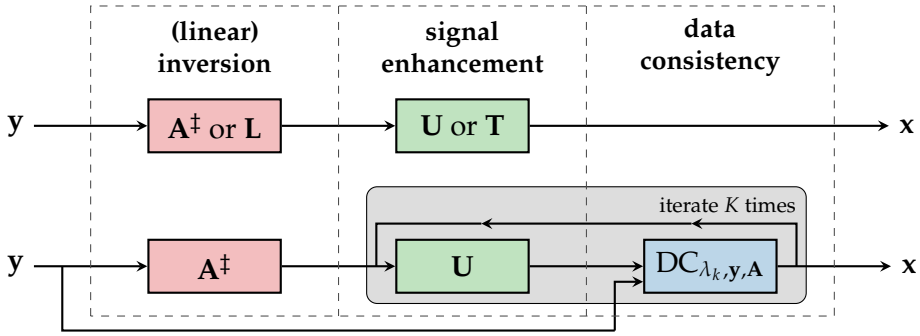


Figure 6.1: **Data-driven Reconstruction Methods.** Schematic network reconstruction pipelines of UNet, TiraFL (top), and ItNet (bottom).

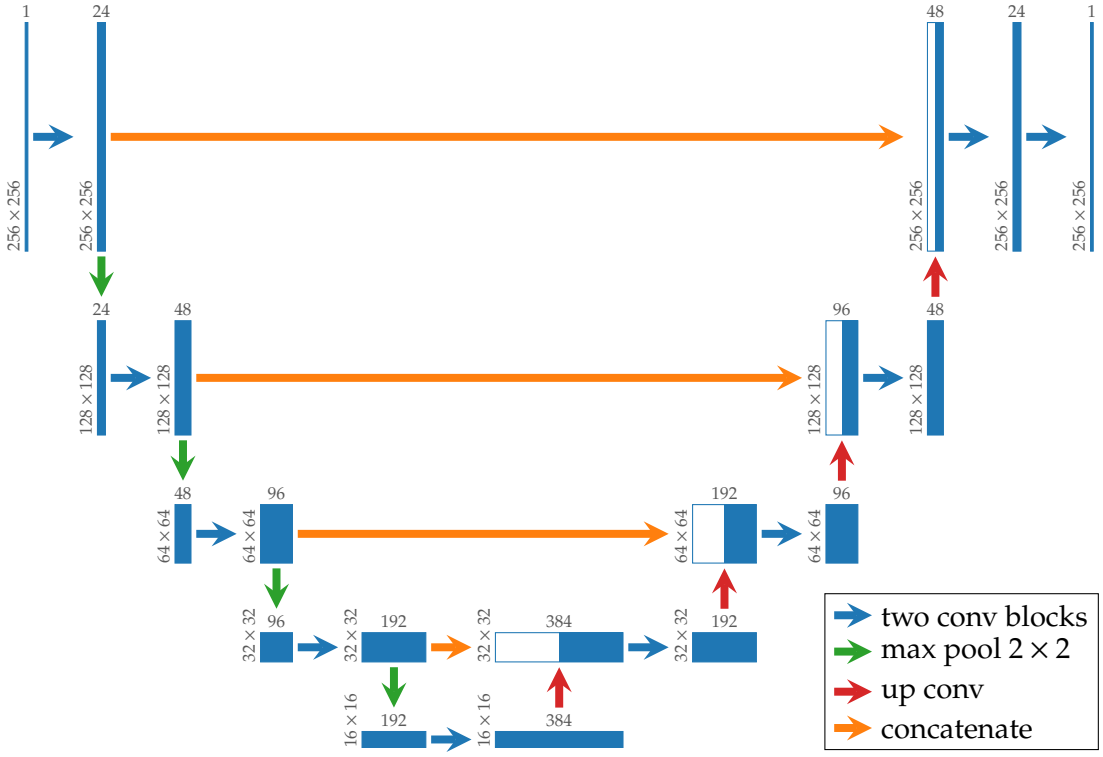


Figure 6.2: **U-Net Architecture.** Each convolutional block consists of a 3×3 convolution, followed by batch normalization, and ReLU activation. Figure adapted from [RFB15].

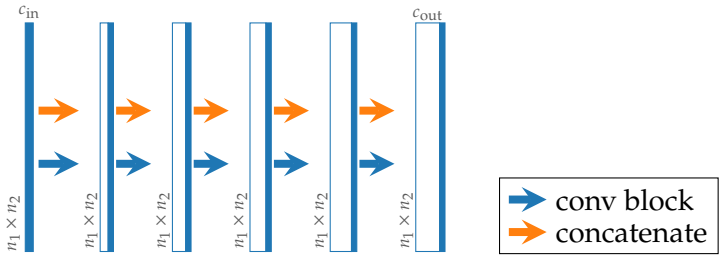


Figure 6.3: **Dense Blocks.** Blocks with five convolutional sub-blocks are used in the Tiramisu architecture. Each convolutional sub-blocks consists of a 3×3 convolution, followed by batch normalization, and ReLU activation. It receives a concatenation of the output of the previous convolutional block (filled box) together with the inputs to all previous blocks (hollow box) as input.

residual network [He+16] to enhance an initial, model-based reconstruction $\mathbf{A}^\ddagger(\mathbf{y})$. Here, $\mathbf{A}^\ddagger \colon \mathbb{R}^m \to \mathbb{R}^n$ is an approximate inversion of the forward operator $\mathbf{A}$, e.g., the filtered back-projection for Radon measurements. Despite its simplicity, it has been demonstrated that UNet is an effective solution method for (IP) [Jin+17], see also [KMY17; Che+17a; Che+17b; Wol+17; Yan+18] for related approaches.

Our second reconstruction scheme is a *fully-learned network:*

$$\text{TiraFL} \colon \mathbb{R}^m \to \mathbb{R}^n, \ \mathbf{y} \mapsto [\mathbf{T} \circ \mathbf{L}](\mathbf{y}),$$

which is closely related to UNet, but differs in two aspects: It is based on the Tiramisu architecture $\mathbf{T} \colon \mathbb{R}^n \to \mathbb{R}^n$ [Jég+17] as a residual network, which can be seen as a refinement of the standard U-Net. While $\mathbf{T}$ shares the same multi-level structure, it is built from fully-convolutional dense-blocks [Hua+17] instead of standard convolutional blocks, see Figure 6.3. More importantly, the fixed inversion $\mathbf{A}^\ddagger$ is replaced by a learnable linear layer $\mathbf{L} \in \mathbb{R}^{n \times m}$, so that TiraFL does not contain fixed model-based components anymore. The approach of TiraFL is similar to [Zhu+18; Sch+19], which makes use of a fully-learned reconstruction map for MRI. For the sake of completeness, we also conduct experiments for Tira (a Tiramisu-based post-processing network) as well as for UNetFL (a U-Net-based fully-learned network), see Appendix D for results.

Finally, we also analyze an *iterative network:*

$$\text{ItNet} \colon \mathbb{R}^m \to \mathbb{R}^n, \ \mathbf{y} \mapsto \left[\left(\bigcirc_{k=1}^{K} [\text{DC}_{\lambda_k, \mathbf{y}, \mathbf{A}} \circ \mathbf{U}] \right) \circ \mathbf{A}^\ddagger \right] (\mathbf{y})$$

where

$$\text{DC}_{\lambda_k, \mathbf{y}, \mathbf{A}} \colon \mathbb{R}^n \to \mathbb{R}^n, \ \mathbf{x} \mapsto \mathbf{x} - \lambda_k \cdot \mathbf{A}^* \left(\mathbf{A}\mathbf{x} - \mathbf{y} \right).$$

The scalar parameters λ_k are learnable and $\mathbf{A}^*$ denotes the adjoint of $\mathbf{A}$. Mathematically, $\text{DC}_{\lambda_k, \mathbf{y}, \mathbf{A}}$ performs a gradient step on the loss $\mathbf{x} \mapsto \frac{\lambda_k}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$, promoting *data consistent* solutions. Therefore, the alternating cascade of ItNet can be seen as a proximal gradient descent scheme, where the proximal operator is replaced by a trainable enhancement network. Here, the U-Net architecture is used again, due to its omnipresence in image-to-image processing tasks. Unrolled methods in the spirit of ItNet are frequently used to solve inverse problems, e.g., see [GL10; Yan+16; Ham+18; Sch+17; AMJ18; AÖ18; Ham+19; Chu+20].

6.1.2 Neural Network Training

All the learnable parameters of the neural networks are trained from sample data pairs $\{(\mathbf{y}^i = \mathbf{A}\mathbf{x}_0^i + \mathbf{e}^i, \mathbf{x}_0^i)\}_{i=1}^M$ by minimizing an empirical loss function. Depending on the use case, the signals $\mathbf{x}_0^i$ are either drawn from a fixed publicly available training dataset or according to a synthetic probability distribution. If $\mathrm{Net}[\boldsymbol{\theta}]\colon \mathbb{R}^m \to \mathbb{R}^n$ denotes a reconstruction network with all learnable parameters collected in $\boldsymbol{\theta}$, then the training amounts to (approximately) solving

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{M} \ell\left(\mathrm{Net}[\boldsymbol{\theta}](\mathbf{y}^i), \mathbf{x}_0^i\right) + \alpha \cdot \|\boldsymbol{\theta}\|_2^2 \tag{6.1}$$

for some cost function $\ell\colon \mathbb{R}^n \times \mathbb{R}^n \to [0, \infty]$, which in this chapter is the squared distance unless stated otherwise. Overfitting is addressed by ℓ_2-regularization with a hyper-parameter $\alpha \geq 0$.[2] In order to solve (6.1), we utilize mini-batch stochastic gradient descent and the Adam optimizer [KB14]. We found that larger mini-batches were beneficial for the training performance during later epochs.[3] Technically, this is achieved by gradient accumulation, i.e., the gradient is cumulatively summed over several mini-batches before executing a descent step.

Due to the ubiquitous presence of noise in inverse problems, it is natural to account for it in the training data. In many applications, measurement noise is modeled as an independent random variable, for instance, following a Gaussian distribution. Therefore, the perturbation $\mathbf{e}^i$ is treated as statistical noise during the training phase, i.e., a fresh realization is randomly drawn in each epoch. This technique is well known as *jittering* in machine learning research, where it is primarily used to avoid overfitting [SD91; HK92; Bis95], see also [Vin+10]. In Section 6.3.1, we relate jittering to the phenomenon of inverse crimes and demonstrate its importance for the robustness of learned reconstruction schemes. Due to varying noise levels in the evaluation of our models, we design $\mathbf{e}^i$ as a centered Gaussian vector with random variance, such that its expected norm $\mathbb{E}\|\mathbf{e}^i\|_2$ is distributed uniformly in a range $[0, \widetilde{\eta}]$ for some $\widetilde{\eta} \geq 0$.

6.1.3 Total Variation Minimization

Dating back to the seminal work of Rudin, Osher, and Fatemi [ROF92], *total-variation (TV) minimization* has become a standard tool for solving signal and image reconstruction tasks [CL97; BB18]. We apply it to the problem (IP) in the following form:

$$\mathrm{TV}[\eta]\colon \mathbb{R}^m \to \mathbb{R}^n,\ \mathbf{y} \mapsto \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n} \|\nabla \mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2 \leq \eta, \tag{6.2}$$

where ∇ denotes a discrete gradient operator. Crucial to the above optimization problem is the use of the ℓ_1-norm, which is known to promote gradient-sparse solutions. Indeed, under suitable assumptions on $\mathbf{A}$, compressed sensing theory suggests an error bound of the form (1.2) for a gradient-sparse signal $\mathbf{x}_0$ and $\mathrm{Rec} = \mathrm{TV}[\eta]$, e.g., see [CRT06; NW13; Poo15; GMS21]. In other words, TV minimization is provably robust with a near-optimal

[2] This is often referred to as *weight decay* in deep learning, since the ℓ_2-term corresponds to a shrinkage of the weights $\boldsymbol{\theta}$ by a constant factor when performing the gradient update, e.g., see [GBC16, Sec. 7.1.1].

[3] One pass through all sample data pairs is called a training *epoch*.

dependence on η. This particularly justifies its use as a reference method, allowing us to empirically characterize the robustness of learned reconstruction schemes.

In our numerical simulations, the problem of (6.2) is solved by the *alternating direction method of multipliers (ADMM)* [GM75; GM76]. For 1D signals, $\nabla \in \mathbb{R}^{n\times n}$ is chosen as a forward finite difference operator with Neumann boundary conditions, extended by a constant row vector to capture the mean of the signal. For image signals, $\nabla \in \mathbb{R}^{2n\times n}$ corresponds to a 2D forward finite difference operator with periodic boundary conditions. Due to the non-separability of $\|\nabla(\cdot)\|_1$ in 2D, the formulation of TV[η] in (6.2) becomes computationally infeasible for finding adversarial noise. In imaging scenarios, we therefore solve the unconstrained version of TV[η] instead, i.e., the objective function is changed to $\mathbf{x} \mapsto \lambda \cdot \|\nabla \mathbf{x}\|_1 + \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$. Note that this strategy is theoretically equivalent [FR13, Appx. B], but requires an appropriate choice of the regularization parameter $\lambda > 0$. A near-optimal selection with respect to the relative ℓ_2-error is determined by grid searches over the test set and a densely sampled range of noise levels η. In any case, we emphasize that TV[η] is explicitly adapted to the amount of perturbation in the measurements.

6.1.4 Adversarial Perturbations

In the setup of (IP), adversarial noise for a given reconstruction method $\mathrm{Rec}\colon \mathbb{R}^m \to \mathbb{R}^n$ can be computed by solving another optimization problem: for a fixed signal $\mathbf{x}_0 \in \mathbb{R}^n$ and noise level $\eta \geq 0$, find an additive perturbation $\mathbf{e}_{\mathrm{adv}} \in \mathbb{R}^m$ of the noiseless measurements $\mathbf{y}_0 = \mathbf{A}\mathbf{x}_0$ that maximizes the reconstruction error, i.e.,

$$\mathbf{e}_{\mathrm{adv}} = \operatorname*{argmax}_{\mathbf{e}\in\mathbb{R}^m} \|\mathrm{Rec}(\mathbf{y}_0 + \mathbf{e}) - \mathbf{x}_0\|_2 \quad \text{subject to} \quad \|\mathbf{e}\|_2 \leq \eta. \tag{6.3}$$

Such an attack strategy is a straightforward adaption of a common approach in adversarial machine learning [Yua+19]. In contrast to [Ant+20], we consider a constrained optimization problem that avoids shortcomings of an unconstrained formulation. In particular, this allows for precise control over the noise level. Moreover, (6.3) explores a natural perturbation model, operating directly in the measurement domain, cf. the discussion in [RBL20].

In order to solve the problem (6.3), we use the projected gradient descent algorithm in conjunction with the Adam optimizer, which was found to be most effective (cf. [CW17]). The non-convexity of (6.3) is accounted for by choosing the worst perturbation out of multiple runs with random initialization. Assuming a whitebox model (i.e., Rec is fully accessible), we use PyTorch's automatic differentiation [Pas+17] to compute gradients of the considered neural network schemes.

A central aspect of our work is that the above perturbation strategy is also applied to TV[η]. This is non-trivial, since the gradient of the implicit map $\mathbf{y} \mapsto \mathrm{TV}[\eta](\mathbf{y})$ has to be computed. The large-scale nature of imaging problems prevents us from using the recent concept of differentiable convex optimization layers [Agr+19]. Instead, we calculate the gradient of the unrolled ADMM scheme for TV minimization by automatic differentiation. In general, a large number of iterations might be required for the convergence of ADMM, which in turn ensures an accurate gradient approximation for TV[η]. This leads to numerical difficulties in automatic differentiation, due to memory and time constraints and error accumulation. We address this issue by decreasing the number of ADMM iterations for the gradient computation (denoted by k_{grad}). To compensate for a loss of accuracy, we

use a pre-initialization of the primal and dual variables by the output of a fully converged ADMM scheme with input $\mathbf{y}_0$. Such a warm start is beneficial, since the gradient is only evaluated in an η-ball around $\mathbf{y}_0$ during the attack. Note that the actual TV reconstructions are always computed by a fully converged ADMM algorithm (with $k_{\text{rec}} \gg k_{\text{grad}}$ iterations).

6.2 Results

This section studies the robustness of neural-network-based solution methods for three different instances of the inverse problem (IP). The goal of our experiments is to assess the loss of reconstruction accuracy caused by noise. To that end, we rely on two types of visualizations:

- *Noise-to-error curves* are generated by plotting the relative noise level $\eta/\|\mathbf{A}\mathbf{x}_0\|_2$ against the relative reconstruction error $\|\mathbf{x}_0 - \text{Rec}(\mathbf{A}\mathbf{x}_0 + \mathbf{e})\|_2/\|\mathbf{x}_0\|_2$.
- *Individual reconstruction results* are shown for different relative noise levels and a randomly selected signal from the test set.

In both cases, the perturbation vector $\mathbf{e}$ is either of *statistical* or *adversarial* type. The former means that $\mathbf{e}$ is a random vector such that $\mathbb{E}[\|\mathbf{e}\|_2^2] = \eta^2$, whereas the latter is found by (6.3). While noise-to-error curves are of quantitative nature, individual reconstructions facilitate a qualitative judgment of robustness. Note that the sensitivity to noise is different in each considered scenario. Therefore, we select the maximal level of adversarial noise such that the benchmark of TV minimization does not yield a (subjectively) acceptable performance anymore. A specification of all empirically selected hyper-parameters can be found in Appendix D (Tables D.9 to D.11).

6.2.1 Case Study A: Compressed Sensing With Gaussian Measurements

Our first study is devoted to sparse recovery of 1D signals from Gaussian measurements, which is a standard benchmark setup in the field of compressed sensing (CS) theory [FR13]. This means that the entries of the forward operator $\mathbf{A}$ in (IP) are independent Gaussian random variables with zero mean and variance $1/m$. We consider two different scenarios based on (approximately) gradient-sparse signals. Note that such a model is canonical for TV minimization and compatible with the local connectivity of our convolutional neural network schemes.

Scenario A1. We draw $\mathbf{x}_0$ from a synthetic distribution of *piecewise constant signals* with zero boundaries and well-controlled random jumps, see Figure 6.5 for an example. In this scenario, we choose $m = 100$, $n = 256$, and use $M = 200$k training samples.

Scenario A2. We sample $\mathbf{x}_0 \in [0,1]^{28\times 28}$ from the widely used *MNIST database* [LeC+98] with $M = 60$k training images of handwritten digits. In the context of (IP), the images are treated as 1D signals[4] of dimension $n = 28^2 = 784$. The number of Gaussian measurements is chosen as $m = 300$.

[4] We have decided for a vectorized data processing (i.e., TV[η] and the neural networks operate on vectorized images), since Scenario A2 is regarded as a direct continuation of the idealistic situation in Scenario A1. However, for visual purposes, all reconstructions are displayed as images, see Figure 6.7.

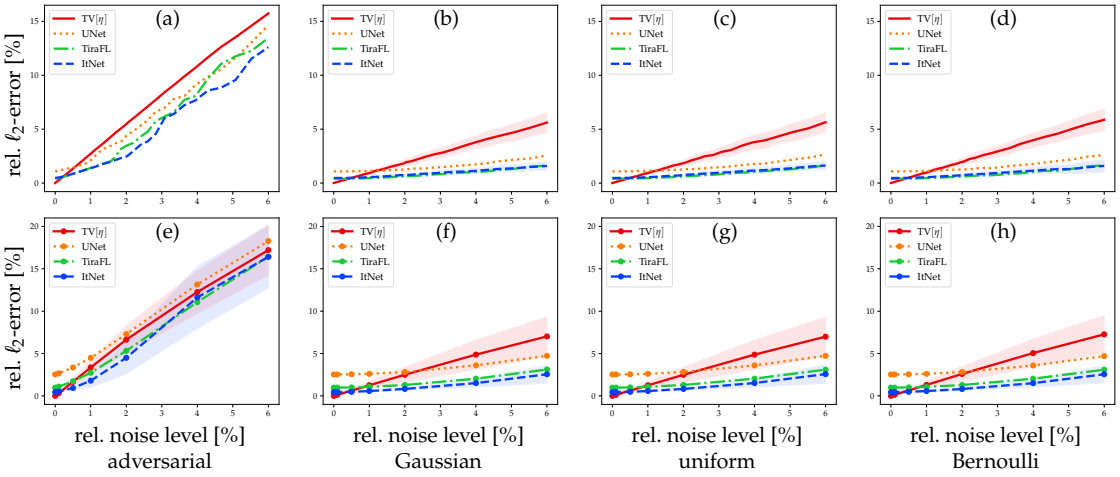


Figure 6.4: **Scenario A1 – CS with 1D Signals.** (a) shows the adversarial noise-to-error curve for the randomly selected signal of Figure 6.5. (b)–(d) show the corresponding noise-to-error curves for Gaussian, uniform, and Bernoulli (symmetrized, $p = 0.025$) noise respectively, where the mean and standard deviation are computed over 200 draws of $\mathbf{e}$. (e)–(h) display the respective curves averaged over 50 signals from the test set. For the sake of clarity, we omit the standard deviations for UNet and TiraFL, which behave similarly.

In both scenarios, we choose the model-based, linear inversion layer of the networks as a generalized Tikhonov matrix, i.e., $\mathbf{A}^{\ddagger} = (\mathbf{A}^{\top}\mathbf{A} + \alpha \cdot \nabla^{\top}\nabla)^{-1}\mathbf{A}^{\top} \in \mathbb{R}^{n\times m}$ with the empirically chosen regularization parameter $\alpha = 0.02$. We were not able to train the neural networks to a comparable reconstruction accuracy with other natural choices, such as $\mathbf{A}^{\ddagger} = \mathbf{A}^{\top}$. The above matrix is also used to initialize the inversion layer $\mathbf{L} \in \mathbb{R}^{n\times m}$ of the fully-learned schemes.

Figure 6.4 shows the noise-to-error curves for *Scenario A1 (CS with 1D Signals)* for adversarial perturbations and three types of random noise; see also Tables D.1 and D.2 in Appendix D. The results for the three different types of random noise are virtually indistinguishable from one another. The associated individual reconstructions for adversarial and Gaussian noise are displayed in Figure 6.5 and Figure 6.6 respectively. Figure 6.9 shows the noise-to-error curves for *Scenario A2 (CS with MNIST)*; see also Tables D.3 and D.4 in Appendix D. The associated individual reconstructions for adversarial and Gaussian noise are displayed in Figure 6.7 and Figure 6.8 respectively. The TV[η]-solutions show horizontal line artifacts due to the fact that the MNIST images are treated as vectorized 1D signals. Remarkably, although relying on 1D convolutional filters, the neural-network-based reconstructions do not suffer from these artifacts.

Conclusion

The above results confirm that the considered neural-network-based schemes are at least as robust to adversarial perturbations as the benchmark of TV minimization. Although TV[η] is perfectly tuned to each noise level η, it is clearly outperformed in the case of statistical noise. The gap between statistical and adversarial perturbations is comparable for all methods.

TV minimization is a perfect match for Scenario A1. In particular, exact recovery from

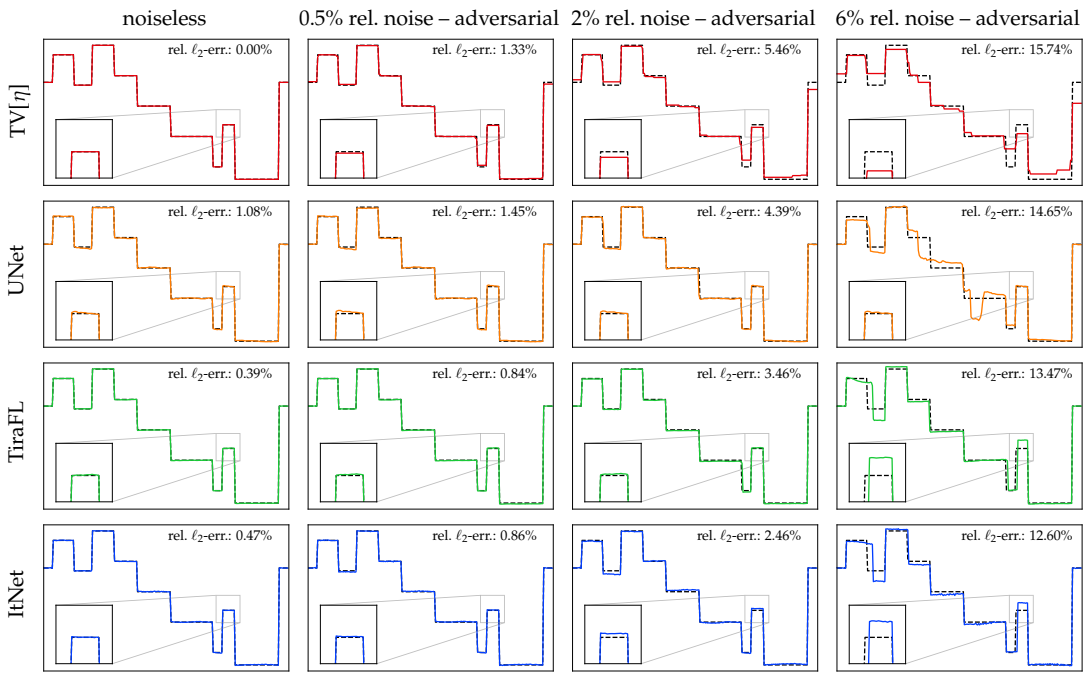


Figure 6.5: **Scenario A1 – CS with 1D Signals.** Individual reconstructions of a randomly selected signal from the test set for different levels of adversarial noise. The ground truth signal is visualized by a dashed line.

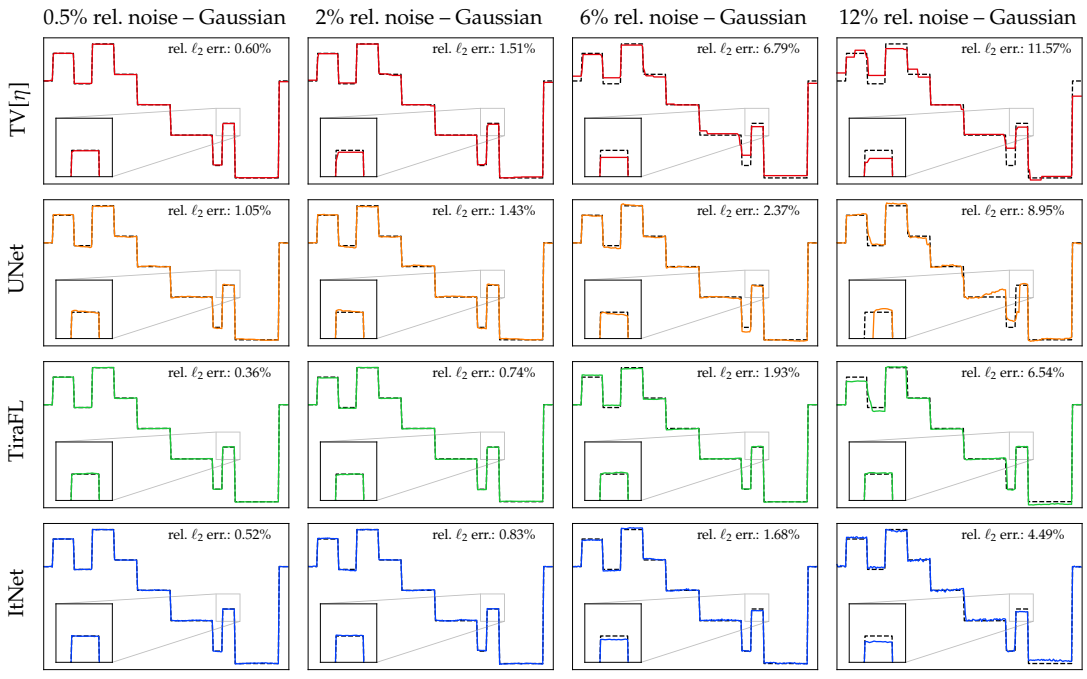


Figure 6.6: **Scenario A1 – CS with 1D Signals.** Individual reconstructions of the signal from Figure 6.5 under Gaussian noise. The ground truth signal is visualized by a dashed line. In favor of the more insightful noise level 12%, we omit the noiseless case.

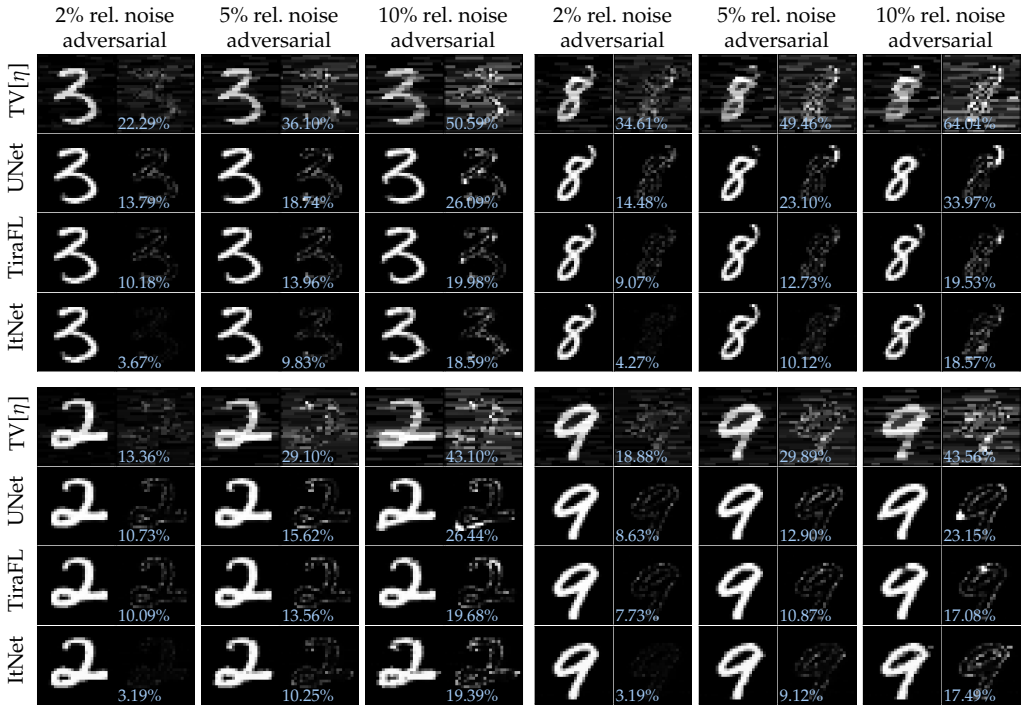


Figure 6.7: **Scenario A2 – CS with MNIST.** Individual reconstructions of four randomly selected digits from the test set for different levels of adversarial noise. Reconstructions and error plots (with relative ℓ_2-error) are displayed in the windows $[0, 1]$ and $[0, 0.6]$, respectively.

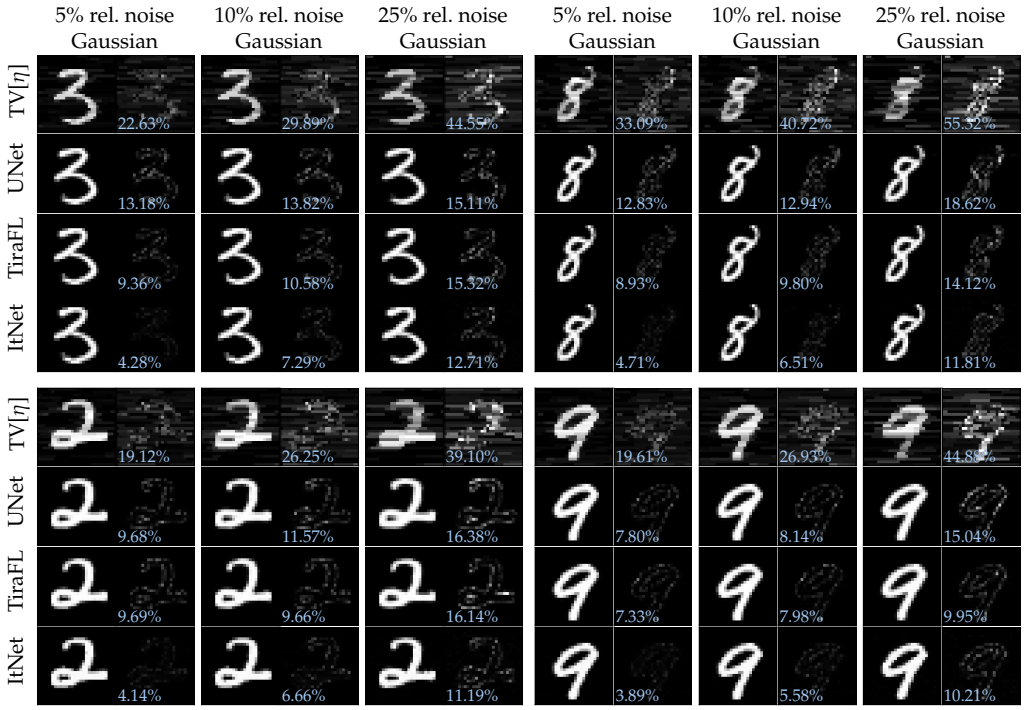


Figure 6.8: **Scenario A2 – CS with MNIST.** Reconstructions of the digits from Figure 6.7 under Gaussian noise. Reconstructions and error plots (with rel. ℓ_2-error) are displayed in the windows $[0, 1]$ and $[0, 0.6]$, respectively. In favor of the more insightful noise level 25%, we omit 2% noise.

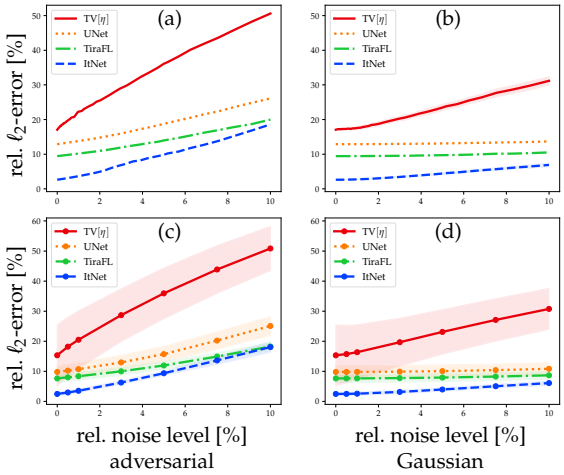


Figure 6.9: **Scenario A2 – CS with MNIST.** (a) shows the adversarial noise-to-error curve for the randomly selected digit three of Figure 6.7. (b) shows the corresponding Gaussian noise-to-error curve, where the mean and standard deviation are computed over 200 draws of $\mathbf{e}$. (c) and (d) display the respective curves averaged over 50 signals from the test set.

noiseless measurements is guaranteed by CS theory [Ame+14; GMS21]. Although this cannot be expected for neural-network-based solvers, they still come with an overall superior robustness against noise. The situation is even more striking in Scenario A2. Here, TV minimization performs worse, since the signals are only approximately gradient-sparse. In contrast, the neural-network-based reconstruction schemes adapt well to the simple MNIST database, leading to significantly better outcomes in every regard. Hence, the increase in accuracy by learned methods does not necessarily imply a loss of robustness.

The performance ranking of the considered deep neural networks is as one might expect: First, data consistency as encouraged by the ItNet-architecture is beneficial. Furthermore, Tables D.1 to D.4 in Appendix D reveal that the Tiramisu architecture is superior to a simple U-Net, and that a learnable inversion layer improves the recovery. The latter observation is not surprising, since Tikhonov regularization is known to work poorly in conjunction with subsampled Gaussian measurements.

6.2.2 Case Study B: Image Recovery of Phantom Ellipses

Our second set of experiments concerns the recovery of phantom ellipses from Fourier or Radon measurements. These tasks correspond to popular simulation studies for biomedical imaging, e.g., see [Jin+17; AÖ17; AÖ18; Bub+19]. We sample $\mathbf{x}_0 \in [0,1]^{256\times256}$ from a distribution of superimposed random ellipses with mild linear intensity gradients and well-controlled geometric properties, see Figure 6.11 for an example. The training is performed on $M = 25$k images. We consider the following two measurement scenarios for (IP), associated with the problems of *compressed sensing MRI* [Lus+08] and *low-dose computed tomography (CT)* [SP08; Jin+17], respectively:

Scenario B1. The forward operator takes the form $\mathbf{A} = \mathbf{PF} \in \mathbb{C}^{m\times n}$, where $\mathbf{F} \in \mathbb{C}^{n\times n}$ is the *2D discrete Fourier transform* and $\mathbf{P} \in \{0,1\}^{m\times n}$ is a subsampling operator defined by a golden-angle radial mask with 40 lines ($m = 10941$ and $n = 256^2 = 65536$). Note that the

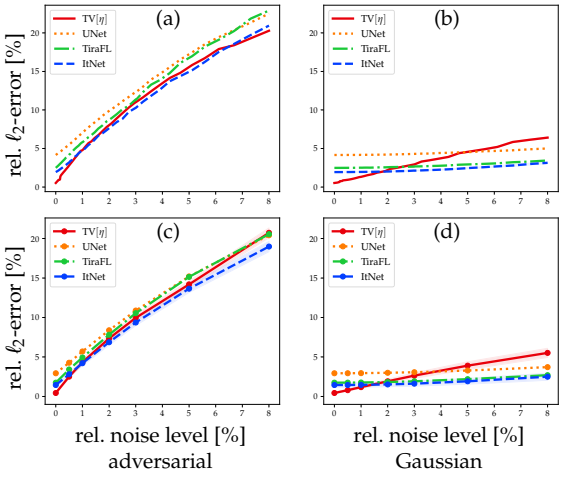


Figure 6.10: **Scenario B1 – Fourier Measurements with Ellipses.** (a) shows the adversarial noise-to-error curve for the randomly selected image of Figure 6.11. (b) shows the corresponding Gaussian noise-to-error curve, where the mean and (almost imperceptible) standard deviation are computed over 50 draws of $\mathbf{e}$. (c) and (d) display the respective curves averaged over 50 images from the test set. For the sake of clarity, we omit the standard deviations for UNet and TiraFL, which behave similarly.

entire data processing is complex-valued, while the actual reconstructions are computed as real-valued magnitude images, as common in MRI. We use the canonical inversion layer $\mathbf{A}^\ddagger = \mathbf{A}^* = \mathbf{F}^{-1}\mathbf{P}^\top \in \mathbb{C}^{n\times m}$.

Scenario B2. The forward operator $\mathbf{A} \in \mathbb{R}^{m\times n}$ is given by a sparse-angle *Radon transform* with 60 views ($m = 21780$ and $n = 65536$).[5] The non-linear inversion layer $\mathbf{A}^\ddagger\colon \mathbb{R}^m \to \mathbb{R}^n$ is chosen as the filtered back-projection algorithm (FBP) with a Hann filter.

In contrast to Case Study A, the aforementioned problems are of significantly higher dimensionality. Therefore, fully-learned schemes are difficult to realize, since the size of the inversion layer scales multiplicatively in the image dimensions. In the Fourier case, the number of free parameters can be reduced by enforcing a Kronecker product structure on $\mathbf{L} \in \mathbb{C}^{n\times m}$; this exploits the fact that $\mathbf{F}$ is a tensor product of two 1D Fourier transforms, cf. [Sch+19]. We do not consider fully-learned schemes in the case of Radon measurements.

Figure 6.10 shows the noise-to-error curves for *Scenario B1 (Fourier Measurements with Ellipses)*; see also Tables D.5 and D.6 in Appendix D. The associated individual reconstructions with adversarial and Gaussian noise are displayed in Figure 6.11 and Figure 6.12 respectively. In the tables and individual reconstructions, we also report the *peak signal-to-noise ratio (PSNR)* and *structural similarity index measure (SSIM)* [Wan+04]. In the case of *Scenario B2 (Radon Measurements with Ellipses)*, we only present individual reconstructions based on TV[η] and UNet; see Figure 6.13 for adversarial noise and Figure 6.14 for the common Poisson noise model. This restriction is due to the more complicated nature of the Radon transform, and in particular, the need for automatic differentiation. The used implementation [Ern20] requires significantly more computational effort, compared to the fast Fourier transform.

[5] See Chapter 7 for a more detailed description of the Radon transform as a model for computed tomography.

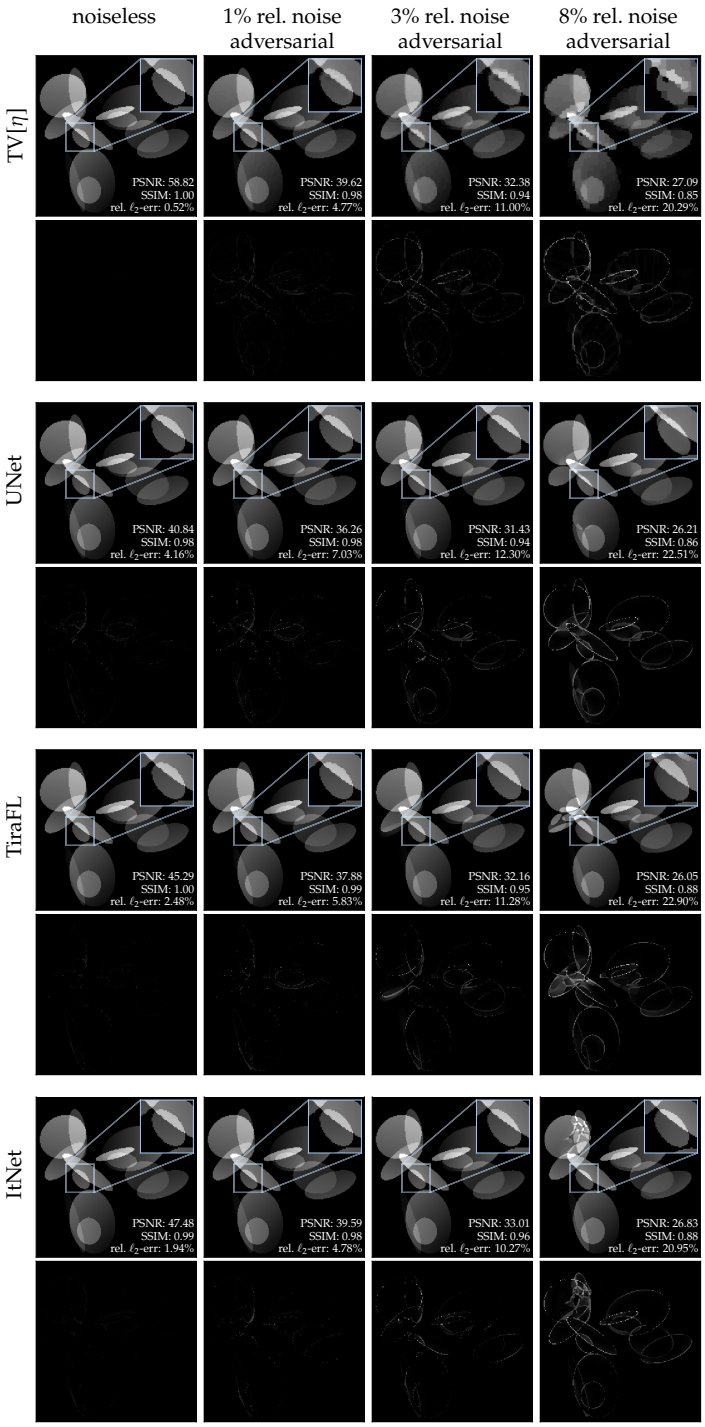


Figure 6.11: **Scenario B1 – Fourier Measurements with Ellipses.** Individual reconstructions of a randomly selected image from the test set for different levels of adversarial noise. The reconstructed images are displayed in the window $[0, 0.9]$, also used for the computation of the PSNR and SSIM. The error plots shown below each reconstruction are displayed in the window $[0, 0.6]$. The ground truth image $\mathbf{x}_0$ is omitted, as it is visually indistinguishable from the noiseless reconstruction by TV$[\eta]$.

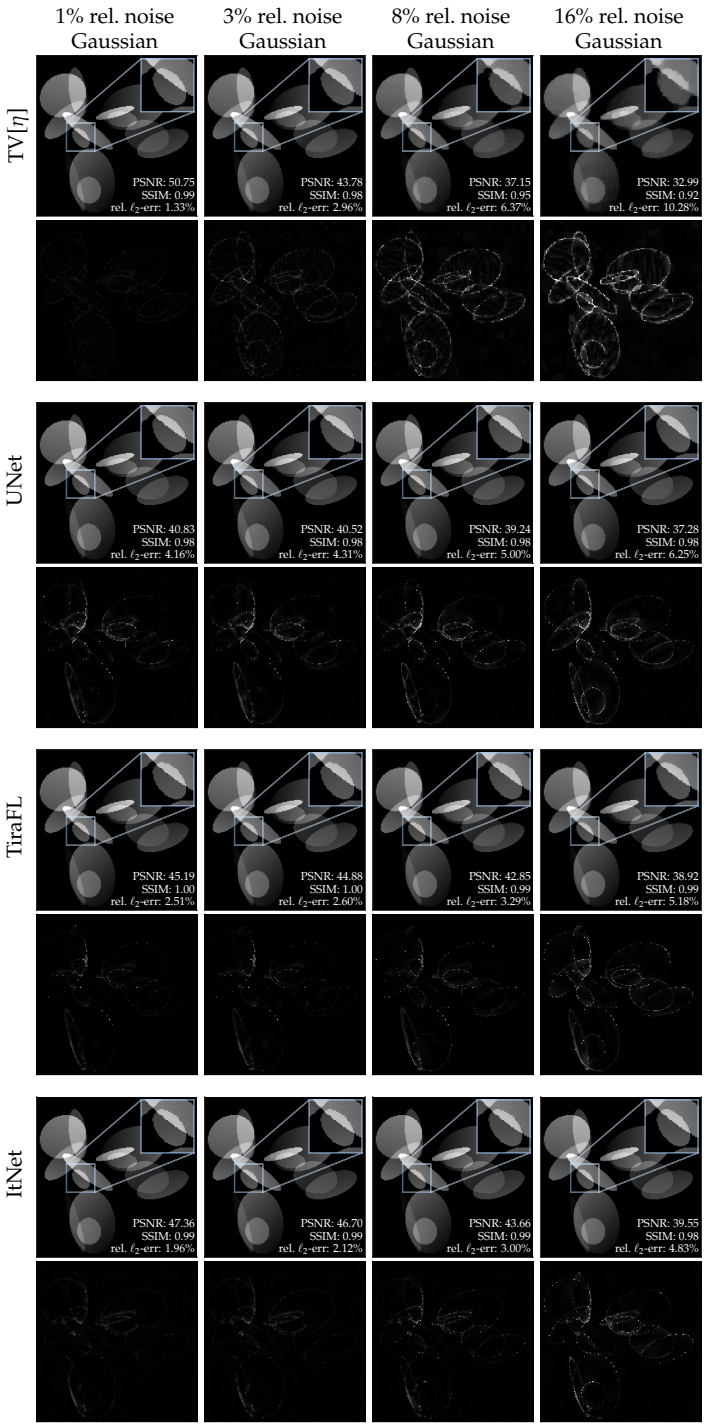


Figure 6.12: **Scenario B1 – Fourier Measurements with Ellipses.** Individual reconstructions of the image from Figure 6.11 under Gaussian noise. The reconstructed images are displayed in the window $[0, 0.9]$, also used for the computation of the PSNR and SSIM. The error plots shown below each reconstruction are displayed in the window $[0, 0.15]$. In favor of the more insightful noise level 16%, we omit the noiseless case.

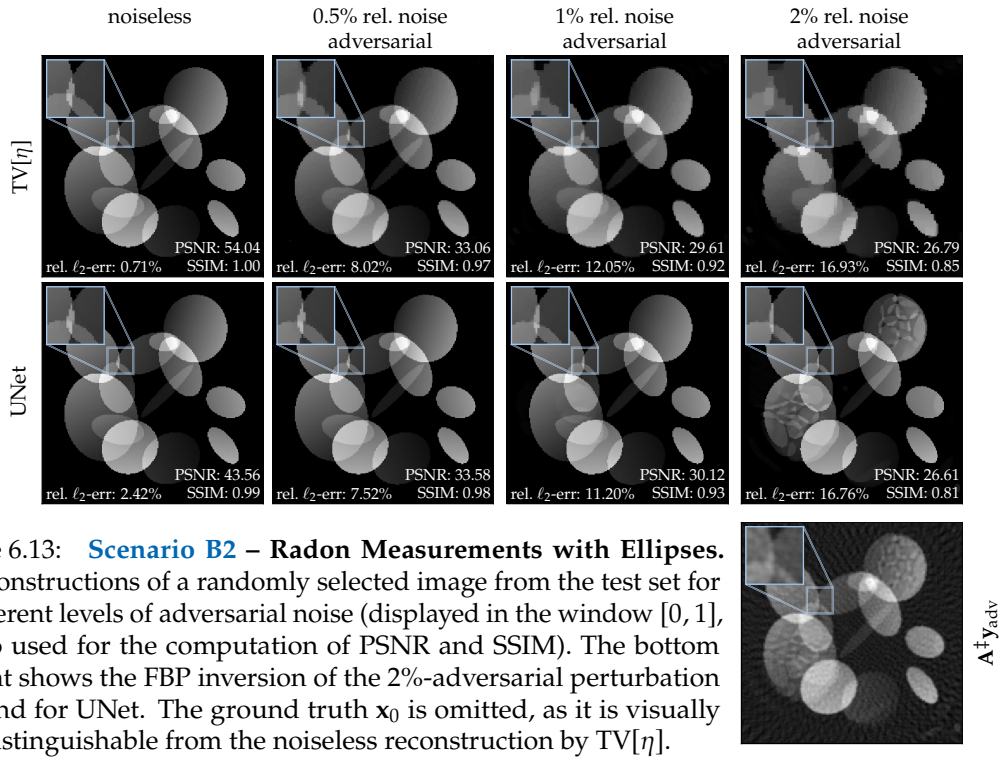


Figure 6.13: **Scenario B2 – Radon Measurements with Ellipses.** Reconstructions of a randomly selected image from the test set for different levels of adversarial noise (displayed in the window $[0, 1]$, also used for the computation of PSNR and SSIM). The bottom right shows the FBP inversion of the 2%-adversarial perturbation found for UNet. The ground truth $\mathbf{x}_0$ is omitted, as it is visually indistinguishable from the noiseless reconstruction by TV[η].

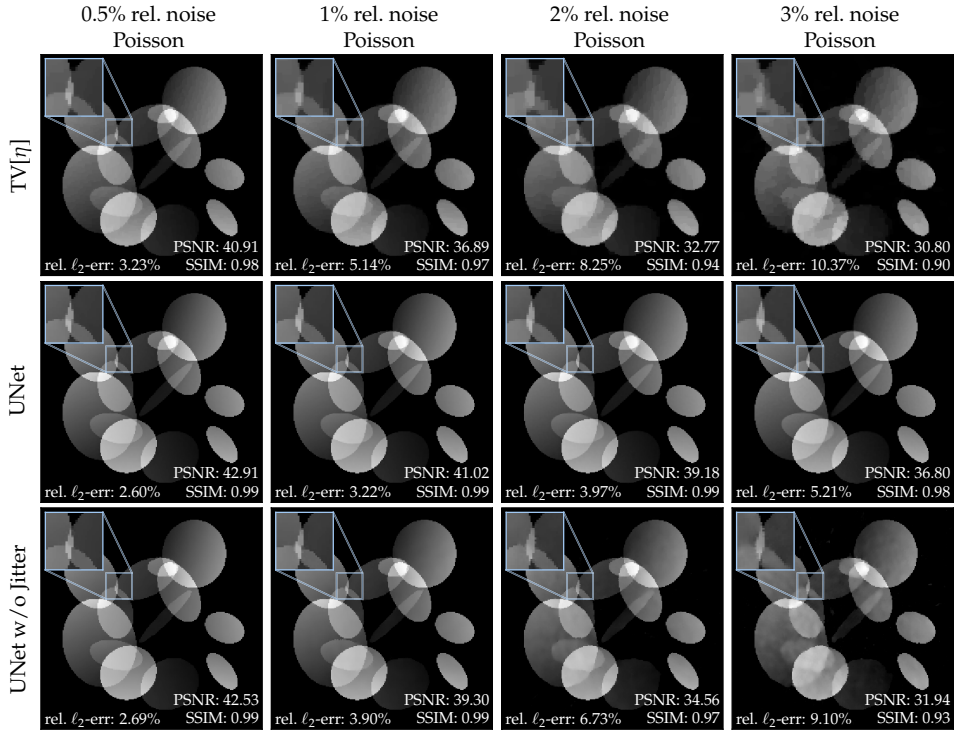


Figure 6.14: **Scenario B2 – Radon Measurements with Ellipses.** Reconstructions of the image from Figure 6.13 under Poisson noise (displayed in the window $[0, 1]$, also used for the computation of PSNR and SSIM). In favor of the more insightful noise level 3%, we omit the noiseless case. The bottom row shows reconstructions for a UNet trained without jittering; see also Section 6.3.1.

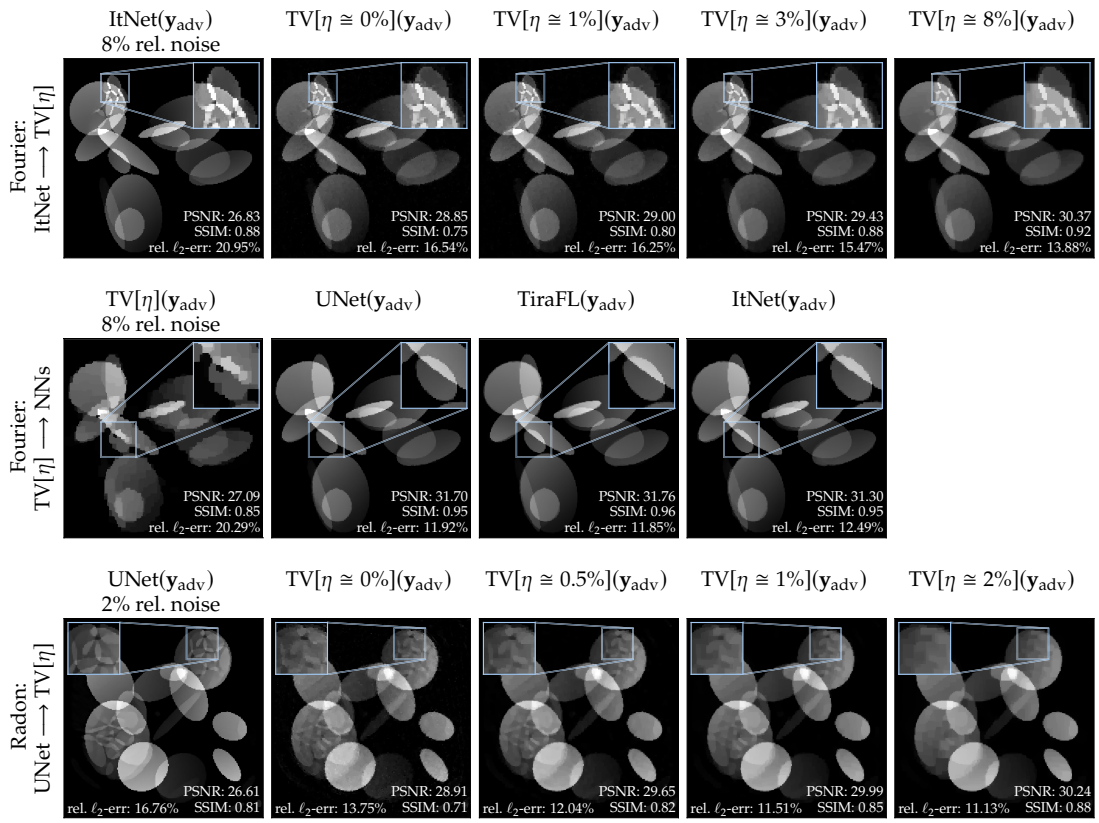


Figure 6.15: **Case Study B – Transferability of Perturbations.** This figure analyzes how adversarial noise transfers between TV minimization and neural-network-based solvers. The top row shows the recovery behavior of TV[η] in the case of Fourier measurements when an adversarial perturbation $\mathbf{y}_{\text{adv}}$ found for ItNet is used as input (cf. Figure 6.11). Here, we also demonstrate the impact of the noise tuning parameter η, which controls the degree of regularization for TV minimization. The middle row presents the reverse experiment: an adversarial perturbation $\mathbf{y}_{\text{adv}}$ found for TV[η] is plugged into each considered neural network. The bottom row is the analog of the top row in the case of Radon measurements (cf. Figure 6.13).

Conclusion

The main findings of Case Study A remain valid: (i) The adversarial robustness of neural-network-based methods and TV minimization is similar with respect to the ℓ_2-error. (ii) Neural networks are more resilient against statistical perturbations in mid- to high-noise regimes. (iii) There is a clear gap between adversarial and statistical noise that is comparable for model-based and learned schemes.

The individual reconstruction results in Figures 6.11 and 6.13 allow for further insights. First, the effect of adversarial noise for TV[η] manifests itself in the well-known staircasing phenomenon, a considerable loss of resolution as well as point-like artifacts (see the zoomed region in Figure 6.11). In contrast, neural-network-based methods always produce sharp images, with almost imperceptible visual errors up to 3% relative noise in the case of Fourier measurements (1% noise in the case of Radon measurements). For the highest noise level, on the other hand, they exhibit unnatural ellipsoidal artifacts.

At first sight, this observation might indicate a vulnerability to adversarial noise.

However, a simple *transferability test* refutes this conclusion (cf. [PMG16]): plugging the perturbed measurements for ItNet into TV[η] leads to the same ellipsoidal artifacts, see Figure 6.15. Furthermore, Figure 6.13 reveals that the corresponding artifacts are already present in the FBP inversion and are not caused by the post-processing network. This shows that the learned solvers do not suffer from undesired instabilities, but the observed artifacts are due to actual features in the corrupted measurements. Interestingly, adversarial perturbations found for TV[η] do not transfer to neural-network-based methods, see Figure 6.15. Overall, the attack strategy of (6.3) has different qualitative effects on each reconstruction paradigm: while known flaws of TV minimization are amplified, the neural networks are perturbed by adding "real" ellipsoidal features to the measurements.

On a final note, we confirm the ranking of architectures as pointed out in Case Study A. Nevertheless, there is no clear superiority of the fully learned schemes as in case of Gaussian measurements, since the inverse Fourier transform appears to be a near-optimal choice of a model-based inversion layer.

6.2.3 Case Study C: MRI on Real-World Data (fastMRI)

The third case study of this chapter is devoted to a real-world MRI scenario. To this end, we use the publicly available *fastMRI* knee dataset, which consists of 1594 multi-coil diagnostic knee MRI scans.[6] Our experiments are based on the subset of 796 coronal proton-density weighted scans without fat-suppression, resulting in $M \approx 17$k training images. We draw magnitude images $\mathbf{x}_0 \in \mathbb{R}^{320\times320}$, obtained from fully-sampled multi-coil[7] data, and consider subsampled Fourier measurements as in Scenario B1 with 50 radial lines ($m = 17178$ and $n = 320^2 = 102400$). As before, the data processing is complex-valued, while the actual reconstructions are computed as real-valued magnitude images. The model-based and learned inversion layers are realized as in Scenario B1. As common in the fastMRI challenge, we have trained all networks with a cost function based on a combination of the ℓ_1- and SSIM-distance, see also [Zha+17]. TV minimization is solved in the unconstrained formulation, with the regularization parameter determined by a grid search over a subset of the validation set.

Figure 6.16 shows the noise-to-error curves; see also Tables D.7 and D.8 in Appendix D. The associated individual reconstructions with adversarial and Gaussian noise are displayed in Figure 6.17 and Figure 6.18 respectively.

[6] Data used in the preparation of this work were obtained from the NYU fastMRI Initiative database [Zbo+18; Kno+20b] (`https://fastmri.med.nyu.edu`). As such, NYU fastMRI investigators provided data but did not participate in analysis or writing of this thesis. The primary goal of fastMRI is to test whether machine learning can aid in the reconstruction of medical images.

[7] Note that our measurement model actually corresponds to the simpler modality of subsampled single-coil MRI. While the fastMRI challenge also provides single-coil data, it is based on retrospective masking of *emulated* Fourier measurements. The subsampling is done by omitting k-space lines in the phase-encoding direction, which we found less suitable for our robustness analysis; see Section 6.3.4 for an experiment with the original setup. Since emulating single-coil measurements is unavoidable, we have decided to sample from the multi-coil magnitude reconstructions in favor of higher image quality. This was found to be particularly important to ensure that TV minimization can serve as a competitive benchmark method, at least for noiseless measurements.

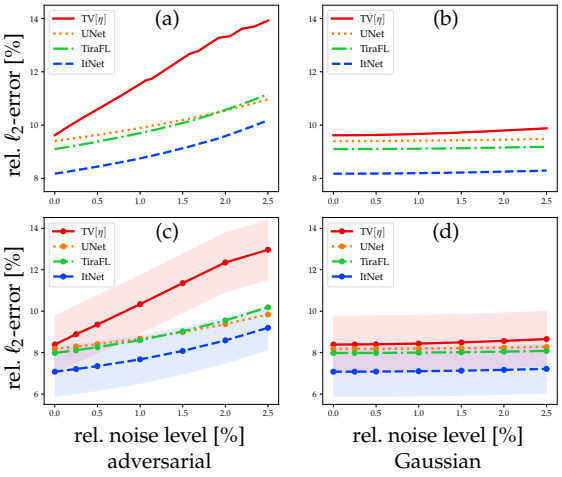


Figure 6.16: **Case Study C – fastMRI.** (a) shows the adversarial noise-to-error curve for the randomly selected image of Figure 6.17. (b) shows the corresponding Gaussian noise-to-error curve, where the mean and (almost imperceptible) standard deviation are computed over 50 draws of $\mathbf{e}$. (c) and (d) display the respective curves averaged over 30 images from the validation set. For the sake of clarity, we omit the standard deviations for UNet and TiraFL, which behave similarly.

Conclusion

Our experimental results show that the main findings of Case Study A and B carry over to real-world data. The noise-to-error curves in Figure 6.16 reveal a superior robustness of the learned reconstruction schemes over TV minimization, even for noiseless measurements (cf. Scenario A2). Figure 6.17 underpins this observation from a qualitative viewpoint: the model-based prior of TV[η] tends to blur fine details in the reconstructed images—this "oil painting" effect becomes stronger with larger perturbations. In contrast, the neural-network-based reconstructions always yield high resolution images. Despite adversarial noise, the central image region—which is of main medical interest—remains largely unaffected, whereas tiny vessel structures appear in the outside (fat) region. Such an amplification of existing patterns is comparable to the ellipsoidal artifacts in Case Study B. We emphasize that this phenomenon only occurs for large adversarial perturbations, where the benchmark of TV minimization already suffers from severe distortions. In particular, the performance of the learned methods is not impaired by the same amount of Gaussian noise (see Figure 6.18).

6.3 Further Aspects of Robustness

This section presents several additional experiments that allow for further insights into the robustness of learned methods.

6.3.1 Training Without Noise – An Inverse Crime?

In this section, the importance of *jittering* for the stability of deep-learning-based reconstruction schemes is discussed (see Section 6.1.2). We have found that this technique can be beneficial for promoting adversarial robustness, in particular, for iterative architectures.

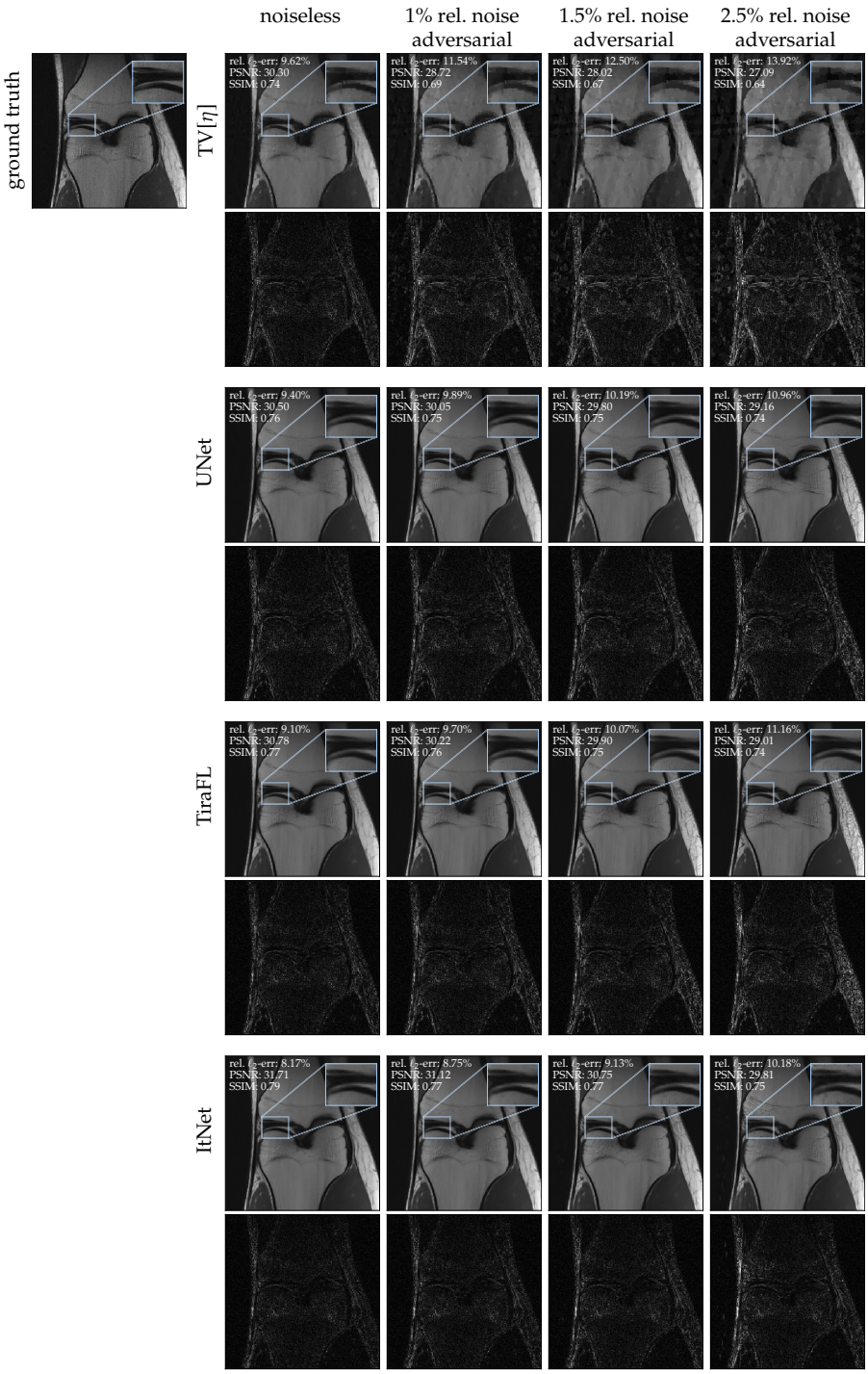


Figure 6.17: **Case Study C – fastMRI.** Individual reconstructions of a central slice of a randomly selected volume from the validation set for different levels of adversarial noise. The reconstructed images are displayed in the window $[0.05, 4.50]$, also used for the computation of the PSNR and SSIM. The error plots shown below each reconstruction are displayed in the window $[0, 1.25]$. The ground truth image $\mathbf{x}_0$ is shown at the top left.

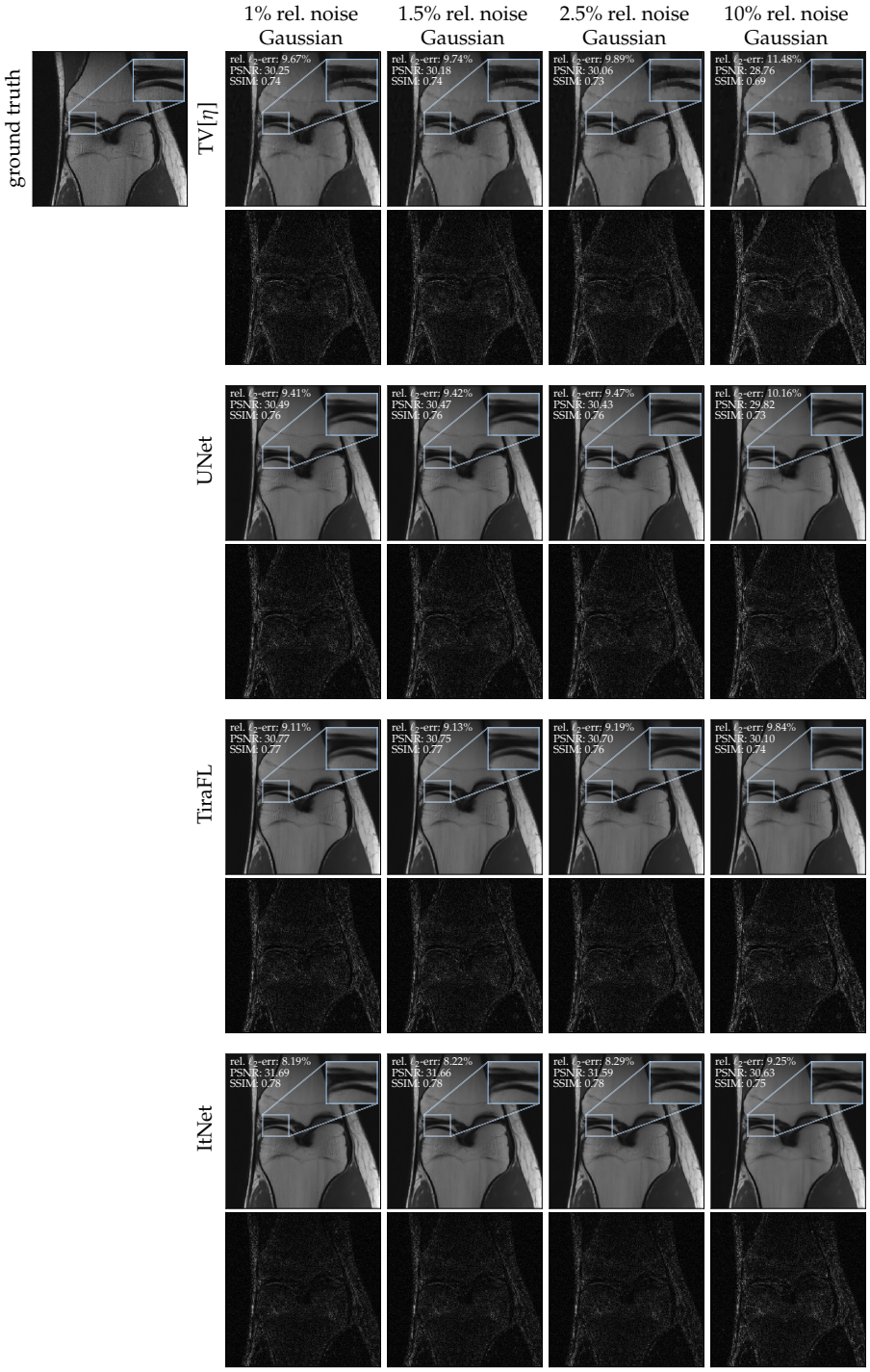


Figure 6.18: **Case Study C – fastMRI.** Individual reconstructions of the image from Figure 6.17 under Gaussian noise. The reconstructed images are displayed in the window $[0.05, 4.50]$, also used for the computation of the PSNR and SSIM. The error plots shown below each reconstruction are displayed in the window $[0, 1.25]$. In favor of the more insightful noise level 10%, we omit the noiseless case.

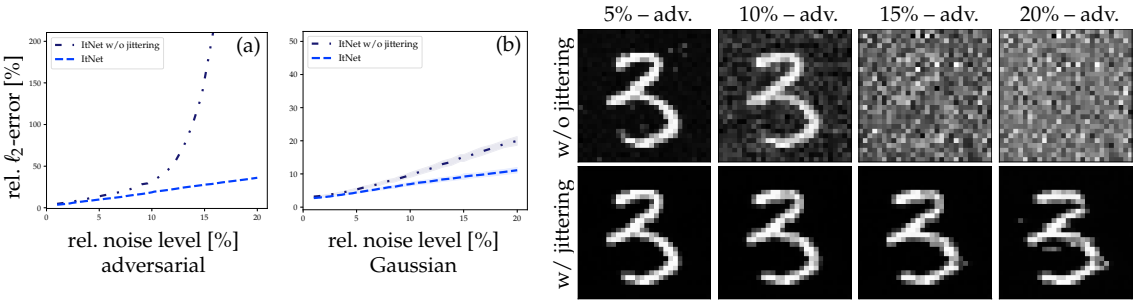


Figure 6.19: **An Inverse Crime?** A comparison between ItNet trained with and without jittering. The noise-to-error curves on the left are generated for the MNIST-digit three from Figure 6.7 with (a) adversarial and (b) Gaussian noise. Individual reconstructions for adversarial noise are shown on the right (the intermediate steps performed by ItNet are visualized in Figure 6.20).

The previous claim is verified by an ablation study, comparing two versions of ItNet for Scenario A2, one trained with jittering and the other without. The resulting noise-to-error curves in Figure 6.19 reveal that noiseless training data can have drastic consequences. Indeed, the relative recovery error blows up at ~15% adversarial noise if jittering is not used. In a similar experiment, we analyze the adversarial robustness of image recovery from Radon measurements as in Scenario B2. The results of Figure 6.21 show a clear superiority of the UNet that was subjected to noise during training (see also Figure 6.14 for the effect of Poisson noise). Without jittering, almost imperceptible distortions in the FBP inversions are intensified by the post-processing network (see blue arrows).

The above observations can be related to the notion of *inverse crimes* in the literature on inverse problems, e.g., see [KS06; MS12]. This term is commonly used to explain the phenomenon of exact, but highly unstable, recovery from noiseless, simulated measurements. In a similar way, networks seem to learn accurate, but unstable, reconstruction rules if they are trained with noiseless data. We note that this does not only concern simulated phantom data but also real-world scenarios. Indeed, in medical imaging applications, one often acquires fully sampled (noisy) reference scans $\{\tilde{\mathbf{y}}^i\}_{i=1}^M$, which are used to generate the ground truth training images $\mathbf{x}_0^i = \mathbf{A}_{\mathtt{full}}^{-1}\tilde{\mathbf{y}}^i$. The measurements are usually subsampled retrospectively by $\mathbf{y}^i = \mathbf{P}\tilde{\mathbf{y}}^i$, where $\mathbf{P}$ denotes an appropriate selection operator. Neural-network-based solution methods for the limited data problem (IP) with $\mathbf{A} = \mathbf{P}\mathbf{A}_{\mathtt{full}}$ are then obtained by training on $\{(\mathbf{y}^i, \mathbf{x}_0^i)\}_{i=1}^M$. Importantly, such data pairs also "commit" an inverse crime, since they follow the noiseless forward model $\mathbf{A}\mathbf{x}_0^i = \mathbf{P}\mathbf{A}_{\mathtt{full}}\mathbf{x}_0^i = \mathbf{y}^i$. Hence, we believe that simulating additional noise might be helpful in the situation of real-world measurements as well. Jittering is a simple and natural remedy in that regard that can additionally reduce overfitting [SD91]. The exploration of further regularization techniques or more sophisticated ways of injecting noise during training is left to future research.

6.3.2 Training With Noise – A Loss of Accuracy?

One might wonder whether the aforementioned robustification via jittering has a detrimental effect on the resulting reconstruction scheme for unperturbed inputs. Indeed, if a method—not necessarily learned—is too insensitive to small changes in the input, it

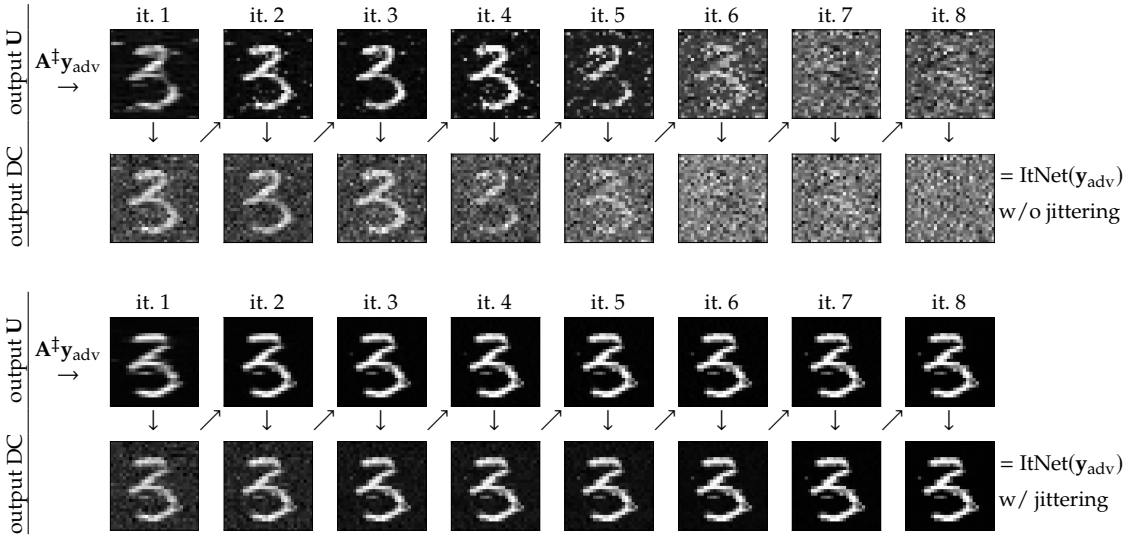


Figure 6.20: **An Inverse Crime?** Intermediate steps performed by ItNet with and without jittering. The 20%-adversarial perturbations correspond to the individual reconstructions shown in Figure 6.19.

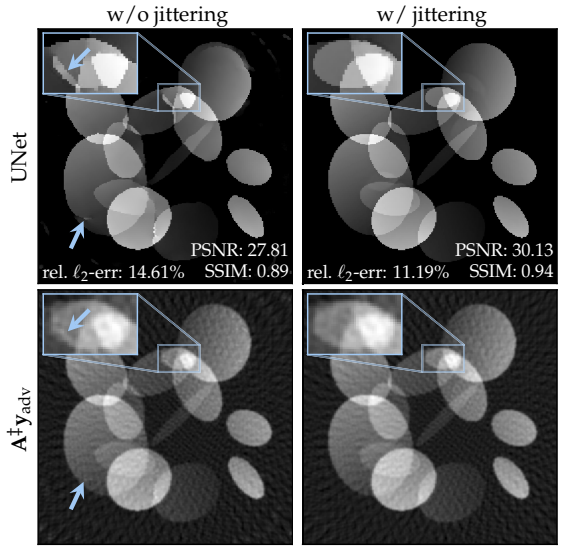


Figure 6.21: **An Inverse Crime?** A comparison between UNet trained with and without jittering for image recovery from sparse-angle Radon measurements, see also Figure 6.13. The reconstructions are obtained for 1% adversarial noise. The bottom figures show the FBP inversions of the found perturbations, respectively. The blue arrows highlight tiny distortions that are amplified by the post-processing network.

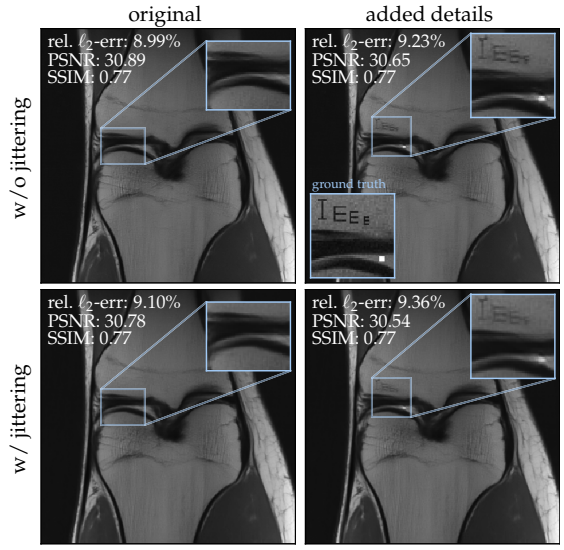


Figure 6.22: **Losing Accuracy?** The left column compares the image of Figure 6.17, when reconstructed by TiraFL with and without jittering, respectively. The right column displays analogous results after adding two out-of-distribution features (text and a 3×3-square) to the ground truth; note that the smallest letter 'E' has the lowest possible resolution. The reconstructed images are displayed in the window $[0.05, 4.50]$, also used for the computation of the PSNR and SSIM.

might become incapable of reconstructing fine details. In the context of our study, it is useful to distinguish between the recovery accuracy with respect to *in-distribution* and *out-of-distribution* (*OOD*) features. The former simply corresponds to a task evaluation on regular images from the test set. Regarding this aspect, we observe only a marginal impact of jittering: across all considered scenarios, no significant performance loss was found when training with noise (e.g., see left column of Figure 6.22), and occasionally, the accuracy even improved slightly (e.g., see Figure 6.19 and left column of Figure 6.14).

The behavior might be different for OOD attributes. Following [Ant+20], we address this situation by exposing a neural-network-based solver to structural details that do not belong to the data distribution. Figure 6.22 shows that inserted text and a 3×3-square are recognizable with and without jittering. While the feature contrast is higher in the latter case, no essential information was missed due to training with noise.

Nevertheless, the OOD generalization of learned methods is still poorly understood in general. Among other factors, the outcome may also depend on the "richness" of the training data; see [Ant+20, Fig. 5] and [ACH21, Fig. 5] for similar experiments as in Figure 6.22 with phantom ellipses (Case Study B). However, we anticipate that a potential loss of OOD performance is a consequence of our naive (non-adaptive) jittering strategy, rather than an intrinsic trade-off between accuracy and stability. Although beyond the scope of this work, an obvious improvement would be a more advanced approach, for instance, by learning noise-aware neural networks for multiple values of η. This is comparable to tuning TV[η] with respect to the given noise level.

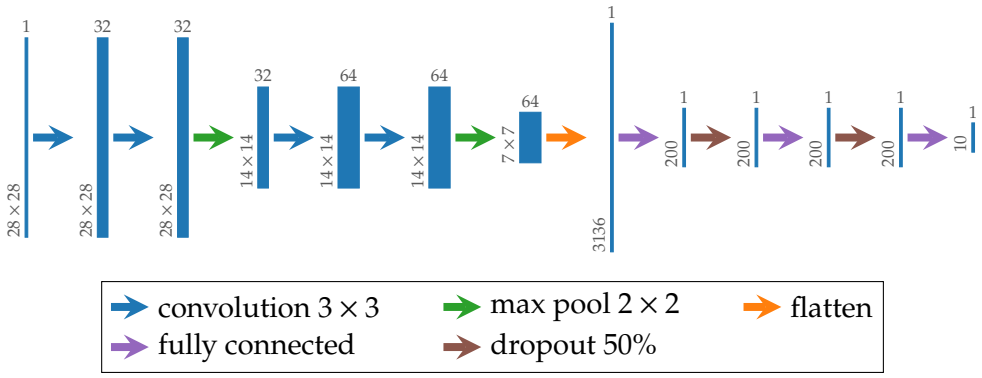


Figure 6.23: **Classification Network.** Illustration of the convolutional neural network architecture for classification from compressed measurements. All convolutions and the first two fully connected layers are followed by ReLU activations. The final layer is followed by a softmax activation.

6.3.3 Adversarial Examples for Classification From Compressed Measurements

In medical healthcare, image recovery is merely one component of the entire data-processing chain. Indeed, machine learning techniques are particularly suitable for automated diagnosis or personalized treatment recommendations. As argued in the introduction of this chapter, the study of adversarial examples for such classification tasks differs from the robustness analysis of reconstruction methods. In this section, we shed further light on this subject by analyzing classification from compressed measurements—think of detecting a tumor from a subsampled MRI scan.

To this end, we revisit the benchmark model of Scenario A2, with the goal to predict MNIST digits from their Gaussian measurements. This is realized by training a basic convolutional neural network classifier ConvNet: $\mathbb{R}^n \to [0,1]^{10}$, mapping images to class probabilities for each of the ten digits, see Figure 6.23. The concatenation with a reconstruction method Rec: $\mathbb{R}^m \to \mathbb{R}^n$ then yields the following classification map:

$$\mathrm{CC}\colon \mathbb{R}^m \to [0,1]^{10},\ \mathbf{y} \mapsto [\mathrm{ConvNet} \circ \mathrm{Rec}](\mathbf{y}). \tag{6.4}$$

The approach of CC can be seen as a simplified model for the automated diagnosis from subsampled measurements; see also [BMR17] and the references therein for the related problem of *compressed classification*. Inspired by [CW17], we adapt the attack strategy (6.3) to the classification setting by (approximately) solving

$$\mathbf{e}_{\mathrm{adv}} = \operatorname*{argmax}_{\|\mathbf{e}\|_2 \le \eta} \max_{k \neq c} [\mathrm{CC}(\mathbf{y}_0 + \mathbf{e})]_k - [\mathrm{CC}(\mathbf{y}_0 + \mathbf{e})]_c$$

where $c \in \{0, 1, \dots, 9\}$ is the true class label of $\mathbf{x}_0$. Figure 6.24 shows a noise-to-*accuracy* curve visualizing the relative amount of correct classifications for different choices of Rec on the left. The corresponding image reconstructions $\mathrm{Rec}(\mathbf{y}_0 + \mathbf{e}_{\mathrm{adv}})$ as well as the predicted classes $\operatorname{argmax}_k [\mathrm{CC}(\mathbf{y}_0 + \mathbf{e}_{\mathrm{adv}})]_k$ for an example digit are presented on the right.

All classifiers exhibit a transition behavior: the success rate is almost perfect for small perturbations and then drops to zero at some point. The associated images show that we have found adversarial examples in the ordinary sense of machine learning. Indeed, every visualized reconstruction is still recognizable as the digit nine. In other words, although

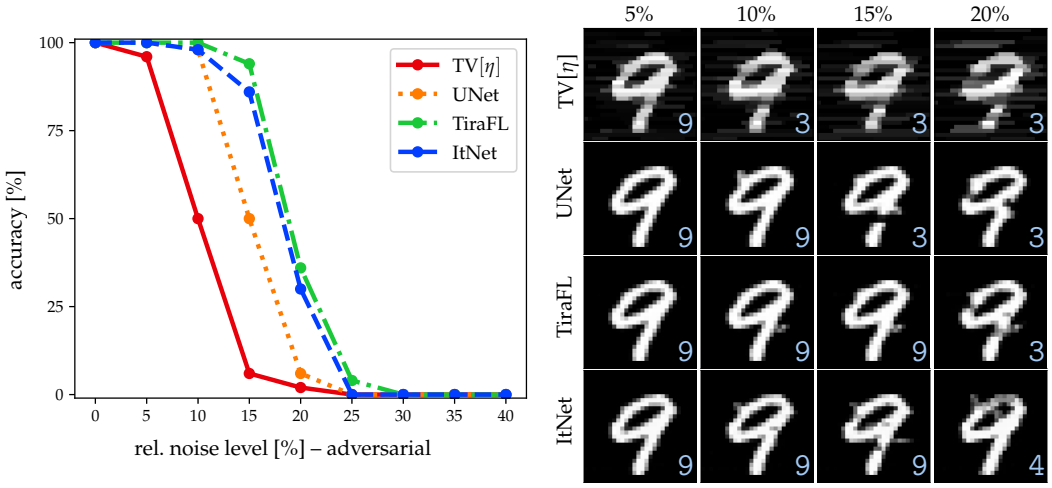


Figure 6.24: **Classification from Compressed Measurements.** The left curve plots the relative adversarial noise level against the prediction accuracy of the classifier (6.4) for different recovery methods (averaged over 50 digits from the test set). The intermediate reconstructions of a randomly selected digit are shown to the right for different noise levels. Their predicted class labels are displayed in the bottom right corner.

being stable, each of the recovery methods is capable of producing slightly perturbed images that fool the ConvNet-part. Remarkably, this phenomenon occurs independently of using a model-based or learned solver for (IP). We conclude that deep-learning-based data-processing pipelines (as in medical healthcare) remain vulnerable to adversarial attacks, even if provably robust reconstruction schemes are employed.

6.3.4 The Original fastMRI Challenge Setup

This section demonstrates that the original fastMRI challenge data for single-coil MRI is more susceptible to adversarial noise. In contrast to Case Study C, the challenge measurement setup is based on omitting k-space lines in the phase-encoding direction (corresponding to 4-fold acceleration), i.e., the subsampling mask is defined by vertical lines. The resulting undersampling ratio of ~23% is higher than in Case Study C (~17%). Figure 6.25 shows individual image reconstructions for TV[η] and Tira.[8] Compared to Figure 6.17, the outcomes indicate a loss of adversarial robustness, as the reconstructed images exhibit undesired line-shaped artifacts (see blue box in Figure 6.25). This phenomenon occurs regardless of using a model-based (TV[η]) or learned method (Tira). In fact, the observed artifacts are a consequence of the underlying measurement system: the anisotropic mask pattern implies that vertical image features become more "aligned" with the kernel of the forward operator. Hence, clearly visible distortions may be caused by relatively small perturbations of the measurements (cf. [Got+20]). This confirms that the design of sampling patterns does not only influence the accuracy of a reconstruction method (e.g., see [Boy+16]), but also its adversarial robustness.

[8] Since the fastMRI challenge setup does not rely on a fixed subsampling mask, the fully-learned approach for Tiramisu is not available here. Our Tira-net performs competitively in the fastMRI public leaderboard: We achieve an SSIM of 0.765, whereas the leading method has 0.783 (`https://fastmri.org/leaderboards/`, teamname `AnItalianDessert`, accessed on 2020-11-08).

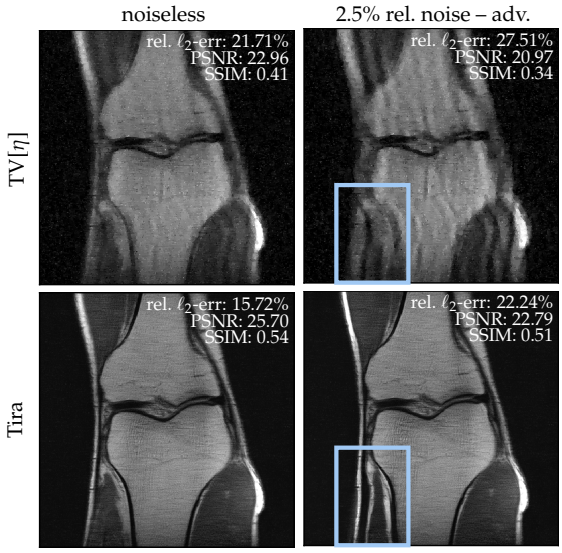


Figure 6.25: **The Original fastMRI Challenge Setup.** Reconstructions of a randomly selected image from the validation set. Compared to the analogous experiment in Figure 6.17, the Fourier subsampling operator is based on vertical lines in the k-space instead of a radial mask. The reconstructed images are displayed in the window $[0.05, 4.50]$, also used for the computation of the PSNR and SSIM. Note that the data are given as emulated single-coil (ESC) measurements, whereas the reconstructions in Figure 6.17 are based on multi-coil images. Hence, the signal-to-noise ratios are not directly comparable.

6.4 Discussion

There are several aspects that go beyond the scope of our study: (i) We are restricted to a selection of end-to-end neural network architectures, excluding other approaches, such as generative models [Bor+17], the deep image prior [UVL18], or learned regularizers [Li+20]. However, since these algorithms typically involve more model-based components, we expect their robustness to be comparable to the schemes considered in the present work. (ii) Due to the non-convexity of (6.3), a theoretical optimality certificate for our attack strategy is lacking. Nevertheless, our results provide empirical evidence that we have solved the problem adequately: The gap between worst-case and statistical perturbations appears consistent across all considered scenarios. More importantly, we have verified the ability to detect an error blowup caused by adversarial noise (see Figure 6.19). (iii) Our analysis takes a mathematical perspective on robustness, thereby relying on standard similarity measures, in particular, the Euclidean norm. It is well known that such quantitative metrics are insensitive to several types of visual distortions. For instance, even the winning networks of the 2019 fastMRI challenge were occasionally unable to capture certain tiny pathological features that rarely appear in the data [Kno+20a]. This issue was specifically addressed in the follow-up 2020 fastMRI challenge, which focuses on pathology depiction instead of an overall image quality assessment [Muc+20]. However, this time, hallucinations were noted, i.e., non-physical features created by the reconstruction networks. Nevertheless, in view of our findings, we suspect that both flaws are not due to perturbations of the measurements. An investigation of causes and remedies seems to be a promising direction for future research, e.g., see [Che+20].

6.5 Conclusion

In an extensive series of experiments, this chapter has analyzed the robustness of deep-learning-based solution methods for inverse problems. Central to our approach was a study of the effects of adversarial noise, i.e., worst-case perturbations of the measurements that maximize the reconstruction error. A systematic comparison with a model-based reference method has shown that standard deep neural network schemes are remarkably resilient against statistical and adversarial distortions. On the other hand, we have demonstrated that instabilities might be caused by the "inverse crime" of training with noiseless data. A simple remedy in that regard is jittering—a standard regularization and robustification technique in deep learning [GBC16]. However, it is well known that this does not cure the adversarial vulnerability of deep neural network classifiers, which requires more sophisticated defense strategies [MSG20]. While such defenses may also improve the robustness in the context of image recovery [RBL20], our results allow for a surprising conclusion: Injecting Gaussian random noise in the training phase seems sufficient to obtain solution methods for inverse problems that are resistant to other types of noise, including adversarial perturbations.

The relevance of artificial intelligence for future healthcare is undeniable. Reliable reconstruction methods are indispensable in this field, since errors caused by instabilities can be fatal. In light of the threat of intentional manipulations in medical imaging [Fin+19], it is reassuring to know the limits of what could go wrong in principle. Of similar practical interest is the robustness against random perturbations, which is the standard noise model for common imaging modalities. We believe that our work makes progress in both regards, by showing optimistic results on the use of deep neural networks for inverse problems in imaging.

7

Accuracy of Reconstruction Methods

In the previous chapter we have studied the robustness of deep-learning-based approaches for solving inverse problems of the following form:

$$\left\{\begin{array}{c}\textit{Given a linear forward operator } \mathbf{A} \in \mathbb{R}^{m\times n} \\ \textit{and corrupted measurements } \mathbf{y} = \mathbf{A}\mathbf{x}_0 + \mathbf{e} \\ \textit{with } \|\mathbf{e}\|_2 \leq \eta, \textit{ reconstruct the signal } \mathbf{x}_0.\end{array}\right\} \qquad \text{(IP, restated)}$$

Recall, that the error of any reconstruction method $\mathrm{Rec}\colon \mathbb{R}^m \to \mathbb{R}^n$ for (IP) can be decomposed as

$$\|\mathbf{x}_0 - \mathrm{Rec}(\mathbf{y})\|_2 \leq \underbrace{\|\mathbf{x}_0 - \mathrm{Rec}(\mathbf{A}\mathbf{x}_0)\|_2}_{(a)} + \underbrace{\|\mathrm{Rec}(\mathbf{A}\mathbf{x}_0) - \mathrm{Rec}(\mathbf{y})\|_2}_{(b)}, \qquad \text{(1.1, restated)}$$

with an *accuracy* term (a) describing the error in the noiseless limit $\eta \to 0$ and a *robustness* term (b) describing the dependence of the error on the strength of the measurement noise. It goes without saying that besides robustness also the ability to accurately solve an inverse problem is indispensable for the reliable use of a reconstruction method in practice.

Critical voices have recently been heard regarding this aspect of neural-network-based schemes. For instance, Sidky et al. have demonstrated that *post-processing* by the prominent U-Net architecture may not yield satisfactory recovery precision in a sparse-view computed tomography (CT) scenario [Sid+21b]. These findings have led to the "AAPM Deep Learning Sparse-View Computed Tomography" grand challenge with the goal of identifying „[. . .] the state-of-the-art in solving the CT inverse problem with data-driven techniques" [Sid+21a]. Here, the expression "solving" is used to describe algorithms that provide perfect recovery in the idealistic situation of noiseless measurements. Thus, the overall vision of the AAPM challenge was to evaluate whether deep-learning-based schemes can achieve (near-)exact precision, similarly to classical benchmark methods such as total-variation (TV) minimization, cf. [Sid+21b].

In Chapter 6 we have considered scenarios that are similar to the sparse-view CT setup of [Sid+21b] in the following sense: Exact signal recovery by TV minimization is possible for noisefree measurements. In such situations, we were also able to train neural networks that provide visually perfect reconstructions. However, it should be noted that Chapter 6 was mainly concerned with noisy measurements and the neural network schemes did not achieve perfect accuracy in all scenarios (cf. left sides of the noise-to-error curves in Figures 6.4, 6.9, 6.10 and 6.16). We took this as a motivation to participate in the AAPM challenge with the goal of improving the accuracy of our iterative neural network scheme from Chapter 6 and design a data-driven recovery workflow for (near-)exact image recovery. Our approach is rooted in the following hypothesis:

> High reconstruction accuracy is possible when the forward model is explicitly incorporated into the reconstruction mapping, e.g., by an iterative promotion of data-consistent solutions.

Highlighting the importance of incorporating the forward operator is by no means novel and we have seen some of the benefits already in the previous chapter. It is one of the central pillars of scientific machine learning, where neural networks are frequently enriched (or constrained) by physical modeling. Indeed, the seminal works on deep-learning-based solution strategies for inverse problems are inspired by unrolling classical algorithms [GL10; Yan+16; Ham+18; AMJ18; AÖ18; Che+18; Sch+19; Ham+19; Chu+20; Hea+21; GOW21]. At the present time, most state-of-the-art methods are based on iterative end-to-end networks and related schemes, e.g., see [Kno+20a; Muc+20; Leu+21] for other recent competition benchmarks.

Our approach to the AAPM challenge is no exception in that respect. We rely on a conceptually simple, yet powerful deep learning workflow, which turns a post-processing U-Net [RFB15] into an iterative reconstruction scheme. In fact, it is a slight modification of the iterative network from Chapter 6. Recall, that from a technical perspective, most of its design components have been previously reported in the literature. However, the overall strategy appears to be novel and differs from more common unrolled networks in several aspects, including:

(a) We make use of a *pre-trained* U-Net as the computational backbone.

(b) Data-consistency is inspired by an ℓ^2-gradient step, but utilizes the *filtered* backprojection (FBP) instead of the regular adjoint (this is different from Chapter 6, where the adjoint was used).

In line with many previous works, our iterative network involves only very few (five) evaluations of the forward operator (respectively six evaluations of the filtered backprojection). However, we are the first to show evidence that this is sufficient to match the precision of model-based solvers such as TV minimization, where the forward model often forms a computational bottleneck due to hundreds or thousands of iterations before convergence.

A distinctive aspect of the AAPM challenge is, that the exact forward model is not disclosed to the participants. It is only known that tomographic fanbeam measurements were taken and tuples $(\mathbf{x}_0, \mathbf{y}_0)$ of ground truth image signals and their corresponding noiseless measurements are provided. Therefore, we pursue a data-driven estimation of the underlying fanbeam geometry as a first step. This is achieved by fitting a generic, parametrized fanbeam operator to the provided sinogram-image pairs in a deep-learning-like fashion (i.e., by gradient descent with backpropagation/automatic differentiation). We hope that this approach can be of further use in the context of geometric calibration and forward operator correction.

Overall, we were able to affirmatively answer the question raised by the AAPM challenge setup: End-to-end neural networks can achieve near-perfect accuracy on the prescribed CT reconstruction task. By matching the precision of widely-accepted classical benchmarks (such as TV minimization) with our learned iterative scheme we were able to win the AAPM challenge. We think that the proposed strategy will be of use for other inverse problems as well, given that it outperformed the runner-up team and other state-of-the-art data-driven approaches, such as the learned primal dual algorithm [AÖ18] by about an

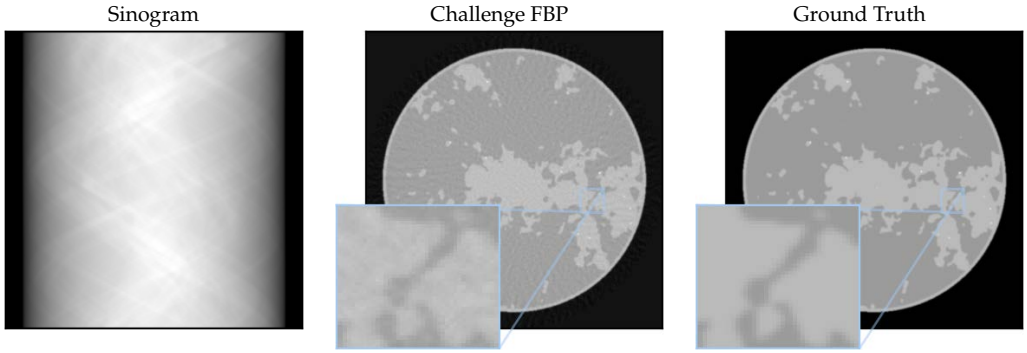


Figure 7.1: **The AAPM Challenge Data.** Example of a 128-view sinogram, corresponding FBP reconstruction, and the ground-truth phantom image taken from the AAPM challenge training dataset.

order of magnitude with respect to the root-mean-square-error (RMSE). Our findings (in particular in combination with the robustness observations of Chapter 6) provide evidence for the possibility of a reliable use of deep-learning-based solutions to inverse problems.

7.1 The AAPM Challenge Setup

The AAPM challenge data is similar to the setting in [Sid+21b], i.e., it is based on synthetic 2D grayscale images of size 512×512 simulating real-world mid-plane breast CT device scans. Four different tissues were modeled: adipose, skin, fibroglandular tissue, and microcalcifications. Gaussian smoothing was applied in order to obtain smooth transitions at the tissue boundaries. A fanbeam geometry with 128 projections over 360 degrees was used to create sinograms and FBPs, see Figure 7.1 for an example. The exact fanbeam geometry was not revealed to the participants. No noise was added to the provided data, neither to the phantom images nor the measurements. The provided training set consisted of 4000 tuples of phantom images, their corresponding sinograms, and FBP reconstructions. A test set of 100 pairs of sinograms and FBPs (without publicly available ground-truth phantoms) was used for the final challenge evaluation. Initially, about 50 international teams participated, out of which 25 submitted their method to the final evaluation. For more details about the challenge setup and results see [Sid+21a; SP22].

7.2 Methods

We now give a more detailed overview of our approach, together with a motivation of its design choices. A specification of all empirically selected hyper-parameters (describing the network architectures and training procedures) can be found in Appendix E (Tables E.1 and E.2).

Step 1 – Data-Driven Geometry Identification

The first step of our reconstruction pipeline estimates the unknown forward operator from the provided training data. The continuous version of *tomographic fanbeam measurements* is

based on computing line integrals

$$p(s,\varphi) = \int_{L(s,\varphi)} x_0(x,y)\,\mathrm{d}(x,y),$$

where x_0 is the unknown image and $L(s,\varphi)$ denotes a line in fanbeam coordinates, i.e., φ is the *fan rotation angle* and s encodes the *sensor position* along the detector array; see [Fes17] for more details. In an idealized[1] situation, the fanbeam model is specified by the following geometric parameters, as illustrated in Figure 7.2(a):

- d_{source} – the distance of the x-ray source to the origin,
- d_{detector} – the distance of the detector array to the origin,
- n_{detector} – the number of detector elements,
- s_{detector} – the spacing of the detector elements along the array,
- n_{angle} – the number of fan rotation angles,
- $\boldsymbol{\varphi} \in [0, 2\pi]^{n_{\text{angle}}}$ – a discrete list of rotation angles.

Here, it is assumed that integrals are only measured along a finite number of lines, determined by $m = n_{\text{detector}} \cdot n_{\text{angle}}$. In the AAPM DL-Sparse-View Challenge, the resulting forward operator is *severely ill-posed*, since only the measurements of a few fan rotation angles n_{angle} are acquired. Furthermore, the geometric setup is not disclosed to the challenge participants. It is only known that fanbeam measurements are taken.

We address this lack of information by a data-driven estimation strategy that fits the above set of parameters to the given training data. To this end, we first observe that the previous parametrization is redundant, and without loss of generality, we may assume that $s_{\text{detector}} = 1$ (by rescaling d_{detector} appropriately). Further, if the field-of-view angle γ is known, then the relation

$$d_{\text{detector}} = \frac{n_{\text{detector}} \cdot s_{\text{detector}}}{2\tan\gamma} - d_{\text{source}} \tag{7.1}$$

can be used to eliminate another parameter, cf. Figure 7.2(b). Thus, the fanbeam geometry is effectively determined by the reduced parameter set $(d_{\text{source}}, n_{\text{detector}}, n_{\text{angle}}, \boldsymbol{\varphi})$. The training data provides pairs of discrete images $\mathbf{x}_0 \in \mathbb{R}^{512\cdot 512=n}$ and its simulated fanbeam measurements $\mathbf{y}_0 \in \mathbb{R}^{128\cdot 1024=m}$, from which the dimensions $n_{\text{angle}} = 128$ and $n_{\text{detector}} = 1024$ can be derived. We determine the field of view as $\gamma = \arcsin(r/d_{\text{source}})$, so that the maximum inscribed circle of radius r in the discrete image is exactly contained within each fan of lines, which is a common choice for fanbeam CT (i.e., the radius corresponds to half of the width or height of the image, thus $r = 256$ for the 512×512 image signals), cf. Figure 7.2(b). Hence, (7.1) leads to

$$d_{\text{detector}} = \frac{n_{\text{detector}} \cdot s_{\text{detector}}}{2 \cdot r} \cdot \sqrt{d_{\text{source}}^2 - r^2} - d_{\text{source}}.$$

[1] We have found that this basic model was enough to accurately describe the AAPM challenge setup. If needed, it would be possible to account for other factors such as non-flat detector arrays, offsets of the axis of rotation from the origin, misalignments of the detector array, etc.

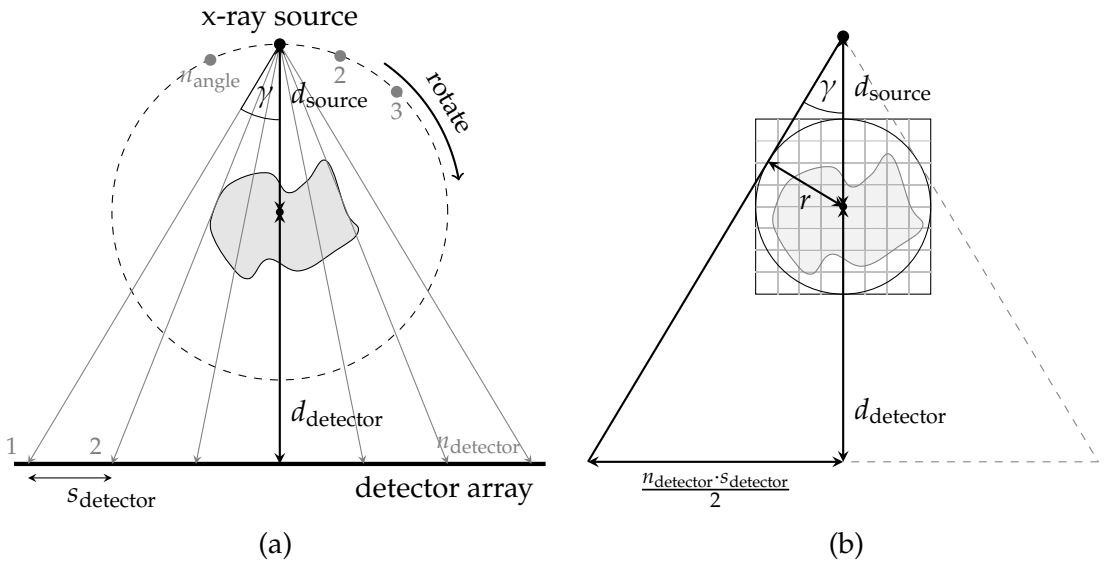


Figure 7.2: **Fanbeam Geometry.** (a) Illustration of the parameters determining the geometry of the fanbeam CT model. (b) The field of view angle γ is chosen to fit the maximum inscribed circle within the discretized image signals. The radius of the circle is equal to half the width and height of the image.

The main difficulty of Step 1 lies in the estimation of the remaining parameters $(d_{\text{source}}, \boldsymbol{\varphi})$. To that end, we have implemented a discrete fanbeam transform from scratch in `PyTorch` (together with its corresponding filtered backprojection). A distinctive aspect of our implementation is the use of a vectorized numerical integration that enables the efficient computation of derivatives with respect to the geometric parameters by means of *automatic differentiation*. This can be exploited for a data-driven parameter identification, for instance, by gradient descent. More precisely, we use a ray-driven numerical integration for the forward model and a pixel-driven and sinogram-reweighting-based filtered backprojection (with a Hamming filter) [Fes17, Sec. 3.9.2]. In addition to the parameters $(d_{\text{source}}, \boldsymbol{\varphi})$, we also introduce learnable scaling factors s_{fwd} and s_{fbp} for the forward and inverse transform, respectively. They account for ambiguities in choosing the discretization units of distance compared to the actual physical units of distance.

We estimate the remaining free parameters $\boldsymbol{\theta}_{\text{fan}} = (s_{\text{fwd}}, d_{\text{source}}, \boldsymbol{\varphi}) \in \mathbb{R}^{130}$ of the implemented forward operator $\mathbf{A}[\boldsymbol{\theta}_{\text{fan}}] \in \mathbb{R}^{m \times n}$ in a deep-learning-like fashion: The ability to compute derivatives $\frac{\mathrm{d}\,\mathbf{A}[\boldsymbol{\theta}_{\text{fan}}]}{\mathrm{d}\boldsymbol{\theta}_{\text{fan}}}$ allows us to make use of the $M = 4000$ sinogram-image pairs $\{(\mathbf{y}_0^i, \mathbf{x}_0^i)\}_{i=1}^M$ by solving

$$\min_{\boldsymbol{\theta}_{\text{fan}}} \frac{1}{M} \sum_{i=1}^{M} \|\mathbf{A}[\boldsymbol{\theta}_{\text{fan}}](\mathbf{x}_0^i) - \mathbf{y}_0^i\|_2^2, \tag{7.2}$$

with a variant of gradient descent (see Remark 7.1 for details). Finally, we determine the last free parameter s_{fbp} of the implemented filtered backprojection operator $\mathbf{A}^{\ddagger}[\boldsymbol{\theta}_{\text{fan}}, s_{\text{fbp}}]$ by solving

$$\min_{s_{\text{fbp}}} \frac{1}{M} \sum_{i=1}^{M} \|\mathbf{x}_0^i - \mathbf{A}^{\ddagger}[\boldsymbol{\theta}_{\text{fan}}, s_{\text{fbp}}](\mathbf{y}_0^i)\|_2^2, \tag{7.3}$$

while keeping the already identified parameters $\boldsymbol{\theta}_{\mathtt{fan}}$ fixed. From now on, we will use the short-hand notation $\mathbf{A}$ and $\mathbf{A}^{\ddagger}$ for the estimated operators $\mathbf{A}[\boldsymbol{\theta}_{\mathtt{fan}}]$ and $\mathbf{A}^{\ddagger}[\boldsymbol{\theta}_{\mathtt{fan}}, s_{\mathtt{fbp}}]$, respectively.

Remark 7.1. The following observations regarding the estimation of the forward and backward operators can be made:

(i) The formulation (7.2) is non-convex and therefore it is not clear whether gradient descent enables an accurate estimation of the underlying fanbeam geometry. Indeed, standard gradient descent was found to be very sensitive to the initialization of $\boldsymbol{\theta}_{\mathtt{fan}}$ and got stuck in bad local minima. To overcome this issue, we solve (7.2) by a *block coordinate descent* instead, which alternatingly optimizes over $s_{\mathtt{fwd}}$, $d_{\mathtt{source}}$, and $\boldsymbol{\varphi}$ with individual learning rates. This strategy was found to effectively account for large deviations of gradient magnitudes of the different parameters. Indeed, we observed a fast convergence and a reliable identification of $\boldsymbol{\theta}_{\mathtt{fan}}$, independently of the initialization.

(ii) In principle, the strategy of (7.2) requires only few training samples to be successful. However, when verifying the robustness of the outlined strategy against measurement noise, we observed that it is beneficial to employ more training data.

(iii) Subsequent to the estimation of an accurate fanbeam geometry, we still noted a systematic error in our forward model. We suspect that it is caused by subtle differences in the numerical integration in comparison to the true forward model of the AAPM challenge. In compensation, we compute the (pixel-wise) mean error over the training set, as an additive correction of the model bias.

Step 2 – Pre-Training a U-Net as Computational Backbone

As in Chapter 6 the centerpiece of our reconstruction scheme is formed by a standard U-Net architecture $\mathbf{U}[\tilde{\boldsymbol{\theta}}]\colon \mathbb{R}^n \to \mathbb{R}^n$ [RFB15]. It is first employed as a residual network to post-process sparse-view filtered backprojection images, i.e., we consider the reconstruction mapping

$$\mathrm{UNet}[\tilde{\boldsymbol{\theta}}]\colon \mathbb{R}^m \to \mathbb{R}^n,\ \mathbf{y} \mapsto \left[\mathbf{U}[\tilde{\boldsymbol{\theta}}] \circ \mathbf{A}^{\ddagger}\right](\mathbf{y}). \tag{7.4}$$

The learnable parameters $\tilde{\boldsymbol{\theta}}$ of the U-Net are trained from the same collection of $M = 4000$ sinogram-image pairs $\{(\mathbf{y}_0^i, \mathbf{x}_0^i)\}_{i=1}^M$ as in Step 1. This is achieved by standard empirical risk minimization, i.e., by (approximately) solving

$$\min_{\tilde{\boldsymbol{\theta}}} \frac{1}{M} \sum_{i=1}^{M} \|\mathbf{x}_0^i - \mathrm{UNet}[\tilde{\boldsymbol{\theta}}](\mathbf{y}_0^i)\|_2^2 + \mu \cdot \|\tilde{\boldsymbol{\theta}}\|_2^2, \tag{7.5}$$

where we choose $\mu = 10^{-3}$. This minimization problem is tackled by 400 epochs of mini-batch stochastic gradient descent and the Adam optimizer [KB14] with initial learning rate 0.0002 and batch size 4.

Remark 7.2. The post-processing strategy of Step 2 was pioneered in [KMY17; Che+17a] and popularized by [Jin+17; Che+17b], among many others. Due to the multi-scale encoder-decoder structure with skip-connections, the U-Net architecture is very efficient in handling image-to-image problems. Therefore, solving (7.5) typically works out-of-the-box without requiring sophisticated initialization or optimization strategies (even in seemingly hopeless situations [HA20]). Making use of a more powerful or a more memory-efficient network would be beneficial, e.g., see results for the Tiramisu network below. However, we preferred to keep our workflow as simple as possible and therefore decided to stick to the standard U-Net as the main computational building block.

Step 3 – Constructing an Iterative Scheme

In this step, we discuss our main reconstruction method. It incorporates the (approximate) forward model $\mathbf{A}$ from Step 1 (and the associated inversion by $\mathbf{A}^\ddagger$) via the following iterative procedure, similar to Chapter 6:

$$\mathrm{ItNet}[\boldsymbol{\theta}]\colon \mathbb{R}^m \to \mathbb{R}^n,\ \mathbf{y} \mapsto \left[\bigcirc_{k=1}^{4} \left(\mathrm{DC}_{\lambda_k,\mathbf{y}} \circ \mathbf{U}[\tilde{\boldsymbol{\theta}}]\right) \circ \mathbf{A}^\ddagger\right](\mathbf{y}), \tag{7.6}$$

with learnable parameters $\boldsymbol{\theta} = [\tilde{\boldsymbol{\theta}}, \lambda_1, \lambda_2, \lambda_3, \lambda_4]$ and the k-th *data-consistency* layer

$$\mathrm{DC}_{\lambda_k,\mathbf{y}}\colon \mathbb{R}^n \to \mathbb{R}^n,\ \mathbf{x} \mapsto \mathbf{x} - \lambda_k \cdot \mathbf{A}^\ddagger(\mathbf{A}\,\mathbf{x} - \mathbf{y}). \tag{7.7}$$

The main differences to the iterative approach in Chapter 6 are the use of the estimated instead of the exact forward model and replacing the adjoint $\mathbf{A}^\top$ by the filtered back-projection $\mathbf{A}^\ddagger$ in the data consistency layers (see below). ItNet is trained by empirical risk minimization analogously to (7.5) with $\mu = 10^{-4}$. We run 500 epochs of mini-batch stochastic gradient descent and Adam with an initial learning rate of $8 \cdot 10^{-5}$ and a batch size of 2 (restarting Adam after 250 epochs). The U-Net parameters $\tilde{\boldsymbol{\theta}}$ are initialized by the weights obtained in Step 2.

In the following, we will briefly discuss central aspects of the architecture in (7.6) and motivate some of the important design choices:

(i) The computational centerpiece of ItNet is formed by the U-Net architecture. This stands in contrast to earlier generations of unrolled iterative schemes, which rely on basic convolutional blocks instead, e.g., see [AÖ18; Yan+16]. We have found that it is advantageous to exploit the efficacy of U-Net-like image-to-image networks as central image-enhancement blocks. This is in line with recent state-of-the-art architectures, which also make use of various advanced sub-networks, e.g., see [Kno+20a; Muc+20; Ham+19; RCS20; Sri+20]. The same U-Net is used in all four iterations (weight sharing), cf. [AMJ18; Ham+19].

(ii) We have observed that it is crucial to initialize the U-Net parameters $\tilde{\boldsymbol{\theta}}$ by the post-processing weights from Step 2. This does not only increase the speed of convergence, but it also significantly improves the final accuracy (see Figure 7.3). In other words, our results show that the initialization of the U-Net block as a post-processing unit makes it possible to find better local minima. To the best of our knowledge, such an effect has not been reported in the literature yet. We emphasize

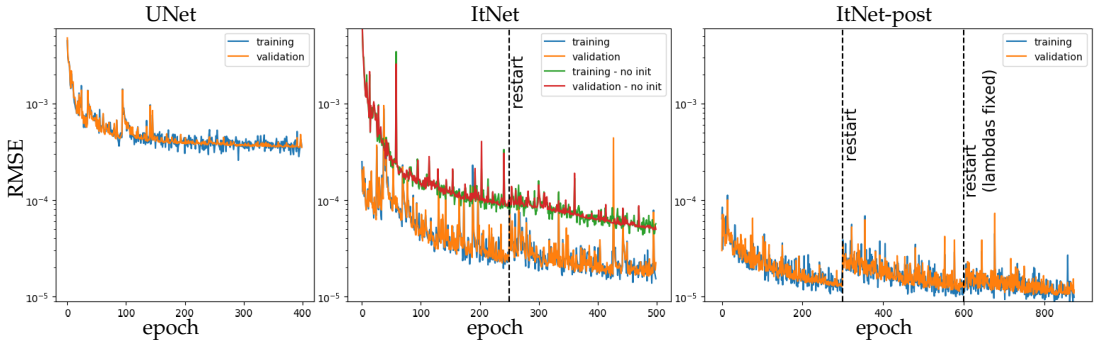


Figure 7.3: **Loss Curves and Network Training.** The first two plots demonstrate that ItNet improves the RMSE by approximately an order of magnitude in comparison to a post-processing by UNet. Furthermore, the gain of our UNet-initialization strategy can be seen in the second graph. The last two plots illustrate the advantages of restarting and of the post-training strategy, respectively. Note that we display the RMSE on the training and validation sets instead of the actual ℓ^2-losses, which behave similarly.

that this initialization strategy is enabled by making use of a powerful enough post-processing sub-network.

(iii) Our data-consistency layer is inspired by a gradient step on the loss $\mathbf{x} \mapsto \frac{\lambda_k}{2}\|\mathbf{A}\,\mathbf{x}-\mathbf{y}\|_2^2$, which would result in the update $\mathbf{x} \mapsto \mathbf{x} - \lambda_k \cdot \mathbf{A}^\top(\mathbf{A}\,\mathbf{x} - \mathbf{y})$. We depart from this scheme by replacing the unfiltered backprojection $\mathbf{A}^\top$ by its filtered counterpart $\mathbf{A}^\ddagger$. This modification leads to significantly improved results for two reasons: (a) It counteracts the fact that the unfiltered backprojection is smoothing. (b) It produces images with pixel values at the right scale. Therefore, we interpret the resulting ItNet as an industry-like iterative CT-algorithm (e.g., see [WN19]), rather than a neurally-augmented convex optimization scheme.

In our experiments, we witnessed only minor effects by computing more than four iterations in (7.6). However, the accuracy was improved by the following post-training strategy: First, the ItNet is extended by one more iteration:

$$\text{ItNet-post}[\boldsymbol{\theta}]\colon \mathbb{R}^m \to \mathbb{R}^n,\ \mathbf{y} \mapsto \left[\bigcirc_{k=1}^{5} \left(\text{DC}_{\lambda_k,\mathbf{y}} \circ \mathbf{U}[\tilde{\boldsymbol{\theta}}_k]\right) \circ \mathbf{A}^\ddagger\right](\mathbf{y}), \tag{7.8}$$

where $\tilde{\boldsymbol{\theta}}_k$ is initialized by the optimized weights of (7.6) for $k = 1, \dots, 5$. Then, ItNet-post is fine-tuned by keeping the weights $\tilde{\boldsymbol{\theta}}_1 = \tilde{\boldsymbol{\theta}}_2 = \tilde{\boldsymbol{\theta}}_3$ of the first three U-Nets fixed and training only the last two iterations (without weight sharing). The RMSE loss curves of our full training pipeline, i.e., UNet $\to$ ItNet + restart $\to$ ItNet-post + 2 $\times$ restart are visualized in Figure 7.3. The obtained improvements indicate that there is a trade-off between increasing the model capacity by more iterations and the difficulty of optimizing the resulting network. A systematic study of such iterative training strategies is left to future research.

We have found that $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3, \lambda_4]$ typically converges to values of the form $\{\lambda_1 < \lambda_2 < \lambda_3 \gg \lambda_4\}$ after sufficiently many training epochs of ItNet. For an additional speed-up of the training, we use the initialization $\boldsymbol{\lambda} = [1.1, 1.3, 1.4, 0.08]$, which was found by pre-training. Similarly, ItNet-post is initialized with the final values of ItNet for $k = 1, 2, 3$, together with $\lambda_4 = 1.0$ and $\lambda_5 = 0.1$.

Remark 7.3. To improve the overall performance of our networks, we have additionally applied the following "tricks", which are ordered by their importance:

(i) Due to statistical fluctuations, the networks typically exhibit slightly different reconstruction errors, despite using the same training pipeline. For the computation of our final reconstructions, we therefore *ensemble* 10 networks, each trained on a different split of the training set.

(ii) Due to the training with small batch sizes, we replace batch normalization of the U-Net architecture by *group normalization* [WH18].

(iii) We equip the U-Net architecture with a few *memory channels*, i.e., one actually has that $\mathbf{U}[\tilde{\boldsymbol{\theta}}]\colon \mathbb{R}^n \times (\mathbb{R}^n)^{c_{\mathrm{mem}}} \to \mathbb{R}^n \times (\mathbb{R}^n)^{c_{\mathrm{mem}}}$ (cf. [PW17; AÖ18]). While the original image-enhancement channel is not altered, the output of the additional channels is propagated through ItNet, playing the role of a hidden state (in the spirit of recurrent neural networks). For our experiments, we have selected $c_{\mathrm{mem}} = 5$.

(iv) It was beneficial to occasionally restart the training of the networks, e.g., see Figure 7.3.

Remark 7.4. The following modifications did not lead to a gain in performance and were omitted:

(i) Improving $\mathbf{A}^\ddagger$ in Step 1 by making some of it components learnable (e.g., the filter), cf. [Wür+16]. Although this is advantageous for the reconstruction quality of $\mathbf{A}^\ddagger$ itself, it leads to worse results for UNet and ItNet. This suggests that a combination of model- and data-based methods benefits most from precise and unaltered physical models.

(ii) Adding additional convolutional-blocks in the measurement domain of ItNet.

(iii) Modifying the standard ℓ^2-loss by incorporating the RMSE or the ℓ^1-norm.

(iv) Utilizing different optimizers such as RAdam [Liu+20] or AdamW [LH19].

7.3 Results

In terms of quantitative similarity measures, we restrict ourselves to reporting the RMSE, which is the main evaluation metric for the challenge [Sid+21a; SP22]. [2]

Winning the Challenge. With our ensembling of ten ItNet-post we were able to achieve near-exact recovery on the test set, thereby winning the challenge with a margin of about

[2] Note that the RMSE is reported for a hold-out subset of 125 images from the training set, which we have used for validation. Hence, these values might differ from the actual results on the official challenge test set. In the final challenge evaluation, our ItNet-post achieved an RMSE of 6.37e-6, see https://www.aapm.org/GrandChallenge/DL-sparse-view-CT/winners.asp.

Table 7.1: **Reconstruction Results.** This table reports the average reconstruction RMSE of different methods over the hold-out validation set. The "Challenge FBP" result corresponds to the filtered-backprojection provided as part of the challenge dataset while "Estimated FBP" results correspond to the estimated filtered-backprojection according to Step 1.

	Baselines		Our Network Variants				Comparison Networks	
	Chall. FBP	Est. FBP	UNet	ItNet	ItNet-post	ItNet-post (ens.)	Tiramisu	LPD
RMSE	5.72e-3	3.40e-3	3.50e-4	1.64e-5	1.05e-5	**6.42e-6**	2.24e-4	1.24e-4

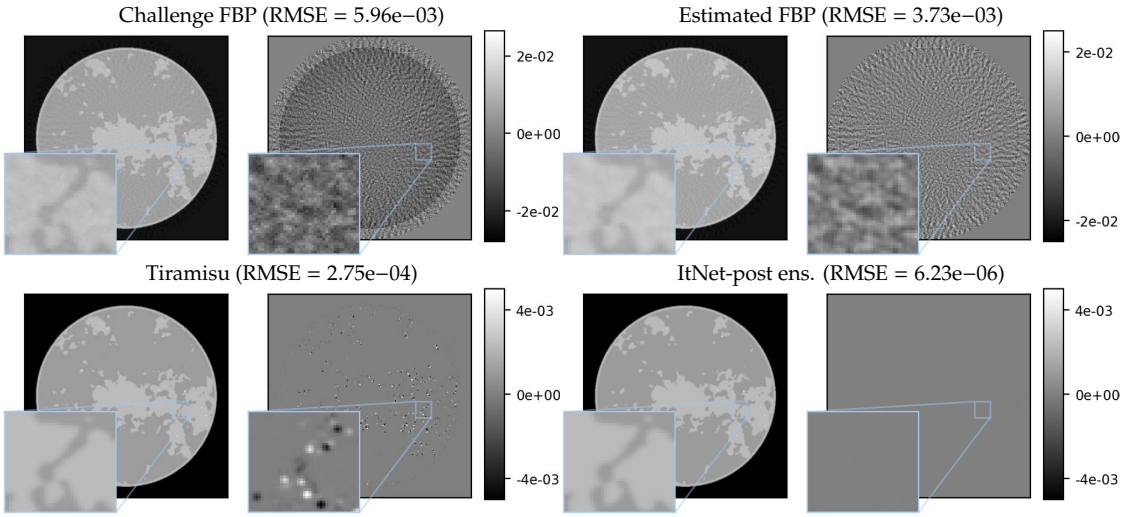


Figure 7.4: **Reconstruction Results.** We display reconstructions of a validation image. The first row compares the filtered backprojection $\mathbf{A}^\ddagger$ provided by the challenge with our own (see Step 1). The second row compares a post-processing by Tiramisu with the (ensembled) ItNet-post. The ground truth image is omitted, since it is visually indistinguishable from the ItNet-post reconstruction.

an order of magnitude ahead of the runner-up team. While the challenge evaluation already provides a comparison to about 25 other methods we additionally provide our own comparison baselines. For this we consider a post-processing of the FBP by the UNet and by the more advanced *Tiramisu* architecture [Bub+19; Jég+17]. Furthermore, we have also trained the iterative *learned primal-dual* (LPD) scheme [AÖ18] (slightly modified by replacing the unfiltered backprojection $\mathbf{A}^\top$ with the filtered backprojection $\mathbf{A}^\ddagger$). LPD has been recently reported as state-of-the-art in the literature, e.g., see [Leu+21]. Table 7.1 lists the average RMSE scores of the considered methods. To give a visual impression as well, reconstructions of an image from the validation set can be found in Figure 7.4. In line with the official challenge evaluation, our own analysis confirms that all ItNet variants outperform the comparison state-of-the-art methods by at least an order of magnitude.

Analysis of Data Consistency. An important feature of solvers for an inverse problem is their data consistency. By this we mean that the reconstructions obtained from a reconstruction method should be consistent with the measurements, in the sense that $\mathbf{y}_0 - \mathbf{A}(\mathrm{Rec}(\mathbf{y}_0))$ is as small as possible. We analyze this aspect in Figure 7.5. We observe that the data-consistency error of the ensembled ItNet-post mostly matches that of the

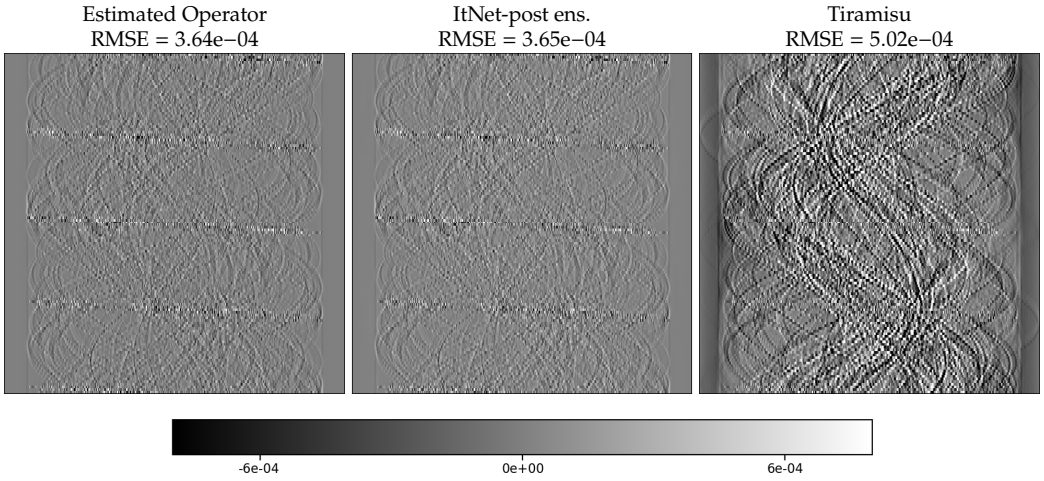


Figure 7.5: **Data-Consistency.** The first image analyzes the accuracy of our forward model by displaying the error $\mathbf{y}_0 - \mathbf{A}\,\mathbf{x}_0$ for a sinogram-image pair $(\mathbf{y}_0, \mathbf{x}_0)$ from the validation set. The visualization of $\mathbf{y}_0 - \mathbf{A} \cdot \text{ItNet-post}(\mathbf{y}_0)$ in the middle is visually nearly indistinguishable, which shows that ItNet-post inherits the inaccuracies from Step 1. Therefore, ItNet-post would allow for even better results if a more accurate forward model was available. It is also interesting to compare with $\mathbf{y}_0 - \mathbf{A} \cdot \text{Tiramisu}(\mathbf{y}_0)$. The considerably larger errors reveal that post-processing by Tiramisu suffers from a lack of data-consistency. All images are shown in the same dynamical range.

estimated forward operator. This suggests that the performance of ItNet-post could still be improved if the exact forward model was available. Further, the data-consistency of the Tiramisu approach is considerably worse, highlighting a typical disadvantage of simple post-processing strategies. Even though the ItNet-post performs only five iterative steps and thus evaluates the forward operator only five times (the corresponding FBP six times), this leads to a significantly improved data-consistency and also improved final reconstruction accuracy. This is in contrast to classical iterative algorithms, such as TV minimization, which often require hundreds of iterations before convergence.

7.4 Discussion

It is worth noting that the reconstruction error of ItNet-post reported in Table 7.1 is not exactly zero, yet comparable to the precision of TV minimization (cf. [Sid+21b]). There is no evidence why even more accurate results should not be achievable, for example, by increasing the internal machine precision of our `PyTorch` implementation (which is ≈1.19e-7 for the default `float32` precision). This would certainly also affect model-based algorithms in the same way. However, non-perfect recovery is not a severe issue from an applied perspective, since it is typically not required for *practical solutions* to inverse problems. We believe that our submission to the AAPM challenge has obtained satisfactory results in that respect (i.e., the reconstruction accuracy is close to machine precision and the reconstructions are visually indistinguishable from the ground-truth phantoms).

While the purpose of data-driven methods is to adapt to a specific data distribution, a generalization to unseen features cannot be taken for granted. Such out-of-distribution

(OOD) features commonly refer to test samples that were not drawn from the same distribution as the training data. In general, the OOD generalization of learned methods is still poorly understood and forms a field of active research [Ant+20; DCH21].

7.5 Conclusion

We have demonstrated that deep-learning-based solvers can produce near-perfect CT reconstructions. While our approach provides some first evidence of feasibility, several aspects remained unexplored as discussed above.

The academic setup of the AAPM challenge has provided an ideal experimental area to test our research hypothesis. Although this has enabled an insightful reliability check, a foundational understanding of learned reconstruction methods is still in its infancy. Similar case studies for different inverse problems and real-world data are important steps for future research. Here, the competitive setting of challenges could help to produce objective results.

Bibliography

[AAT14] R. F. Astudillo, A. Abad, and I. Trancoso. „Accounting for the Residual Uncertainty of Multi-Layer Perceptron Based Features“. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Institute of Electrical and Electronics Engineers (IEEE), 2014, pp. 6859–6863. DOI: 10.1109/ICASSP.2014.6854929.

[ACH21] V. Antun, M. J. Colbrook, and A. C. Hansen. „Can Stable and Accurate Neural Networks be Computed? On the Barriers of Deep Learning and Smale's 18th Problem“. Preprint, arXiv: 2101.08286. 2021.

[Agr+19] A. Agrawal et al. „Differentiable Convex Optimization Layers“. In: *Advances in Neural Information Processing Systems 32 (NeurIPS)*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 9562–9574.

[Ake78] S. B. Akers. „Binary Decision Diagrams“. In: *IEEE Transactions on Computers* C-27.6 (1978), pp. 509–516. DOI: 10.1109/TC.1978.1675141.

[Alb+18] M. Alber et al. „iNNvestigate Neural Networks!“ Preprint, arXiv: 1808.04260. 2018.

[Ame+14] D. Amelunxen et al. „Living on the Edge: Phase Transitions in Convex Programs With Random Data“. In: *Information and Inference: A Journal of the IMA* 3.3 (2014), pp. 224–294. DOI: 10.1093/imaiai/iau005.

[AMJ18] H. K. Aggarwal, M. P. Mani, and M. Jacob. „MoDL: Model-Based Deep Learning Architecture for Inverse Problems“. In: *IEEE transactions on Medical Imaging* 38.2 (2018), pp. 394–405. DOI: 10.1109/TMI.2018.2865356.

[AMT20] A. Arnab, O. Miksik, and P. H. Torr. „On the Robustness of Semantic Segmentation Models to Adversarial Attacks“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.12 (2020), pp. 3040–3053. DOI: 10.1109/TPAMI.2019.2919707.

[AN11] R. Astudillo and J. Neto. „Propagation of Uncertainty Through Multilayer Perceptrons for Robust Automatic Speech Recognition“. In: *12th Annual Conference of the International Speech Communication Association*. International Speech Communication Association (ISCA), 2011, pp. 461–464. DOI: 10.21437/Interspeech.2011-196.

[Ant+20] V. Antun et al. „On Instabilities of Deep Learning in Image Reconstruction and the Potential Costs of AI“. In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30088–30095. DOI: 10.1073/pnas.1907377117.

[AÖ17] J. Adler and O. Öktem. „Solving Ill-Posed Inverse Problems Using Iterative Deep Neural Networks“. In: *Inverse Problems* 33.12 (2017), p. 124007. DOI: 10.1088/1361-6420/aa9581.

[AÖ18] J. Adler and O. Öktem. „Learned Primal-Dual Reconstruction“. In: *IEEE Transactions on Medical Imaging* 37.6 (2018), pp. 1322–1332. DOI: 10.1109/TMI.2018.2799231.

[Arr+19] S. Arridge et al. „Solving Inverse Problems Using Data-Driven Models“. In: *Acta Numerica* 28 (2019), pp. 1–174. DOI: 10.1017/S0962492919000059.

[Bac+15] S. Bach et al. „On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation“. In: *PLOS ONE* 10.7 (2015), pp. 1–46. DOI: 10.1371/journal.pone.0130140.

[BB18] M. Benning and M. Burger. „Modern Regularization Methods for Inverse Problems“. In: *Acta Numererica* 27 (2018), pp. 1–111. DOI: 10.1017/S0962492918000016.

[BBK19] E. Begoli, T. Bhattacharya, and D. Kusnezov. „The Need for Uncertainty Quantification in Machine-Assisted Medical Decision Making“. In: *Nature Machine Intelligence* 1.1 (2019), pp. 20–23.

[BCP21] M. Besançon, A. Carderera, and S. Pokutta. „FrankWolfe.jl: A High-Performance and Flexible Toolbox for Frank-Wolfe Algorithms and Conditional Gradients“. Preprint, arXiv: 2104.06675. 2021.

[Ber03] T. Berger. „Rate-Distortion Theory“. In: *Wiley Encyclopedia of Telecommunications*. Ed. by J. G. Proakis. American Cancer Society, 2003. DOI: 10.1002/0471219282.eot142.

[BGV92] B. E. Boser, I. M. Guyon, and V. N. Vapnik. „A Training Algorithm for Optimal Margin Classifiers“. In: *Proceedings of the 5th Annual Workshop on Computational Learning Theory*. Association for Computing Machinery (ACM), 1992, pp. 144–152. DOI: 10.1145/130385.130401.

[BHN99] R. H. Byrd, M. E. Hribar, and J. Nocedal. „An Interior Point Algorithm for Large-Scale Nonlinear Programming“. In: *SIAM Journal on Optimization* 9.4 (1999), pp. 877–900. DOI: 10.1137/S1052623497325107.

[Bis95] C. M. Bishop. „Training With Noise is Equivalent to Tikhonov Regularization“. In: *Neural Computation* 7.1 (1995), pp. 108–116. DOI: 10.1162/neco.1995.7.1.108.

[BK98] X. Boyen and D. Koller. „Tractable Inference for Complex Stochastic Processes“. In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*. Morgan Kaufmann Publishers, Inc., 1998, pp. 33–42.

[BMR17] A. S. Bandeira, D. G. Mixon, and B. Recht. „Compressive Classification and the Rare Eclipse Problem“. In: *Compressed Sensing and its Applications: 2nd International MATHEON Conference 2015*. Ed. by H. Boche et al. Applied and Numerical Harmonic Analysis. Springer Cham, 2017, pp. 197–220. DOI: 10.1007/978-3-319-69802-1_6.

[Bor+17] A. Bora et al. „Compressed Sensing Using Generative Models“. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. Ed. by D. Precup and Y. W. Teh. Vol. 70. PMLR, 2017, pp. 537–546.

[Boy+16] C. Boyer et al. „On the Generation of Sampling Schemes for Magnetic Resonance Imaging“. In: *SIAM Journal on Imaging Sciences* 9.4 (2016), pp. 2039–2072. DOI: 10.1137/16M1059205.

[BPZ19] G. Braun, S. Pokutta, and D. Zink. „Lazifying Conditional Gradient Algorithms“. In: *Journal of Machine Learning Research* 20.1 (2019), pp. 2577–2618.

[Bre+84] L. Breiman et al. *Classification and Regression Trees*. Taylor and Francis, 1984.

[Bri89] J. S. Bridle. „Training Stochastic Model Recognition Algorithms as Networks Can Lead to Maximum Mutual Information Estimation of Parameters“. In: *Advances in Neural Information Processing Systems 2 (NIPS)*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1989.

[Bri90] J. S. Bridle. „Probabilistic Interpretation of Feedforward Classification Network Outputs, With Relationships to Statistical Pattern Recognition“. In: *Neurocomputing*. Ed. by F. F. Soulié and J. Hérault. Springer, Berlin Heidelberg, 1990, pp. 227–236.

[Bro+17] T. B. Brown et al. „Adversarial Patch“. Preprint, arXiv: 1712.09665. 2017.

[Bry86] R. E. Bryant. „Graph-Based Algorithms for Boolean Function Manipulation“. In: *IEEE Transactions on Computers* 100.8 (1986), pp. 677–691. DOI: 10.1109/TC.1986.1676819.

[Bub+19] T. A. Bubba et al. „Learning the Invisible: A Hybrid Deep Learning-Shearlet Framework for Limited Angle Computed Tomography“. In: *Inverse Problems* 35.6 (2019), p. 064002. DOI: 10.1088/1361-6420/ab10ca.

[Byr+95] R. H. Byrd et al. „A Limited Memory Algorithm for Bound Constrained Optimization“. In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208. DOI: 10.1137/0916069.

[Car22] N. Carlini. *A Complete List of All (arXiv) Adversarial Example Papers*. Available online: https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html, accessed on 2022-03-16. 2022.

[CBP21] A. Carderera, M. Besançon, and S. Pokutta. „Simple Steps Are All You Need: Frank-Wolfe and Generalized Self-Concordant Functions“. Preprint, arXiv: 2105.13913. 2021.

[CGL79] T. F. Chan, G. H. Golub, and R. J. LeVeque. *Updating Formulae and a Pairwise Algorithm for Computing Sample Variances*. Tech. rep. Stanford University, 1979.

[Cha+17] S. Chakraborty et al. „Interpretability of Deep Learning Models: A Survey of Results“. In: *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld / SCALCOM / UIC / ATC / CBDCom / IOP / SCI)*. Institute of Electrical and Electronics Engineers (IEEE), 2017, pp. 1–6. DOI: 10.1109/UIC-ATC.2017.8397411.

[Che+17a] H. Chen et al. „Low-dose CT via Convolutional Neural Network“. In: *Biomedical Optics Express* 8.2 (2017), pp. 679–694. DOI: 10.1364/BOE.8.000679.

[Che+17b] H. Chen et al. „Low-dose CT With a Residual Encoder-Decoder Convolutional Neural Network“. In: *IEEE Transactions on Medical Imaging* 36.12 (2017), pp. 2524–2535. DOI: 10.1109/TMI.2017.2715284.

[Che+18] H. Chen et al. „LEARN: Learned Experts' Assessment-Based Reconstruction Network for Sparse-Data CT“. In: *IEEE Transactions on Medical Imaging* 37.6 (2018), pp. 1333–1347. DOI: 10.1109/TMI.2018.2805692.

[Che+20] K. Cheng et al. „Addressing the False Negative Problem of Deep Learning MRI Reconstruction Models by Adversarial Attacks and Robust Training“. In: *Proceedings of the 3rd Conference on Medical Imaging With Deep Learning (MIDL)*. Ed. by T. Arbel et al. Vol. 121. PMLR, 2020, pp. 121–135.

[Cho+14] K. Cho et al. „Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation“. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics (ACL), 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

[Cho+15] F. Chollet et al. *Keras*. Available online: https://github.com/keras-team/keras, accessed on 2019-02-06. 2015.

[Chu+20] I. Y. Chun et al. „Momentum-Net: Fast and Convergent Iterative Neural Network for Inverse Problems“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. DOI: 10.1109/TPAMI.2020.3012955.

[Cir+12] D. Cireşan et al. „Multi-Column Deep Neural Network for Traffic Sign Classification“. In: *Neural Networks* 32 (2012), pp. 333–338. DOI: 10.1016/j.neunet.2012.02.023.

[CJ08] C. P. de Campos and Q. Ji. „Strategy Selection in Influence Diagrams Using Imprecise Probabilities“. In: *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 2008, pp. 121–128.

[CL97] A. Chambolle and P.-L. Lions. „Image Recovery via Total Variation Minimization and Related Problems“. In: *Numerische Mathematik* 76.2 (1997), pp. 167–188. DOI: 10.1007/s002110050258.

[CM92] O. Coudert and J. C. Madre. „Implicit and Incremental Computation of Primes and Essential Primes of Boolean Functions.“ In: *Proceedings 29th ACM/IEEE Design Automation Conference*. Vol. 92. Institute of Electrical and Electronics Engineers (IEEE), 1992, pp. 36–39. DOI: 10.1109/DAC.1992.227866.

[CNL11] A. Coates, A. Ng, and H. Lee. „An Analysis of Single-Layer Networks in Unsupervised Feature Learning“. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. PMLR, 2011, pp. 215–223.

[Com21] C. W. Combettes. „Frank-Wolfe Methods for Optimization and Machine Learning“. PhD thesis. Georgia Institute of Technology, 2021.

[Coo71] S. A. Cook. „The Complexity of Theorem-Proving Procedures“. In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery (ACM), 1971, pp. 151–158. DOI: 10.1145/800157.805047.

[Cor+09] T. H. Cormen et al. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[CP21] C. W. Combettes and S. Pokutta. „Complexity of Linear Minimization and Projection on Some Sets“. Preprint, arXiv: 2101.10040. 2021.

[CRT06] E. J. Candès, J. K. Romberg, and T. Tao. „Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information“. In: *IEEE Transactions on Information Theory* 52.2 (2006), pp. 489–509. DOI: 10.1109/TIT.2005.862083.

[CV95] C. Cortes and V. Vapnik. „Support Vector Networks“. In: *Machine Learning* 20 (1995), pp. 273–297. DOI: 10.1007/BF00994018.

[CW17] N. Carlini and D. Wagner. „Towards Evaluating the Robustness of Neural Networks“. In: *IEEE Symposium on Security and Privacy*. Institute of Electrical and Electronics Engineers (IEEE), 2017, pp. 39–57. DOI: 10.1109/SP.2017.49.

[Dar00] A. Darwiche. „On the Tractable Counting of Theory Models and Its Application to Belief Revision and Truth Maintenance“. Preprint, arXiv: cs/0003044. 2000.

[Dar11] A. Darwiche. „SDD: A New Canonical Representation of Propositional Knowledge Bases“. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*. Ed. by T. Walsh. AAAI Press/IJCAI, 2011, pp. 819–826. DOI: 10.5591/978-1-57735-516-8/IJCAI11-143.

[DB90] M. Drummond and J. Bresina. *Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction*. NASA, Ames Research Center, Artificial Intelligence Research Branch, 1990.

[DCH21] M. Z. Darestani, A. Chaudhari, and R. Heckel. „Measuring Robustness in Deep Learning Based Compressive Sensing“. In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. Ed. by M. Meila and T. Zhang. Vol. 139. PMLR, 2021, pp. 2433–2444.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley New York, 1973.

[DK17] F. Doshi-Velez and B. Kim. „Towards a Rigorous Science of Interpretable Machine Learning“. Preprint, arXiv: 1702.08608. 2017.

[Don+16] C. Dong et al. „Image Super-Resolution Using Deep Convolutional Networks“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (2016), pp. 295–307. DOI: 10.1109/TPAMI.2015.2439281.

[DP94] X. Deng and C. H. Papadimitriou. „On the Complexity of Cooperative Solution Concepts“. In: *Mathematics of Operations Research* 19.2 (1994), pp. 257–266.

[Duc+08] J. Duchi et al. „Efficient Projections Onto the ℓ_1-Ball for Learning in High Dimensions“. In: *Proceedings of the 25th International Conference on Machine Learning (ICML)*. Association for Computing Machinery (ACM), 2008, pp. 272–279. DOI: 10.1145/1390156.1390191.

[Dug66] J. Dugundji. *Topology*. Ed. by I. Kaplansky. Allyn and Bacon, Inc., 1966.

[Edg08] G. Edgar. *Measure, Topology, and Fractal Geometry*. Ed. by S. Axler and K. A. Ribet. 2nd. Spinger, New York, 2008. DOI: 10.1007/978-0-387-74749-1.

[EG95] T. Eiter and G. Gottlob. „The Complexity of Logic-Based Abduction“. In: *Journal of the ACM (JACM)* 42.1 (1995), pp. 3–42.

[Ela17] M. Elad. *Deep, Deep Trouble: Deep Learning's Impact on Image Processing, Mathematics, and Humanity*. Available online: https://sinews.siam.org/Details-Page/deep-deep-trouble, accessed on 2020-09-21. SIAM News. 2017.

[Ern20] P. Ernst. *Pytorch Implementation of Scikit-Image's Radon Function, Version 0.1.4*. Available online: https://github.com/phernst/pytorch_radon, accessed on 2020-10-29. 2020.

[Eyk+18] K. Eykholt et al. „Robust Physical-World Attacks on Deep Learning Visual Classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE), 2018, pp. 1625–1634. DOI: 10.1109/CVPR.2018.00175.

[FA14] H. Fayed and A. 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.

[Fan+21] F.-L. Fan et al. „On Interpretability of Artificial Neural Networks: A Survey". In: *IEEE Transactions on Radiation and Plasma Medical Sciences* 5.6 (2021), pp. 741–760. DOI: 10.1109/TRPMS.2021.3066428.

[Fes17] J. A. Fessler. *Analytical Tomographic Image Reconstruction Methods (Chapter 3 of Book Draft)*. 2017. URL: https://web.eecs.umich.edu/~fessler/book/c-tomo.pdf.

[FH85] E. Fix and J. Hodges. *Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties*. Vol. 1-2. USAF School of Aviation Medicine, 1985.

[Fin+19] S. G. Finlayson et al. „Adversarial Attacks on Medical Machine Learning". In: *Science* 363.6433 (2019), pp. 1287–1289. DOI: 10.1126/science.aaw4399.

[FR13] S. Foucart and H. Rauhut. *A Mathematical Introduction to Compressive Sensing*. Applied and Numerical Harmonic Analysis. Birkhäuser Basel, 2013. DOI: 10.1007/978-0-8176-4948-7.

[Fuk80] K. Fukushima. „Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". In: *Biological Cybernetics* 36 (1980), pp. 193–202. DOI: 10.1007/BF00344251.

[FV17] R. C. Fong and A. Vedaldi. „Interpretable Explanations of Black Boxes by Meaningful Perturbation". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Institute of Electrical and Electronics Engineers (IEEE), 2017, pp. 3429–3437. DOI: 10.1109/ICCV.2017.371.

[FW56] M. Frank and P. Wolfe. „An Algorithm for Quadratic Programming". In: *Naval Research Logistics Quarterly* 3.1-2 (1956), pp. 95–110. DOI: 10.1002/nav.3800030109.

[FWJ08] S. S. Fatima, M. Wooldridge, and N. R. Jennings. „A Linear Approximation Method for the Shapley Value". In: *Artificial Intelligence* 172.14 (2008), pp. 1673–1699. DOI: 10.1016/j.artint.2008.05.003.

[GBB11] X. Glorot, A. Bordes, and Y. Bengio. „Deep Sparse Rectifier Neural Networks“. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. PMLR, 2011, pp. 315–323.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Gil+18] L. H. Gilpin et al. „Explaining Explanations: An Overview of Interpretability of Machine Learning“. In: *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. Institute of Electrical and Electronics Engineers (IEEE), 2018, pp. 80–89. DOI: 10.1109/DSAA.2018.00018.

[Gil74] J. T. Gill. „Computational Complexity of Probabilistic Turing Machines“. In: *Proceedings of the 6th Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery (ACM), 1974, pp. 91–95. DOI: 10.1145/800119.803889.

[GKX10] M. D. Gupta, S. Kumar, and J. Xiao. „L1 Projections With Box Constraints“. Preprint, arXiv: 1010.0141. 2010.

[GL10] K. Gregor and Y. LeCun. „Learning Fast Approximations of Sparse Coding“. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML)*. Ed. by J. Fürnkranz and T. Joachims. Omnipress, 2010, pp. 399–406.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. English. Vol. 2. Algorithms and Combinatorics. Springer, 1988.

[GM75] R. Glowinski and A. Marroco. „Sur l'Approximation, par Éléments Finis d'Ordre Un, et la Résolution, par Pénalisation-Dualité d'une Classe de Problèmes de Dirichlet Non Linéaires“. In: *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* 9.R2 (1975), pp. 41–76.

[GM76] D. Gabay and B. Mercier. „A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation“. In: *Computers and Mathematics With Applications* 2.1 (1976), pp. 17–40. DOI: 10.1016/0898-1221(76)90003-1.

[GM86] J. Guélat and P. Marcotte. „Some Comments of Wolfe's "Away Step"“. In: *Mathematical Programming* 35.1 (1986), pp. 110–119.

[GMM20] M. Genzel, J. Macdonald, and M. März. „Solving Inverse Problems With Deep Neural Networks – Robustness Included?“ Preprint, arXiv: 2011.04268. 2020.

[GMM21] M. Genzel, J. Macdonald, and M. März. „AAPM DL-Sparse-View CT Challenge Submission Report: Designing an Iterative Network for Fanbeam-CT With Unknown Geometry“. Preprint, arXiv: 2106.00280. 2021.

[GMS21] M. Genzel, M. März, and R. Seidel. „Compressed Sensing With 1D Total Variation: Breaking Sample Complexity Barriers via Non-Uniform Recovery“. Preprint, arXiv: 2001.09952. 2021.

[Got+20] N. M. Gottschling et al. „The Troublesome Kernel: Why Deep Learning for Inverse Problems is Typically Unstable“. Preprint, arXiv: 2001.01258. 2020.

[GOW21] D. Gilton, G. Ongie, and R. Willett. „Deep Equilibrium Architectures for Inverse Problems in Imaging“. Preprint, arXiv: 2102.07944. 2021.

[GR18] J. Gast and S. Roth. „Lightweight Probabilistic Deep Networks“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE), 2018, pp. 3369–3378. DOI: 10.1109/CVPR.2018.00355.

[GSS15] I. J. Goodfellow, J. Shlens, and C. Szegedy. „Explaining and Harnessing Adversarial Examples“. Preprint, arXiv: 1412.6572. 2015.

[HA15] J. M. Hernández-Lobato and R. P. Adams. „Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks“. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*. Vol. 37. JMLR, 2015, pp. 1861–1869.

[HA20] A. Hauptmann and J. Adler. „On the Unreasonable Effectiveness of CNNs“. Preprint, arXiv: 2007.14745. 2020.

[Ham+18] K. Hammernik et al. „Learning a Variational Network for Reconstruction of Accelerated MRI Data“. In: *Magnetic Resonance in Medicine* 79.6 (2018), pp. 3055–3071. DOI: 10.1002/mrm.26977.

[Ham+19] K. Hammernik et al. „Σ-net: Systematic Evaluation of Iterative Deep Neural Networks for Fast Parallel MR Image Reconstruction“. Preprint, arXiv: 1912.09278. 2019.

[Has+20] H. Hassani et al. „Stochastic Conditional Gradient++“. Preprint, arXiv: 1902.06992. 2020.

[Hau+08] J. Haupt et al. „Compressed Sensing for Networked Data“. In: *IEEE Signal Processing Magazine* 25.2 (2008), pp. 92–101. DOI: 10.1109/MSP.2007.914732.

[Hau+20] A. Hauptmann et al. „Multi-Scale Learned Iterative Reconstruction“. In: *IEEE Transactions on Computational Imaging* 6 (2020), pp. 843–856. DOI: 10.1109/TCI.2020.2990299.

[HCC18] T. Huster, C.-Y. J. Chiang, and R. Chadha. „Limitations of the Lipschitz Constant as a Defense Against Adversarial Examples“. In: *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD) Workshops*. Springer, Cham, 2018, pp. 16–29. DOI: 10.1007/978-3-030-13453-2_2.

[He+16] K. He et al. „Deep Residual Learning for Image Recognition“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE), 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[Hea+21] H. Heaton et al. „Feasibility-Based Fixed Point Networks“. Preprint, arXiv: 2104.14090. 2021.

[HK92] L. Holmstrom and P. Koistinen. „Using Additive Noise in Back-Propagation Training“. In: *IEEE Transactions on Neural Networks* 3.1 (1992), pp. 24–38. DOI: 10.1109/72.105415.

[HL16] E. Hazan and H. Luo. „Variance-Reduced and Projection-Free Stochastic Optimization“. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. PMLR, 2016, pp. 1263–1271.

[Hoe94] W. Hoeffding. „Probability Inequalities for Sums of Bounded Random Variables“. In: *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[Hol+17] A. Holzinger et al. „What Do We Need to Build Explainable AI Systems for the Medical Domain?“ Preprint, arXiv: 1712.09923. 2017.

[Hor91] K. Hornik. „Approximation Capabilities of Multilayer Feedforward Networks“. In: *Neural Networks* 4.2 (1991), pp. 251–257.

[Hoy02] P. Hoyer. „Non-Negative Sparse Coding“. In: *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*. Institute of Electrical and Electronics Engineers (IEEE), 2002, pp. 557–565. DOI: 10.1109/NNSP.2002.1030067.

[Hua+17] G. Huang et al. „Densely Connected Convolutional Networks“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE), 2017, pp. 4700–4708. DOI: 10.1109/CVPR.2017.243.

[Hua+18] Y. Huang et al. „Some Investigations on Robustness of Deep Learning in Limited Angle Tomography“. In: *21st International Conference Medical Image Computing and Computer Assisted Intervention (MICCAI), Proceedings, Part I*. Ed. by A. F. Frangi et al. Springer Cham, 2018, pp. 145–153. DOI: 10.1007/978-3-030-00928-1_17.

[Hub81] P. J. Huber. *Robust Statistics*. John Wiley and Sons, 1981.

[INM19] A. Ignatiev, N. Narodytska, and J. Marques-Silva. „Abduction-Based Explanations for Machine Learning Models“. In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. Vol. 33. Association for the Advancement of Artificial Intelligence (AAAI), 2019, pp. 1511–1519. DOI: 10.1609/aaai.v33i01.33011511.

[IS15] S. Ioffe and C. Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Ed. by F. Bach and D. Blei. Vol. 37. PMLR, 2015, pp. 448–456.

[Iva71] A. G. Ivakhnenko. „Polynomial Theory of Complex Systems“. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-1.4 (1971), pp. 364–378. DOI: 10.1109/TSMC.1971.4308320.

[Jég+17] S. Jégou et al. „The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE), 2017, pp. 11–19. DOI: 10.1109/CVPRW.2017.156.

[Jin+17] K. H. Jin et al. „Deep Convolutional Neural Network for Inverse Problems in Imaging“. In: *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4509–4522. DOI: 10.1109/TIP.2017.2713099.

[JMB20] D. Janzing, L. Minorics, and P. Bloebaum. „Feature Relevance Quantification in Explainable AI: A Causal Problem“. In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by S. Chiappa and R. Calandra. Vol. 108. PMLR, 2020, pp. 2907–2916.

[JNV14] P. Jylänki, A. Nummenmaa, and A. Vehtari. „Expectation Propagation for Neural Networks With Sparsity-Promoting Priors“. In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1849–1901.

[KB14] D. P. Kingma and J. Ba. „Adam: A Method for Stochastic Optimization“. Preprint, arXiv: 1412.6980. 2014.

[KC17] J. Kim and J. Canny. „Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Institute of Electrical and Electronics Engineers (IEEE), 2017, pp. 2942–2950. DOI: 10.1109/ICCV.2017.320.

[KGB17] A. Kurakin, I. Goodfellow, and S. Bengio. „Adversarial Examples in the Physical World“. Preprint, arXiv: 1607.02533. 2017.

[Kho+19] P. Khosravi et al. „What to Expect of Classifiers? Reasoning About Logistic Regression With Missing Features“. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI, 2019, pp. 2716–2724. DOI: 10.24963/ijcai.2019/377.

[Kin+17] P.-J. Kindermans et al. „The (Un)reliability of Saliency Methods“. Preprint, arXiv: 1711.00867. 2017.

[KMY17] E. Kang, J. Min, and J. C. Ye. „A Deep Convolutional Neural Network Using Directional Wavelets for Low-Dose X-Ray CT Reconstruction“. In: *Medical Physics* 44.10 (2017), e360–e375. DOI: 10.1002/mp.12344.

[Kno+20a] F. Knoll et al. „Advancing Machine Learning for MR Image Reconstruction With an Open Competition: Overview of the 2019 fastMRI Challenge“. In: *Magnetic Resonance in Medicine* 84.6 (2020), pp. 3054–3070. DOI: 10.1002/mrm.28338.

[Kno+20b] F. Knoll et al. „fastMRI: A Publicly Available Raw k-Space and DICOM Dataset of Knee Images for Accelerated MR Image Reconstruction Using Machine Learning“. In: *Radiology: Artificial Intelligence* 2.1 (2020), e190007. DOI: 10.1148/ryai.2020190007.

[Kob+17] E. Kobler et al. „Variational Networks: Connecting Variational Methods and Deep Learning“. In: *Pattern Recognition*. Ed. by V. Roth and T. Vetter. Springer, Cham, 2017, pp. 281–293.

[Kob+20] E. Kobler et al. „Total Deep Variation: A Stable Regularizer for Inverse Problems“. Preprint, arXiv: 2006.08789. 2020.

[KS06] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Vol. 160. Applied Mathematical Sciences. Springer, New York, 2006. DOI: 10.1007/b138659.

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. „ImageNet Classification With Deep Convolutional Neural Networks“. In: *Advances in Neural Information Processing Systems 25 (NIPS)*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012, pp. 1097–1105.

[Kuh55] H. W. Kuhn. „The Hungarian Method for the Assignment Problem“. In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. DOI: 10.1002/nav.3800020109.

[Lap+16] S. Lapuschkin et al. „Analyzing Classifiers: Fisher Vectors and Deep Neural Networks“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE), 2016, pp. 2912–2920. DOI: 10.1109/CVPR.2016.318.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. „Deep Learning“. In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[LeC+89] Y. LeCun et al. „Backpropagation Applied to Handwritten Zip Code Recognition“. In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[LeC+98] Y. LeCun et al. „Gradient-Based Learning Applied to Document Recognition“. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[Leu+21] J. Leuschner et al. „Quantitative Comparison of Deep Learning-Based Image Reconstruction Methods for Low-Dose and Sparse-Angle CT Applications“. In: *Journal of Imaging* 7.3 (2021), pp. 1–49. DOI: 10.3390/jimaging7030044.

[LGM98] M. L. Littman, J. Goldsmith, and M. Mundhenk. „The Computational Complexity of Probabilistic Planning“. In: *Journal of Artificial Intelligence Research* 9 (1998), pp. 1–36. DOI: 10.1613/jair.505.

[LH18] B. Lyu and A. 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*. Association for Computing Machinery (ACM), 2018, pp. 89–96.

[LH19] I. Loshchilov and F. Hutter. „Decoupled Weight Decay Regularization“. In: *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[Li+20] H. Li et al. „NETT: Solving Inverse Problems With Deep Neural Networks“. In: *Inverse Problems* 36.6 (2020), p. 065005. DOI: 10.1088/1361-6420/ab6d57.

[LIT92] P. Langley, W. Iba and, and K. Thompson. „An Analysis of Bayesian Classifiers“. In: *Proceedings of the 10th National Conference on Artificial Intelligence*. AAAI Press, 1992, pp. 223–228.

[Liu+18] X. Liu et al. „DPatch: Attacking Object Detectors With Adversarial Patches“. Preprint, arXiv: 1806.02299. 2018.

[Liu+20] L. Liu et al. „On the Variance of the Adaptive Learning Rate and Beyond“. In: *8th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

[LJ15] S. Lacoste-Julien and M. Jaggi. „On the Global Linear Convergence of Frank-Wolfe Optimization Variants“. In: *Advances in Neural Information Processing Systems 28 (NIPS)*. Vol. 28. MIT Press, 2015, pp. 496–504.

[LL17] S. M. Lundberg and S.-I. Lee. „A Unified Approach to Interpreting Model Predictions“. In: *Advances in Neural Information Processing Systems 30 (NIPS)*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 4765–4774.

[LMP01] M. L. Littman, S. M. Majercik, and T. Pitassi. „Stochastic Boolean Satisfiability“. In: *Journal of Automated Reasoning* 27.3 (2001), pp. 251–296.

[LP66] E. Levitin and B. Polyak. „Constrained Minimization Methods“. In: *USSR Computational Mathematics and Mathematical Physics* 6.5 (1966), pp. 1–50. DOI: 10.1016/0041-5553(66)90114-5.

[Lus+08] M. Lustig et al. „Compressed Sensing MRI“. In: *IEEE Signal Processing Magazine* 25.2 (2008), pp. 72–82. DOI: 10.1109/MSP.2007.914728.

[Mac+19] J. Macdonald et al. „A Rate-Distortion Framework for Explaining Neural Network Decisions“. Preprint, arXiv: 1905.11092. 2019.

[Mac+20] j. Macdonald et al. „Explaining Neural Network Decisions Is Hard“. XXAI Workshop, 37th International Conference on Machine Learning (ICML). 2020.

[Mar+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Available online: http://tensorflow.org/, accessed on 2019-02-06. 2015.

[Mar00] P. Marquis. „Consequence Finding Algorithms“. In: *Handbook of Defeasible Reasoning and Uncertainty Management Systems: Algorithms for Uncertainty and Defeasible Reasoning*. Ed. by J. Kohlas and S. Moral. Springer Netherlands, 2000, pp. 41–145. DOI: 10.1007/978-94-017-1737-3_3.

[Mar91] P. Marquis. „Extending Abduction from Propositional to First-Order Logic“. In: *International Workshop on Fundamentals of Artificial Intelligence Research (FAIR)*. Springer, 1991, pp. 141–155.

[MB17] A. Mukherjee and A. Basu. „Lower Bounds Over Boolean Inputs for Deep Neural Networks With ReLU Gates“. Preprint, arXiv: 1711.03073. 2017.

[MBP21] J. Macdonald, M. Besançon, and S. Pokutta. „Interpretable Neural Networks With Frank-Wolfe: Sparse Relevance Maps and Relevance Orderings“. Preprint, arXiv: 2110.08105. 2021.

[McB+18] M. P. McBee et al. „Deep Learning in Radiology“. In: *Academic Radiology* 25.11 (2018), pp. 1472–1480. DOI: 10.1016/j.acra.2018.02.018.

[McC86] R. K. McConnell. „Method of and Apparatus for Pattern Recognition“. 1986. URL: https://www.osti.gov/biblio/6007283.

[MHK20] A. Mokhtari, H. Hassani, and A. Karbasi. „Stochastic Conditional Gradient Methods: From Convex Minimization to Submodular Maximization“. In: *Journal of Machine Learning Research* 21.105 (2020), pp. 1–49.

[MHN13] A. L. Maas, A. Y. Hannun, and A. Y. Ng. „Rectifier Nonlinearities Improve Neural Network Acoustic Models“. Deep Learning for Audio, Speech and Language Processing Workshop, 30th International Conference on Machine Learning (ICML). 2013.

[Min01] T. P. Minka. „A Family of Algorithms for Approximate Bayesian Inference“. AAI0803033. PhD thesis. Massachusetts Institute of Technology, 2001.

[MKG18] R. Michelmore, M. Kwiatkowska, and Y. Gal. „Evaluating Uncertainty Quantification in End-To-End Autonomous Driving Control“. Preprint, arXiv: 1811.06817. 2018.

[MOM98] V. M. Manquinho, A. L. Oliveira, and J. Marques-Silva. „Models and Algorithms for Computing Minimum-Size Prime Implicants“. In: *Proceedings of the 3rd International Workshop on Boolean Problems (IWBP)*. 1998.

[MP43] W. McCulloch and W. Pitts. „A Logical Calculus of Ideas Immanent in Nervous Activity“. In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 127–147. DOI: 10.1007/BF02478259.

[MS12] J. L. Mueller and S. Siltanen. *Linear and Nonlinear Inverse Problems With Practical Applications*. SIAM, 2012. DOI: 10.1137/1.9781611972344.

[MSG20] G. R. Machado, E. Silva, and R. R. Goldschmidt. „Adversarial Machine Learning in Image Classification: A Survey Towards the Defender's Perspective“. Preprint, arXiv: 2009.03728. 2020.

[MSM18] G. Montavon, W. Samek, and K.-R. Müller. „Methods for Interpreting and Understanding Deep Neural Networks“. In: *Digital Signal Processing* 73 (2018), pp. 1–15. DOI: 10.1016/j.dsp.2017.10.011.

[Muc+20] M. J. Muckley et al. „State-of-the-art Machine Learning MRI Reconstruction in 2020: Results of the Second fastMRI Challenge“. Preprint, arXiv: 2012.06318. 2020.

[MW22] J. Macdonald and S. Wäldchen. „A Complete Characterisation of ReLU-Invariant Distributions“. In: *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by G. Camps-Valls, F. J. R. Ruiz, and I. Valera. Vol. 151. PMLR, 2022, pp. 1457–1484.

[Nég+20] G. Négiar et al. „Stochastic Frank-Wolfe for Constrained Finite-Sum Minimization“. Preprint, arXiv: 2002.11860. 2020.

[NH10] V. Nair and G. E. Hinton. „Rectified Linear Units Improve Restricted Boltzmann Machines“. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML)*. Omnipress, 2010, pp. 807–814.

[Nie18] M. A. Nielsen. *Neural Networks and Deep Learning*. Available online: http://neuralnetworksanddeeplearning.com/, accessed on 2019-01-15. Online Book. 2018.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. second. Springer, 2006.

[NW13] D. Needell and R. Ward. „Near-Optimal Compressed Sensing Guarantees for Total Variation Minimization“. In: *IEEE Transactions on Image Processing* 22.10 (2013), pp. 3941–3949. DOI: 10.1109/TIP.2013.2264681.

[Oal+20] L. Oala et al. „ML4H Auditing: From Paper to Practice“. In: *Proceedings of the Machine Learning for Health NeurIPS Workshop*. Vol. 136. PMLR, 2020, pp. 280–317.

[Ong+20] G. Ongie et al. „Deep Learning Techniques for Inverse Problems in Imaging“. In: *IEEE Journal on Selected Areas in Information Theory* 1.1 (2020), pp. 39–56. DOI: 10.1109/JSAIT.2020.2991563.

[Ort+20] G. Ortiz-Jimenez et al. „Optimism in the Face of Adversity: Understanding and Improving Deep Learning Through Adversarial Robustness“. Preprint, arXiv: 2010.09624. 2020.

[OW98] M. Opper and O. Winther. „A Bayesian Approach to On-Line Learning“. In: *On-Line Learning in Neural Networks* (1998), pp. 363–378. DOI: 10.1017/CBO9780511569920.017.

[OWT19] J. Oramas, K. Wang, and T. Tuytelaars. „Visual Explanation by Interpretation: Improving Visual Feedback Capabilities of Deep Neural Networks“. In: *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[Par02] J. D. Park. „MAP Complexity Results and Approximation Methods“. In: *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI)*. Morgan Kaufmann Publishers, Inc., 2002, pp. 388–396.

[Par96] I. Parberry. *Circuit Complexity and Feedforward Neural Networks*. Hillsdale, NJ: Lawrence Erlbaum, 1996.

[Pas+17] A. Paszke et al. *Automatic Differentiation in PyTorch*. Available online: https://openreview.net/forum?id=BJJsrmfCZ, accessed on 2020-09-21. Autodiff Workshop, Conference on Neural Information Processing Systems (NIPS). 2017.

[PD07] F. Perronnin and C. Dance. „Fisher Kernels on Visual Vocabularies for Image Categorization“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE), 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383266.

[PMG16] N. Papernot, P. McDaniel, and I. Goodfellow. „Transferability in Machine Learning: From Phenomena to Black-Box Attacks Using Adversarial Samples“. Preprint, arXiv: 1605.07277. 2016.

[Poo15] C. Poon. „On the Role of Total Variation in Compressed Sensing“. In: *SIAM Journal on Imaging Sciences* 8.1 (2015), pp. 682–720. DOI: 10.1137/140978569.

[Pro56] Y. V. Prokhorov. „Convergence of Random Processes and Limit Theorems in Probability Theory“. In: *Theory of Probability and Its Applications* 1.2 (1956), pp. 157–214. DOI: 10.1137/1101016.

[PRV18] P. Petersen, M. Raslan, and F. Voigtlaender. „Topological Properties of the Set of Functions Generated by Neural Networks of Fixed Size“. Preprint, arXiv: 1806.08459. 2018.

[PSM10] F. Perronnin, J. Sánchez, and T. Mensink. „Improving the Fisher Kernel for Large-Scale Image Classification“. In: *Computer Vision – ECCV 2010, Proceedings, Part IV*. Springer, Berlin Heidelberg, 2010, pp. 143–156. DOI: 10.1007/978-3-642-15561-1_11.

[PV18] P. Petersen and F. Voigtlaender. „Optimal Approximation of Piecewise Smooth Functions Using Deep ReLU Neural Networks“. In: *Neural Networks* 108 (2018), pp. 296–330. DOI: 10.1016/j.neunet.2018.08.019.

[PW17] P. Putzky and M. Welling. „Recurrent Inference Machines for Solving Inverse Problems“. Preprint, arXiv: 1706.04008. 2017.

[Qui86] J. R. Quinlan. „Induction of Decision Trees“. In: *Machine Learning* 1.1 (1986), pp. 81–106. DOI: 10.1023/A:1022643204877.

[Qui93] E. T. Quinto. „Singularities of the X-Ray Transform and Limited Data Tomography in $\mathbb{R}^2$ and $\mathbb{R}^3$“. In: *SIAM Journal on Mathematical Analysis* 24.5 (1993), pp. 1215–1225. DOI: 10.1137/0524069.

[RBL20] A. Raj, Y. Bresler, and B. Li. „Improving Robustness of Deep-Learning-Based Image Reconstruction“. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. Ed. by H. Daumé and A. Singh. Vol. 119. PMLR, 2020, pp. 7932–7942.

[RCS20] Z. Ramzi, P. Ciuciu, and J.-L. Starck. „XPDNet for MRI Reconstruction: An Application to the fastMRI 2020 Brain Challenge“. Preprint, arXiv: 2010.07290. 2020.

[Red+16] S. J. Reddi et al. „Stochastic Frank-Wolfe Methods for Nonconvex Optimization“. In: *54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. Institute of Electrical and Electronics Engineers (IEEE), 2016, pp. 1244–1251. DOI: 10.1109/ALLERTON.2016.7852377.

[Řeh11] R. Řehůřek. „Scalability of Semantic Analysis in Natural Language Processing“. PhD thesis. Masaryk University, 2011.

[RFB15] O. Ronneberger, P. Fischer, and T. Brox. „U-Net: Convolutional Networks for Biomedical Image Segmentation“. In: *18th International Conference Medical Image Computing and Computer Assisted Intervention (MICCAI), Proceedings, Part III*. Ed. by N. Navab et al. Springer Cham, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[ROF92] L. I. Rudin, S. Osher, and E. Fatemi. „Nonlinear Total Variation Based Noise Removal Algorithms“. In: *Physica D: Nonlinear Phenomena* 60.1–4 (1992), pp. 259–268. DOI: 10.1016/0167-2789(92)90242-F.

[RSG16] M. T. Ribeiro, S. Singh, and C. 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*. Association for Computing Machinery (ACM), 2016, pp. 1135–1144.

[RSG18] M. T. Ribeiro, S. Singh, and C. Guestrin. „Anchors: High-Precision Model-Agnostic Explanations“. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. Vol. 32. 1. Association for the Advancement of Artificial Intelligence (AAAI), 2018, pp. 1527–1535.

[RZL17] P. Ramachandran, B. Zoph, and Q. V. Le. „Searching for Activation Functions“. Preprint, arXiv: 1710.05941. 2017.

[Sag94] H. Sagan. *Space-Filling Curves*. Ed. by J. H. Ewing, F. W. Gehring, and P. R. Halmos. Spinger, New York, 1994. DOI: 10.1007/978-1-4612-0871-6.

[SAH20] J. Schwab, S. Antholzer, and M. Haltmeier. „Big in Japan: Regularizing Networks for Solving Inverse Problems“. In: *Journal of Mathematical Imaging and Vision* 62.3 (2020), pp. 445–455. DOI: 10.1007/s10851-019-00911-1.

[Sam+17] W. Samek et al. „Evaluating the Visualization of What a Deep Neural Network Has Learned“. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.11 (2017), pp. 2660–2673. DOI: 10.1109/TNNLS.2016.2599820.

[SCC17] S. Sun, C. Chen, and L. Carin. „Learning Structured Weight Uncertainty in Bayesian Neural Networks“. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by A. Singh and J. Zhu. Vol. 54. PMLR, 2017, pp. 1283–1292.

[SCD18] A. Shih, A. Choi, and A. Darwiche. „A Symbolic Approach to Explaining Bayesian Network Classifiers“. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press/IJCAI, 2018, pp. 5103–5111.

[Sch+17] J. Schlemper et al. „A Deep Cascade of Convolutional Neural Networks for Dynamic MR Image Reconstruction“. In: *IEEE Transactions on Medical Imaging* 37.2 (2017), pp. 491–503. DOI: 10.1109/TMI.2017.2760978.

[Sch+19] J. Schlemper et al. „dAUTOMAP: Decomposing AUTOMAP to Achieve Scalability and Enhance Performance“. Preprint, arXiv: 1909.10995. 2019.

[SD91] J. Sietsma and R. J. Dow. „Creating Artificial Neural Networks That Generalize“. In: *Neural Networks* 4.1 (1991), pp. 67–79. DOI: 10.1016/0893-6080(91)90033-2.

[SGK17] A. Shrikumar, P. Greenside, and A. Kundaje. „Learning Important Features Through Propagating Activation Differences“. Preprint, arXiv: 1704.02685. 2017.

[Sha53] L. S. Shapley. „A Value for n-Person Games“. In: *Contributions to the Theory of Games II*. Ed. by H. W. Kuhn and A. W. Tucker. Princeton University Press, 1953, pp. 307–317. DOI: 10.1515/9781400881970-018.

[She+19] Z. Shen et al. „Complexities in Projection-Free Stochastic Non-convex Minimization“. In: *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by K. Chaudhuri and M. Sugiyama. Vol. 89. PMLR, 2019, pp. 2868–2876.

[SHM14] D. Soudry, I. Hubara, and R. Meir. „Expectation Backpropagation: Parameter-Free Training of Multilayer Neural Networks With Continuous or Discrete Weights“. In: *Advances in Neural Information Processing Systems 27 (NIPS)*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[Sid+21a] E. Y. Sidky et al. *Deep Learning for Inverse Problems: Sparse-View Computed Tomography Image Reconstruction (DL-Sparse-View CT)*. 2021. URL: https://www.aapm.org/GrandChallenge/DL-sparse-view-CT/.

[Sid+21b] E. Y. Sidky et al. „Do CNNs Solve the CT Inverse Problem?“ In: *IEEE Transactions on Biomedical Engineering* 68.6 (2021), pp. 1799–1810. DOI: 10.1109/TBME.2020.3020741.

[Sim75] J. Simon. „On Some Central Problems in Computational Complexity.“ PhD thesis. Cornell University, 1975.

[Smi+17] D. Smilkov et al. „SmoothGrad: Removing Noise by Adding Noise“. Preprint, arXiv: 1706.03825. 2017.

[SP08] E. Y. Sidky and X. Pan. „Image Reconstruction in Circular Cone-Beam Computed Tomography by Constrained, Total-Variation Minimization“. In: *Physics in Medicine and Biology* 53.17 (2008), pp. 4777–4807. DOI: 10.1088/0031-9155/53/17/021.

[SP22] E. Y. Sidky and X. Pan. „Report on the AAPM Deep-Learning Sparse-View CT (DL-Sparse-View CT) Grand Challenge“. Preprint, arXiv: 2109.09640. 2022.

[SPM02] J. L. Starck, E. Pantin, and F. Murtagh. „Deconvolution in Astronomy: A Review“. In: *Publications of the Astronomical Society of the Pacific* 114.800 (2002), pp. 1051–1069. DOI: 10.1086/342606.

[Spr+15] J. T. Springenberg et al. „Striving for Simplicity: The All Convolutional Net“. ICLR Workshop Track. 2015.

[Sri+14] N. Srivastava et al. „Dropout: A Simple Way to Prevent Neural Networks from Overfitting“. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[Sri+20] A. Sriram et al. „End-To-End Variational Networks for Accelerated MRI Reconstruction“. In: *23rd International Conference Medical Image Computing and Computer Assisted Intervention (MICCAI), Proceedings, Part II*. Ed. by A. L. Martel et al. Vol. 12262. Lecture Notes in Computer Science. Springer, 2020, pp. 64–73. DOI: 10.1007/978-3-030-59713-9_7.

[STE13] C. Szegedy, A. Toshev, and D. Erhan. „Deep Neural Networks for Object Detection“. In: *Advances in Neural Information Processing Systems 26 (NIPS)*. Ed. by C. J. C. Burges et al. Vol. 26. Curran Associates, Inc., 2013, pp. 2553–2561.

[SVZ13] K. Simonyan, A. Vedaldi, and A. Zisserman. „Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps“. Preprint, arXiv: 1312.6034. 2013.

[SWM17] W. Samek, T. Wiegand, and K. Müller. „Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models“. Preprint, arXiv: 1708.08296. 2017.

[SWS17] D. Shen, G. Wu, and H.-I. Suk. „Deep Learning in Medical Image Analysis“. In: *Annual Review of Biomedical Engineering* 19.1 (2017), pp. 221–248. DOI: 10.1146/annurev-bioeng-071516-044442.

[SZ14] K. Simonyan and A. Zisserman. „Very Deep Convolutional Networks for Large-Scale Image Recognition“. Preprint, arXiv: 1409.1556. 2014.

[Sze+14] C. Szegedy et al. „Intriguing Properties of Neural Networks“. Preprint, arXiv: 1312.6199. 2014.

[TK10] E. Å. Trumbelj and I. Kononenko. „An Efficient Explanation of Individual Classifications Using Game Theory“. In: *Journal of Machine Learning Research* 11.1 (2010), pp. 1–18.

[TV81] A. Tarantola and B. Valetta. „Inverse Problems = Quest for Information“. In: *Journal of Geophysics* 50.1 (1981), pp. 159–170.

[UVL18] D. Ulyanov, A. Vedaldi, and V. Lempitsky. „Deep Image Prior“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE), 2018, pp. 9446–9454. DOI: 10.1109/CVPR.2018.00984.

[Vas+17] A. Vaswani et al. „Attention is All You Need“. In: *Advances in Neural Information Processing Systems 30 (NIPS)*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008.

[Vid+15] M. M.-C. Vidovic et al. „Opening the Black Box: Revealing Interpretable Sequence Motifs in Kernel-Based Learning Algorithms“. In: *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Vol. 2. Springer, Cham, 2015, pp. 137–153.

[Vin+10] P. Vincent et al. „Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network With a Local Denoising Criterion“. In: *Journal of Machine Learning Research* 11.110 (2010), pp. 3371–3408.

[VS18] A. Virmaux and K. Scaman. „Lipschitz Regularity of Deep Neural Networks: Analysis and Efficient Estimation“. In: *Advances in Neural Information Processing Systems 31 (NeurIPS)*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018, pp. 3839–3848.

[Wäl+21] S. Wäldchen et al. „The Computational Complexity of Understanding Binary Classifier Decisions“. In: *Journal of Artificial Intelligence Research* 70 (2021), pp. 351–387. DOI: 10.1613/jair.1.12359.

[Wan+04] Z. Wang et al. „Image Quality Assessment: From Error Visibility to Structural Similarity“. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[WH18] Y. Wu and K. He. „Group Normalization“. In: *Computer Vision – ECCV 2018, Proceedings, Part XIII*. Springer, Cham, 2018, pp. 3–19. DOI: 10.1007/978-3-030-01261-8_1.

[Wil94] M. R. William T. Freeman. *Orientation Histograms for Hand Gesture Recognition*. Tech. rep. TR94-03. MERL - Mitsubishi Electric Research Laboratories, 1994.

[WM13] S. Wang and C. Manning. „Fast Dropout Training“. In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. 2. PMLR, 2013, pp. 118–126.

[WN19] M. J. Willemink and P. B. Noël. „The Evolution of Image Reconstruction for CT – From Filtered Back Projection to Artificial Intelligence“. In: *European Radiology* 29.5 (2019), pp. 2185–2195. DOI: 10.1007/s00330-018-5810-7.

[Wol+17] J. M. Wolterink et al. „Generative Adversarial Networks for Noise Reduction in Low-Dose CT“. In: *IEEE Transactions on Medical Imaging* 36.12 (2017), pp. 2536–2545. DOI: 10.1109/TMI.2017.2708987.

[Wol70] P. Wolfe. „Convergence theory in nonlinear programming“. In: *Integer and Nonlinear Programming*. Ed. by J. Abadie. North-Holland, 1970, pp. 1–36.

[Wu+19] A. Wu et al. „Deterministic Variational Inference for Robust Bayesian Neural Networks“. In: *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[Wür+16] T. Würfl et al. „Deep Learning Computed Tomography“. In: *19th International Conference Medical Image Computing and Computer-Assisted Intervention (MICCAI), Proceedings, Part III*. Springer, Cham, 2016, pp. 432–440. DOI: 10.1007/978-3-319-46726-9_50.

[Yam+90] K. Yamaguchi et al. „A Neural Network for Speaker-Independent Isolated Word Recognition“. In: *Proceedings of the 1st International Conference on Spoken Language Processing (ICSLP)*. International Speech Communication Association (ISCA), 1990, pp. 1077–1080.

[Yan+16] Y. Yang et al. „Deep ADMM-Net for Compressive Sensing MRI“. In: *Advances in Neural Information Processing Systems 29 (NIPS)*. Ed. by D. D. Lee et al. Vol. 29. Curran Associates, Inc., 2016, pp. 10–18.

[Yan+18] Q. Yang et al. „Low-dose CT Image Denoising Using a Generative Adversarial Network With Wasserstein Distance and Perceptual Loss“. In: *IEEE Transactions on Medical Imaging* 37.6 (2018), pp. 1348–1357. DOI: 10.1109/TMI.2018.2827462.

[YSC19] A. Yurtsever, S. Sra, and V. Cevher. „Conditional Gradient Methods via Stochastic Path-Integrated Differential Estimator“. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. PMLR, 2019, pp. 7282–7291.

[Yua+19] X. Yuan et al. „Adversarial Examples: Attacks and Defenses for Deep Learning“. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.9 (2019), pp. 2805–2824. DOI: 10.1109/TNNLS.2018.2886017.

[Zbo+18] J. Zbontar et al. „fastMRI: An Open Dataset and Benchmarks for Accelerated MRI“. Preprint, arXiv: 1811.08839. 2018.

[ZF14] M. D. Zeiler and R. Fergus. „Visualizing and Understanding Convolutional Networks“. In: *Computer Vision – ECCV 2014, Proceedings, Part I*. Ed. by D. Fleet et al. Springer, Cham, 2014, pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53.

[Zha+17] H. Zhao et al. „Loss Functions for Image Restoration With Neural Networks“. In: *IEEE Transactions on Computational Imaging* 3.1 (2017), pp. 47–57. DOI: 10.1109/TCI.2016.2644865.

[Zha+18] Z. Zhang et al. „Opening the Black Box of Neural Networks: Methods for Interpreting Neural Network Models in Clinical Applications“. In: *Annals of Translational Medicine* 6.11 (2018), pp. 1–11. DOI: 10.21037/atm.2018.05.32.

[Zha+21] Y. Zhang et al. „A Survey on Neural Network Interpretability“. Preprint, arXiv: 2012.14261. 2021.

[Zhu+18] B. Zhu et al. „Image Reconstruction by Domain-Transform Manifold Learning“. In: *Nature* 555.7697 (2018), pp. 487–492. DOI: 10.1038/nature25988.

III

Appendices

A

Deferred Proofs of Chapter 3

A.1 Raising the Probability Threshold

We give constructive proofs of Lemmas 3.18 and 3.22, starting with the first.

Proof of Lemma 3.18. Let $\Psi\colon \{0,1\}^n \to \{0,1\}$ be arbitrary and $0 \leq \delta_1 < \delta_2 < 1$. We will construct a monotone function $\Pi = \Pi^{\delta_2}_{\delta_1,n}\colon \{0,1\}^d \to \{0,1\}$ such that

$$\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z})] > \delta_1 \quad \Longleftrightarrow \quad \mathbb{P}_{(\mathbf{z},\mathbf{r})}[\Psi(\mathbf{z}) \vee \Pi(\mathbf{r})] \geq \delta_2, \tag{A.1}$$

with

$$d \in O\left(\left(n + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right)^2\right).$$

In our context δ_1 and δ_2 are considered as constant and therefore $d \in O(n^2)$.

Denote $\Psi'\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}\colon (\mathbf{z},\mathbf{r}) \mapsto \Psi(\mathbf{z}) \vee \Pi(\mathbf{r})$, then

$$\mathbb{P}[\Psi'] = \mathbb{P}[\Psi] + (1 - \mathbb{P}[\Psi])\mathbb{P}[\Pi], \tag{A.2}$$

which is monotonically increasing in both $\mathbb{P}[\Psi]$ and $\mathbb{P}[\Pi]$. Thus, it suffices to consider the edge case when $\mathbb{P}[\Psi]$ is close to δ_1. Since $\mathbb{P}[\Psi]$ can only take values in $\left\{\frac{0}{2^n}, \frac{1}{2^n}, \ldots, \frac{2^n}{2^n}\right\}$ we see that (A.1) is equivalent to the two conditions

$$\begin{aligned}
\mathbb{P}[\Psi] = \frac{\lfloor \delta_1 2^n \rfloor}{2^n} \quad &\Longrightarrow \quad \mathbb{P}[\Psi'] < \delta_2, \\
\mathbb{P}[\Psi] = \frac{\lfloor \delta_1 2^n \rfloor + 1}{2^n} \quad &\Longrightarrow \quad \mathbb{P}[\Psi'] \geq \delta_2,
\end{aligned}$$

which together with (A.2) is equivalent to

$$\frac{\lfloor \delta_1 2^n \rfloor}{2^n} + \frac{2^n - \lfloor \delta_1 2^n \rfloor}{2^n}\mathbb{P}[\Pi] < \delta_2 \tag{A.3}$$

$$\frac{\lfloor \delta_1 2^n \rfloor + 1}{2^n} + \frac{2^n - \lfloor \delta_1 2^n \rfloor - 1}{2^n}\mathbb{P}[\Pi] \geq \delta_2. \tag{A.4}$$

In the case $\delta_1 < \delta_2 \leq \frac{\lfloor \delta_1 2^n \rfloor + 1}{2^n}$ these conditions are already fulfilled if we simply set $\Pi \equiv 0$. Otherwise, if $\delta_2 > \frac{\lfloor \delta_1 2^n \rfloor + 1}{2^n}$, rearranging (A.3) and (A.4) yields the bounds

$$a \leq \mathbb{P}[\Pi] < b$$

on $\mathbb{P}[\Pi]$, where

$$a = \frac{\delta_2 2^n - \lfloor \delta_1 2^n \rfloor - 1}{2^n - \lfloor \delta_1 2^n \rfloor - 1},$$
$$b = \frac{\delta_2 2^n - \lfloor \delta_1 2^n \rfloor}{2^n - \lfloor \delta_1 2^n \rfloor}.$$

It is not hard to check that indeed we have $0 \leq a < b \leq 1$.

In Appendix A.4 we show for $\eta \in [0,1]$ and $\ell \in \mathbb{N}$ the existence of a monotone DNF-function $\Pi_{\eta,\ell} \colon \{0,1\}^d \to \{0,1\}$ such that $\Pi_{\eta,\ell}(\mathbf{0}_d) = 0$, $\Pi_{\eta,\ell}(\mathbf{1}_d) = 1$, and

$$|\mathbb{P}[\Pi_{\eta,\ell}] - \eta| \leq 2^{-\ell}$$

with $d \leq \frac{\ell(\ell+3)}{2} \in O(\ell^2)$. We conclude by choosing

$$\eta = \frac{b+a}{2},$$
$$\ell = \left\lfloor -\log_2\left(\frac{b-a}{2}\right)\right\rfloor + 1 \in O\left(n + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right),$$

and setting $\Pi = \Pi_{\eta,\ell}$. We get $d \in O(\ell^2) = O\left(\left(n + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right)^2\right)$, which finishes the proof of Lemma 3.18. □

Proof of Lemma 3.22. We proceed analogously to before. For $0 \leq \delta_1 \leq \delta_2 < 1$, we construct a monotone function $\Pi = \Pi^{\delta_2}_{\delta_1,n} \colon \{0,1\}^d \to \{0,1\}$ such that

$$\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z})] \geq \delta_1 \iff \mathbb{P}_{(\mathbf{z},\mathbf{r})}[\Psi(\mathbf{z}) \vee \Pi(\mathbf{r})] > \delta_2,$$

with

$$d \in O\left(\left(d + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right)^2\right).$$

Again, δ_1 and δ_2 are considered constant in our setting and therefore $d \in O(n^2)$.

Similar to before, in the case that $\delta_1 \leq \delta_2 < \frac{\lceil \delta_1 2^n \rceil}{2^n}$, we can simply set $\Pi \equiv 0$. Otherwise, we get the bounds

$$a < \mathbb{P}[\Pi] \leq b$$

with

$$a = \frac{\delta_2 2^n - \lceil \delta_1 2^n \rceil}{2^n - \lceil \delta_1 2^n \rceil},$$
$$b = \frac{\delta_2 2^n - \lceil \delta_1 2^n \rceil + 1}{2^n - \lceil \delta_1 2^n \rceil + 1}.$$

Again, we can check that $0 \leq a < b \leq 1$, and set

$$\eta = \frac{b+a}{2},$$
$$\ell = \left\lfloor -\log_2\left(\frac{b-a}{2}\right)\right\rfloor + 1 \in O\left(n + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right),$$

and $\Pi = \Pi_{\eta,\ell}$ with $d \in O(\ell^2) = O\left(\left(n + \log_2\left(\frac{1-\delta_1}{1-\delta_2}\right)\right)^2\right)$, which concludes the proof of Lemma 3.22. □

A.2 Lowering the Probability Threshold

We give constructive proofs of Lemmas 3.19 and 3.23, starting with the first.

Proof of Lemma 3.19. Let $\Psi\colon \{0,1\}^n \to \{0,1\}$ be arbitrary and $0 < \delta_1 \leq \delta_2 \leq 1$. We will construct a monotone function $\Pi = \Pi^{\delta_2}_{\delta_1,n}\colon \{0,1\}^d \to \{0,1\}$ such that

$$\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z})] > \delta_2 \quad \Longleftrightarrow \quad \mathbb{P}_{(\mathbf{z},\mathbf{r})}[\Psi(\mathbf{z}) \wedge \Pi(\mathbf{r})] \geq \delta_1, \tag{A.5}$$

with

$$d \in O\left(\left(n + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right)^2\right).$$

In our context δ_1 and δ_2 are considered constant and therefore $d \in O(n^2)$.

Denote $\Psi'\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}\colon (\mathbf{z},\mathbf{r}) \mapsto \Psi(\mathbf{z}) \wedge \Pi(\mathbf{r})$, then

$$\mathbb{P}[\Psi'] = \mathbb{P}[\Psi]\mathbb{P}[\Pi], \tag{A.6}$$

which is monotonically increasing in both $\mathbb{P}[\Psi]$ and $\mathbb{P}[\Pi]$. Thus, it suffices to consider the edge case when $\mathbb{P}[\Psi]$ is close to δ_2. Since $\mathbb{P}[\Psi]$ can only take values in $\left\{\frac{0}{2^n}, \frac{1}{2^n}, \ldots, \frac{2^n}{2^n}\right\}$ we see that (A.5) is equivalent to the two conditions

$$\begin{aligned} \mathbb{P}[\Psi] = \frac{\lfloor \delta_2 2^n \rfloor}{2^n} \quad &\Longrightarrow \quad \mathbb{P}[\Psi'] < \delta_1, \\ \mathbb{P}[\Psi] = \frac{\lfloor \delta_2 2^n \rfloor + 1}{2^n} \quad &\Longrightarrow \quad \mathbb{P}[\Psi'] \geq \delta_1, \end{aligned}$$

which together with (A.6) is equivalent to

$$\frac{\lfloor \delta_2 2^n \rfloor}{2^n}\mathbb{P}[\Pi] < \delta_1 \tag{A.7}$$

$$\frac{\lfloor \delta_2 2^n \rfloor + 1}{2^n}\mathbb{P}[\Pi] \geq \delta_1. \tag{A.8}$$

In the case $\frac{\lfloor \delta_2 2^n \rfloor}{2^n} < \delta_1 \leq \delta_2$ these conditions are already fulfilled if we simply set $\Pi \equiv 1$. Otherwise, if $\delta_1 \leq \frac{\lfloor \delta_2 2^n \rfloor}{2^n}$, rearranging (A.7) and (A.8) yields the bounds

$$a < \mathbb{P}[\Pi] \leq b$$

on $\mathbb{P}[\Pi]$, where

$$\begin{aligned} a &= \frac{\delta_1 2^n}{\lfloor \delta_2 2^n \rfloor + 1}, \\ b &= \frac{\delta_1 2^n}{\lfloor \delta_2 2^n \rfloor}. \end{aligned}$$

It is not hard to check that indeed we have $0 \leq a < b \leq 1$.

In Appendix A.4, we show for $\eta \in [0,1]$ and $\ell \in \mathbb{N}$ the existence of a monotone DNF-function $\Pi_{\eta,\ell} \colon \{0,1\}^d \to \{0,1\}$ such that $\Pi_{\eta,\ell}(\mathbf{0}_d) = 0$, $\Pi_{\eta,\ell}(\mathbf{1}_d) = 1$, and

$$|\mathbb{P}[\Pi_{\eta,\ell}] - \eta| \leq 2^{-\ell}$$

with $d \leq \frac{\ell(\ell+3)}{2} \in O(\ell^2)$. We conclude by choosing

$$\eta = \frac{b+a}{2},$$
$$\ell = \left\lfloor -\log_2\left(\frac{b-a}{2}\right)\right\rfloor + 1 \in O\left(n + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right),$$

and setting $\Pi = \Pi_{\eta,\ell}$. We get $d \in O(\ell^2) = O\left(\left(n + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right)^2\right)$, which finishes the proof of Lemma 3.19. □

Proof of Lemma 3.23. We proceed analogously to before. For $0 < \delta_1 < \delta_2 \leq 1$, we construct a monotone function $\Pi = \Pi^{\delta 2}_{\delta_1,n} \colon \{0,1\}^d \to \{0,1\}$ such that

$$\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z})] \geq \delta_2 \quad \Longleftrightarrow \quad \mathbb{P}_{(\mathbf{z},\mathbf{r})}[\Psi(\mathbf{z}) \wedge \Pi(\mathbf{r})] > \delta_1,$$

with

$$d \in O\left(\left(n + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right)^2\right).$$

Again, δ_1 and δ_2 are considered constant in our setting and therefore $d \in O(n^2)$.

Similar to before, in case that $\frac{\lceil \delta_2 2^n \rceil - 1}{2^n} \leq \delta_1 < \delta_2$, we can simply set $\Pi \equiv 1$. Otherwise, we get the bounds

$$a < \mathbb{P}[\Pi] \leq b$$

with

$$a = \frac{\delta_1 2^n}{\lceil \delta_2 2^n \rceil}$$
$$b = \frac{\delta_1 2^n}{\lceil \delta_2 2^n \rceil - 1}.$$

Again, we can check that $0 \leq a < b \leq 1$, and set

$$\eta = \frac{b+a}{2},$$
$$\ell = \left\lfloor -\log_2\left(\frac{b-a}{2}\right)\right\rfloor + 1 \in O\left(n + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right),$$

and $\Pi = \Pi_{\eta,\ell}$ with $d \in O(\ell^2) = O\left(\left(n + \log_2\left(\frac{\delta_2}{\delta_1}\right)\right)^2\right)$, which concludes the proof of Lemma 3.23. □

A.3 Neutral Operation

We provide a constructive proof of Lemma 3.20.

Proof of Lemma 3.20. Let $\Psi\colon \{0,1\}^n \to \{0,1\}$ be arbitrary and $0 < \delta < 1$. We will construct a function $\Gamma = \Gamma_{\delta,n}\colon \{0,1\}^d \to \{0,1\}$ so that for some $T_{\delta,n} \in \mathbb{N}$ we have

$$\mathbb{P}_{\mathbf{z}}[\Psi(\mathbf{z})] \geq \delta \quad \Longleftrightarrow \quad \mathbb{P}_{(\mathbf{z},\mathbf{r},\mathbf{t})}\left[(\Psi(\mathbf{z}) \wedge \Gamma(\mathbf{r})) \vee \left(\bigwedge_{j=1}^{T} t_j\right)\right] \geq \delta, \tag{A.9}$$

for all $T \geq T_{\delta,n}$ and

$$d + T_{\delta,n} \in O\left(\log\left(\frac{1}{\delta}\right) + n^2\right).$$

We introduce $\Psi' = \Psi \wedge \Gamma$ and $\Psi'' = \Psi' \vee (\bigwedge_{j=1}^{T} t_j)$. Let us distinguish three cases

$$\begin{aligned}
&\textbf{Case I:} && \delta \leq 2^{-n}, \\
&\textbf{Case II:} && \delta > 2^{-n} \quad \text{and} \quad \lceil \delta 2^n \rceil - \delta 2^n \geq \frac{2}{3}, \\
&\textbf{Case III:} && \delta > 2^{-n} \quad \text{and} \quad \lceil \delta 2^n \rceil - \delta 2^n < \frac{2}{3}.
\end{aligned}$$

Let us begin with the construction for the first case. Here, we see that $\mathbb{P}[\Psi] < \delta$ is equivalent to $\mathbb{P}[\Psi] = 0$. We simply set $\Gamma \equiv 1$ and $T_{\delta,n} = \lceil \log(\frac{1}{\delta}) \rceil + 1$. It is easy to check that this satisfies (A.9).

Next, for the second case, we want to construct Γ such that

$$\mathbb{P}[\Psi] = \frac{\lceil \delta 2^n \rceil}{2^n} \quad \Longrightarrow \quad \mathbb{P}[\Psi'] \geq \delta, \tag{A.10}$$

$$\mathbb{P}[\Psi] = \frac{\lceil \delta 2^n \rceil - 1}{2^n} \quad \Longrightarrow \quad \mathbb{P}[\Psi'] < \delta - \frac{1}{3}2^{-n}, \tag{A.11}$$

which results in the condition

$$a \leq \mathbb{P}[\Gamma] < b$$

with

$$\begin{aligned}
a &= \frac{\delta 2^n}{\lceil \delta 2^n \rceil} \\
b &= \frac{\delta 2^n - \frac{1}{3}}{\lceil \delta 2^n \rceil - 1}.
\end{aligned}$$

Thus we can set $\Gamma = \Pi_{\eta,\ell}$ according to Appendix A.4 with $\eta = \frac{b+a}{2}$ and $\ell = \lfloor \log(\frac{2}{b-a}) \rfloor + 1$. Using the fact that $\delta > 2^{-n}$ and hence

$$b - a = \frac{\delta 2^n - \frac{1}{3}\lceil \delta 2^n \rceil}{\lceil \delta 2^n \rceil (\lceil \delta 2^n \rceil - 1)} \geq \frac{1}{3}2^{-2n},$$

we obtain $\ell \leq 2n + \lfloor \log(6) \rfloor + 1 = 2n + 3$. From Appendix A.4 we known that $d \in O(\ell^2)$ and thus $d \in O(n^2)$. We continue to construct Ψ'' by choosing $T_{\delta,n}$ such that

$$\begin{aligned} \mathbb{P}[\Psi'] \geq \delta \quad &\Longrightarrow \quad \mathbb{P}[\Psi''] \geq \delta, \\ \mathbb{P}[\Psi'] < \delta - \frac{1}{3}2^{-d} \quad &\Longrightarrow \quad \mathbb{P}[\Psi''] < \delta, \end{aligned}$$

holds for all $T \geq T_{\delta,n}$. The first condition is automatically fulfilled. From

$$\mathbb{P}[\Psi''] = \mathbb{P}[\Psi'] + (1 - \mathbb{P}[\Psi'])2^{-T},$$

as well as $(1 - \mathbb{P}[\Psi']) \leq 1$ we observe that

$$2^{-T} < \frac{1}{3}2^{-n}$$

is sufficient for the other condition. Thus, we choose $T_{\delta,n} = n + \lfloor \log(3) \rfloor + 1 = n + 2$.

Finally, for the third case, we again want to construct Γ so that (A.10) and (A.11) hold. Here, this is already satisfied by setting $\Gamma \equiv 1$. We continue analogously as in the second case and choose the same $T_{\delta,n}$. □

A.4 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 $\Pi_{\eta,\ell} \colon \{0,1\}^d \to \{0,1\}$ in disjunctive normal form with $d \in O(\ell^2)$, $\Pi_{\eta,\ell}(\mathbf{0}_d) = 0$, $\Pi_{\eta,\ell}(\mathbf{1}_d) = 1$, and

$$|\eta - \mathbb{P}[\Pi_{\eta,\ell}]| \leq 2^{-\ell}.$$

If $\eta \leq 2^{-\ell}$, we can simply choose $\Pi_{\eta,\ell}(z_1, \dots, z_\ell) = \bigwedge_{j=1}^{\ell} z_j$. So from now on assume $2^{-\ell} < \eta \leq 1$. In this case we construct a sequence of functions $\Pi_i \colon \{0,1\}^{d_i} \to \{0,1\}$ such that $p_i = \mathbb{P}[\Pi_i]$ 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 $d_{i+1} = d_i + \Delta d_i$ with

$$\Delta d_i = \operatorname{argmin}\{\, d \in \mathbb{N} \,:\, p_i + (1 - p_i)2^{-d} \leq \eta \,\}, \tag{A.12}$$

and

$$\Pi_{i+1}(z_1, \dots, z_{d_{i+1}}) = \Pi_i(z_1, \dots, z_{d_i}) \vee \left(\bigwedge_{j=d_i+1}^{d_{i+1}} z_j \right).$$

Clearly, we obtain $p_{i+1} = p_i + (1 - p_i)2^{-\Delta d_i}$. We will see below that Δd_i can not be too large and thus (A.12) can be efficiently computed by sequential search.

Lemma A.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 Δd_i, we have $p_i \leq \eta$ for all $i \in \mathbb{N}$. Also from (A.12) we know that $p_i + (1 - p_i)2^{-(\Delta d_i - 1)} > \eta$ since otherwise Δd_i would be chosen smaller. Therefore,

$$\begin{aligned}\eta - p_{i+1} &= \eta - p_i - (1 - p_i)2^{-\Delta d_i} \\ &= \eta - \frac{1}{2}p_i - \frac{1}{2}(p_i + (1 - p_i)2^{-(\Delta d_i - 1)}) \\ &\leq \frac{1}{2}(\eta - p_i).\end{aligned}$$

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 A.2. *For any $i \in \mathbb{N}$, we have $\Delta d_i < -\log_2(\eta - p_i) + 1$.*

Proof. As before we know $p_i + (1 - p_i)2^{-(\Delta d_i - 1)} > \eta$ since otherwise Δd_i would be chosen smaller. This implies

$$2^{-(\Delta d_i - 1)} > \frac{\eta - p_i}{1 - p_i} \geq \eta - p_i,$$

and therefore $\Delta d_i < -\log_2(\eta - p_i) + 1$. □

This can finally be used to bound how many variables are used in total.

Lemma A.3. *The total number of variables for $\Pi_{\eta,\ell} = \Pi_\ell$ is*

$$d = d_\ell = \sum_{i=1}^{\ell} \Delta d_{i-1} \in O(\ell^2).$$

Proof. From Lemma A.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 A.2, this immediately results in

$$\begin{aligned} d &= \sum_{i=1}^{\ell} \Delta d_{i-1} \\ &\leq \sum_{i=1}^{\ell} -\log_2(\eta - p_{i-1}) + 1 \\ &\leq \sum_{i=1}^{\ell} -\log_2(2^{\ell-i}(\eta - p_{\ell-1})) + 1 \\ &\leq \sum_{i=1}^{\ell} -\log_2(2^{-i}) + 1 \\ &= \frac{\ell(\ell+1)}{2} + \ell \in O(\ell^2). \end{aligned}$$

□

B

Additions to Chapter 4

B.1 Algorithm Descriptions

The RDE variants (L-RDE), (RC-RDE), and (Ord-RDE) of Section 4.3 result in constrained optimization problems with non-convex objectives over convex and compact domains $C \subseteq \mathbb{R}^n$. We will now describe two possibilities for numerically solving such problems of the form

$$\begin{aligned} &\text{minimize} && D(\mathbf{s}) \\ &\text{subject to} && \mathbf{s} \in C. \end{aligned}$$

B.1.1 Projected Gradient Descent

Variants of Gradient Descent (GD) are by far the most popular optimization methods when working with neural networks and can also be used for RDE. [1] Incorporating the constraint $\mathbf{s} \in C$ can be achieved through Projected Gradient Descent (PGD). This requires a projection step

$$\mathbf{s}_{t+1} = \text{proj}_C\left(\mathbf{s}_t - \eta_t \nabla D(\mathbf{s}_t)\right)$$

with step size $\eta_t > 0$ for each update in order to maintain feasible iterates. Here $\text{proj}_C(\cdot)$ denotes the Euclidean projection

$$\text{proj}_C(\mathbf{x}) = \underset{\mathbf{v} \in C}{\text{argmin}} \|\mathbf{v} - \mathbf{x}\|_2.$$

Depending on the feasible region C projections can be challenging and computationally costly. Algorithm 1 and Algorithm 2 show PGD for the (L-RDE) and (RC-RDE) problems respectively. In contrast, PGD can not directly be used for the (Ord-RDE) problem, since no exact projection method specific to its feasible region (the Birkhoff polytope) is known, see Appendix B.1.3.

B.1.2 Frank-Wolfe Algorithms

An alternative projection-free first-order method is the Frank-Wolfe (FW) algorithm [FW56] or Conditional Gradient method [LP66]. At its core, the Frank-Wolfe algorithm solves a Linear Minimization Oracle (LMO)

$$\mathbf{v}_t = \underset{\mathbf{v} \in C}{\text{argmin}} \left\langle \nabla D(\mathbf{s}_t), \mathbf{v} \right\rangle, \tag{B.1}$$

[1] More generally, proximal methods can be considered for non-smooth problems, e.g., including an ℓ_1-norm sparsity penalty as in (L-RDE).

over the feasible region C and then moves in direction $\mathbf{v}_t$ via the update

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \eta_t(\mathbf{v}_t - \mathbf{s}_t),$$

with step size $\eta_t \in [0, 1]$. Feasibility is maintained for convex regions C since the new iterate is a convex combination of two feasible points $\mathbf{s}_t$ and $\mathbf{v}_t$. Since its first appearance, several algorithmic variations have been developed that enhance the performance of the original algorithm, while maintaining many of its advantages. They offer potential improvements over vanilla Frank-Wolfe in terms of iteration count or runtime in certain specialized settings. In our experiments, we consider vanilla Frank-Wolfe (FW), Away-Step Frank-Wolfe (AFW), Lazified Conditional Gradients (LCG), Lazified Away-Step Frank-Wolfe (LAFW), and Stochastic Frank-Wolfe (SFW). The interested reader is referred to a more detailed presentation of the algorithm variants and their implementation in [BCP21].

Vanilla Frank-Wolfe (FW). In its basic version, see Algorithm 3, a linear approximation to the objective function at the current iterate (obtained from first-order information) is minimized over the feasible region. Such a Linear Minimization Oracle (LMO) is often computationally less costly than a corresponding projection (which amounts to solving a quadratic problem). The next FW iterate is obtained as a convex combination of the solution to the LMO and the current iterate. For convex regions C, this guarantees that the algorithm produces iterates that remain feasible throughout all iterations. This basic variant has the lowest memory requirements among all deterministic variants since it only requires keeping track of the current iterate. Hence, it is well suited for large-scale problems. However, other variants can achieve improvements in terms of convergence speed (iteration count and time for more specialized setups).

Stochastic Frank-Wolfe (SFW). In some cases, evaluating the full objective function and its gradients is expensive, but cheaper unbiased estimators are available. The typical example is that of objective functions that are sums of a large number of terms, such as in the (Ord-RDE) formulation. An estimator is given by evaluating the sum only over a randomly chosen subset of terms. The stochastic version of Frank-Wolfe [HL16] developed in Algorithm 4 uses a gradient estimate instead of the exact gradient in combination with a momentum term [MHK20] to build the linear approximation to the objective in each iteration. Then the LMO is evaluated and a step is taken exactly as in the vanilla FW algorithm. Different variants of SFW have recently been studied also in the non-convex setting [YSC19; She+19; Has+20; Nég+20].

Away-Step Frank-Wolfe (AFW). While the vanilla Frank-Wolfe algorithm can only move *toward* an extreme point of the feasible set (solution of the LMO at the current iterate), the Away-Step Frank-Wolfe algorithm [Wol70; GM86; LJ15] is allowed to move *away* from some extreme points. More specifically, it maintains an *active set* of extreme points used in the previous iterations as well as a convex decomposition of the current iterate in terms of the active set. At each iteration either a standard FW step toward a new extreme point or a step away from an extreme point in the active set is taken, whichever promises a better decrease in the objective function. This can result in faster convergence (in terms of iteration count and time) but requires additional memory to store the active set.

Lazified Conditional Gradients (LCG). In some cases the evaluation of the LMO might be costly (even if it is still cheaper than a corresponding projection). In such a setting, the idea of *lazy* FW steps can help to avoid unnecessary evaluations of the LMO. Instead of exactly solving the LMO subproblem, an approximate solution that guarantees enough progress is used [BPZ19]. In other words, the LMO can be replaced by a weak separation oracle [BPZ19], i.e., an oracle returning an extreme point with sufficient decrease of the linear objective or a certificate that such a point does not exist. More precisely, the algorithm maintains a cache of previous extreme points and at each iteration searches the cache for a direction that provides sufficient progress. If this is not possible, a new extreme point is obtained via the LMO and added to the cache. The lazification can result in increased performance (due to fewer LMO evaluations) but requires additional memory to store the cache of previous extreme points.

Lazified Away-Step Frank-Wolfe (LAFW). LAFW uses the same idea of a weak separation oracle as in LCG. The search for an appropriate direction providing sufficient progress is carried out over the active set of AFW.

Parameter Choices. The basic step size rule

$$\eta_t = \frac{1}{\sqrt{t+1}} \tag{B.2}$$

can be used for non-convex objectives [Red+16; Com21]. An adaptive step-size choice similar to one proposed by [CBP21] is

$$\eta_t = \frac{2^{-r_t}}{\sqrt{t+1}} \tag{B.3}$$

where $r_t \in \mathbb{N}$ is found by repeated increments starting from r_{t-1} until primal progress is made. This ensures monotonicity in the objective, which is not necessarily the case for the basic rule (B.2). In our experiments, we use the monotone rule (B.3) for FW, AFW, LCG, and LAFW and a corresponding (rescaled) variant also for PGD. Enforcing monotonicity does not make sense in the stochastic setting and we use the basic rule (B.2) for SFW. We test multiple configurations of SFW with constant or increasing batch sizes and momentum factors as proposed by Hazan and Luo [HL16] and Mokhtari, Hassani, and Karbasi [MHK20] respectively. A comparison can be found in Figure B.8. In all other experiments we only show the best performing configuration with constant batch size $b_t = 40$ and no momentum, i.e., $\rho_t = 0$. In all cases, we terminate after a maximal number of $T = 2000$ iterations or if the dual gap $\langle \mathbf{s}_t - \mathbf{v}_t, \nabla D(\mathbf{s}_t) \rangle$ (respectively $\langle \mathbf{\Pi}_t - \mathbf{V}_t, \mathbf{M}_t \rangle$ for SFW) drops below the prescribed threshold $\varepsilon = 10^{-7}$.

B.1.3 Feasible Regions, Projections, and Linear Minimization Oracles

Different feasible regions are of importance for the different RDE variants. For each, we give a brief definition and description of the corresponding Linear Minimization Oracle and Euclidean projection. In the following we denote the component-wise clipping of a

vector $\mathbf{s}$ to lower and upper bounds given by vectors $\boldsymbol{\ell}$ and $\mathbf{u}$ as

$$(\text{clip}(\mathbf{s}, \boldsymbol{\ell}, \mathbf{u}))_j = \begin{cases} \ell_j, & \text{if } s_j \leq \ell_j \\ s_j, & \text{if } \ell_j < s_j < u_j \\ u_j, & \text{if } s_j \geq u_j. \end{cases}$$

Unit Cube. The *unit cube* in $\mathbb{R}^n$ is the set $[0,1]^n$.

LMO: A valid solution $\mathbf{v}$ to (B.1) for the unit cube is given by the vector with components $v_j = 1$ if $(\nabla D(\mathbf{s}))_j < 0$ and $v_j = 0$ otherwise. More generally, for a box with component-wise lower and upper bounds given by vectors $\boldsymbol{\ell}$ and $\mathbf{u}$ a LMO solution is $v_j = \ell_j$ if $(\nabla D(\mathbf{s}))_j > 0$, $v_j = u_j$ if $(\nabla D(\mathbf{s}))_j < 0$, and $v_j \in [\ell_j, u_j]$ if $(\nabla D(\mathbf{s}))_j = 0$. The complexity of finding this solution is $O(n)$.

Projection: Projections onto the unit cube are given by the clipping function $\text{clip}(\mathbf{s}, \mathbf{0}_n, \mathbf{1}_n)$. More generally, for a box with component-wise lower and upper bounds given by vectors $\boldsymbol{\ell}$ and $\mathbf{u}$ the projection is $\text{clip}(\mathbf{s}, \boldsymbol{\ell}, \mathbf{u})$. The complexity of finding the projection is $O(n)$.

The feasible region $C = [0,1]^n$ for the (L-RDE) problem is the unit cube.

k-Sparse Polytope. For $k \in [n]$, the *k-sparse polytope of radius $\tau > 0$* is the intersection of the closed ℓ_1-ball $B_1(\tau k)$ of radius τk and the closed ℓ_∞-ball (hypercube) $B_\infty(\tau)$ of radius τ. It is the convex hull of vectors in $\mathbb{R}^n$ with exactly k non-zero entries, each taking the values τ or $-\tau$.

LMO: A valid solution $\mathbf{v}$ to (B.1) for the k-sparse polytope is given by the vector with exactly k non-zero entries at the components where $|\nabla D(\mathbf{s})|$ takes its k largest values. If v_j is such a non-zero entry then it is equal to $-\tau \operatorname{sign}((\nabla D(\mathbf{s}))_j)$. The complexity of finding this solution is $O(n \log k)$.

Projection: Gupta, Kumar, and Xiao proposed an $O(n)$ algorithm for projections onto ℓ_1-balls with box-constraints [GKX10], extending the work of Duchi et al. on efficient projections onto ℓ_1-balls [Duc+08]. It is based on linear time median finding, see e.g., [Cor+09].

More relevant to us is the following variation:

Non-Negative k-Sparse Polytope. For $k \in [n]$ and $\tau > 0$ the *non-negative k-sparse polytope of radius τ* is defined as the intersection of the k-sparse polytope of radius τ with the non-negative orthant $\mathbb{R}^n_{\geq 0}$.

LMO: A valid solution $\mathbf{v}$ to (B.1) for the non-negative k-sparse polytope is given by the vector with at most k non-zero entries at the components where $\nabla D(\mathbf{s})$ is negative and takes its k smallest values (thus largest in magnitude as above). If $\nabla D(\mathbf{s})$ has fewer than k negative entries, then $\mathbf{v}$ has fewer than k non-zero entries. If v_j is a non-zero entry then it is equal to τ. This is presented in Algorithm 6. As above the complexity of finding this solution is $O(n \log k)$.

Projection: The same $O(n)$ algorithm for projections onto ℓ_1-balls with box-constraints as above can be used. A slightly simpler $O(n \log n)$ variant based on sorting is presented in Algorithm 5 and used in our experiments.

The feasible region $C = \{\mathbf{s} \in [0,1]^n : \|\mathbf{s}\|_1 \leq k\}$ for the (RC-RDE) problem is the non-negative k-sparse polytope of radius $\tau = 1$.

Birkhoff Polytope. The *Birkhoff polytope* B_n is the set of doubly-stochastic $(n \times n)$ matrices. It is the convex hull $\operatorname{conv}(S_n)$ of the set S_n of $(n \times n)$ permutation matrices.

LMO: The Birkhoff polytope arises in matching and ranking problems. Linear minimization over B_n results in a linear program, which can be solved with $O(n^3)$ complexity using the Hungarian method [Kuh55]. Linear minimization can also be performed using the standard or network simplex algorithms, opening the possibility for optimized and potentially parallelizable implementations. In our experiments we found that the LMO was nonetheless more efficient, both in terms of runtime and memory footprint, when using the Hungarian algorithm compared to off-the-shelf simplex solvers.

Projection: To the best of our knowledge, there is no exact projection method specific to the Birkhoff polytope. An approximate method based on the Douglas-Ratchford splitting algorithm was proposed in [CP21]. Its complexity to achieve ϵ-convergence is $O(n^2c^2/\epsilon^2)$ where c is not known a-priory and depends on the distance of the initial guess for the algorithm to a fixed point of the proximal operator evaluated in each iteration.

The set B_n is used as the feasible region for the (Ord-RDE) problem.

Algorithm 5 Projection onto the non-negative k-sparse polytope

Input: $\mathbf{s} \in \mathbb{R}^n$, $k \in [n]$
Output: projection of $\mathbf{s}$ onto $[0,1]^n \cap \{\mathbf{s} \in \mathbb{R}^n \,:\, \|\mathbf{s}\|_1 \leq k\}$

if $\|\operatorname{clip}(\mathbf{s}, \mathbf{0}_n, \mathbf{1}_n)\|_1 \leq k$ **then**
  return $\operatorname{clip}(\mathbf{s}, \mathbf{0}_n, \mathbf{1}_n)$ ▷ projection onto $[0,1]^n$ already satisfies ℓ_1-constraint
else
  $\boldsymbol{\theta} \leftarrow \operatorname{concatenate}(\mathbf{s}, \mathbf{s} - \mathbf{1}_n)$ ▷ possible break-point locations
  $\boldsymbol{\theta} \leftarrow \operatorname{sort}(\max(\boldsymbol{\theta}, \mathbf{0}_{2n}))$ ▷ only non-negative break-points are valid
  $\ell, u \leftarrow 1, 2n$ ▷ initialize bisection interval $[\theta_\ell, \theta_u]$
  while $u > \ell + 1$ **do**
    $m \leftarrow \lfloor \frac{\ell+u}{2} \rfloor$
    if $\|\operatorname{clip}(\mathbf{s} - \theta_m \cdot \mathbf{1}_n, \mathbf{0}_n, \mathbf{1}_n)\|_1 \leq k$ **then**
      $u \leftarrow m$ ▷ continue bisection in left half $[\theta_\ell, \theta_m]$
    else
      $\ell \leftarrow m$ ▷ continue bisection in right half $[\theta_m, \theta_u]$
    end if
  end while
  $k_\ell = \|\operatorname{clip}(\mathbf{s} - \theta_\ell \cdot \mathbf{1}_n, \mathbf{0}_n, \mathbf{1}_n)\|_1$
  $k_u = \|\operatorname{clip}(\mathbf{s} - \theta_u \cdot \mathbf{1}_n, \mathbf{0}_n, \mathbf{1}_n)\|_1$
  $\theta = \theta_\ell + (\theta_u - \theta_\ell)\frac{(k-k_\ell)}{k_u - k_\ell}$ ▷ find correct value within interval $[\theta_\ell, \theta_u]$
  return $\operatorname{clip}(\mathbf{s} - \theta \cdot \mathbf{1}_n, \mathbf{0}_n, \mathbf{1}_n)$
end if

B.2 Supplementary Experimental Results

Below we present additional results for the experiments from Section 4.4. In particular, we show comparisons of different FW variants for solving (RC-RDE), (MR-RDE),

Algorithm 6 Linear Minimization Oracle for the non-negative k-sparse polytope

Input: $\nabla D(\mathbf{s}) \in \mathbb{R}^n$, $k \in [n]$
Output: a minimizer of $\langle \nabla D(\mathbf{s}), \mathbf{v} \rangle$ over $[0,1]^n \cap \{\mathbf{v} \in \mathbb{R}^n \,:\, \|\mathbf{v}\|_1 \leq k\}$

$S \leftarrow \{\text{indices of } k \text{ smallest components of } \nabla D(\mathbf{s})\} \cap \{j \in [n] \,:\, (\nabla D(\mathbf{s}))_j < 0\}$
$\mathbf{v}_{S^c} \leftarrow \mathbf{0}_{n-|S|}$
$\mathbf{v}_S \leftarrow \mathbf{1}_{|S|}$
return $\mathbf{v}$

and (Ord-RDE) as well as selected results comparing the effects of using the diagonal or the low-rank ADF approximation to evaluating the distortion functional $D(\mathbf{s})$.

B.2.1 Synthetic Binary Strings

We show additional relevance mapping results for the two binary string signals from Figure 4.3 comparing (L-RDE) to (RC-RDE) solved with different variants of Frank-Wolfe algorithms as well as Projected Gradient Descent in Figure B.3. In all cases we present the diagonal and low-rank ADF approximation to evaluating the distortion functional $D(\mathbf{s})$. The rate $k = 6$ for (RC-RDE) is intentionally chosen larger than the optimal rate $k = 5$ that allows for zero distortion in order to see how the different RDE methods distribute the "excess" relevance across the signal. It also better corresponds to sparsity of the (L-RDE) solutions that are indirectly adapted to a specific rate via the regularization parameter λ. The first signal contains a single relevant block of five consecutive ones that is correctly located by all RDE methods. The low-rank variants distribute excess relevance more evenly around the relevant block whereas the diagonal variants put more emphasis on other ones in the signal that could potentially form another relevant block. The PGD solution is most similar to the vanilla FW solution. The second signal contains a single relevant block of five consecutive ones but also an additional disjoint block that is almost relevant (four out of five consecutive components are ones). Most RDE methods still locate the correct relevant block. Again PGD is most similar to FW. The (RC-RDE) solutions with AFW (diagonal and low-rank) and LAFW (only low-rank) could not locate the relevant block and get distracted by additional ones in the signal. In all three cases we observe that this constitutes a rare exception of non-robust behavior of RDE relevance mappings with respect to the rate k. All three methods correctly locate the relevant block for larger rates $k \geq 7$ (not shown). We expect this to be an artifact of the synthetic and constructed nature of this experiment. In fact, the chosen reference distribution $\mathcal{V} = \mathcal{U}(\{0,1\}^n)$ is completely uninformative of any structure in the signals. We do not observe a similar behavior of RDE relevance mappings in the An8flower, MNIST, and STL-10 experiments with more realistic datasets.

B.2.2 An8flower Dataset

We show additional relevance mapping results for the *yellow stem* image from Figure 4.5 comparing (L-RDE) to (RC-RDE) solved with different variants of Frank-Wolfe algorithms as well as Projected Gradient Descent in Figure B.2. In all cases we present the diagonal and low-rank ADF approximation to evaluating the distortion functional $D(\mathbf{s})$. All RDE

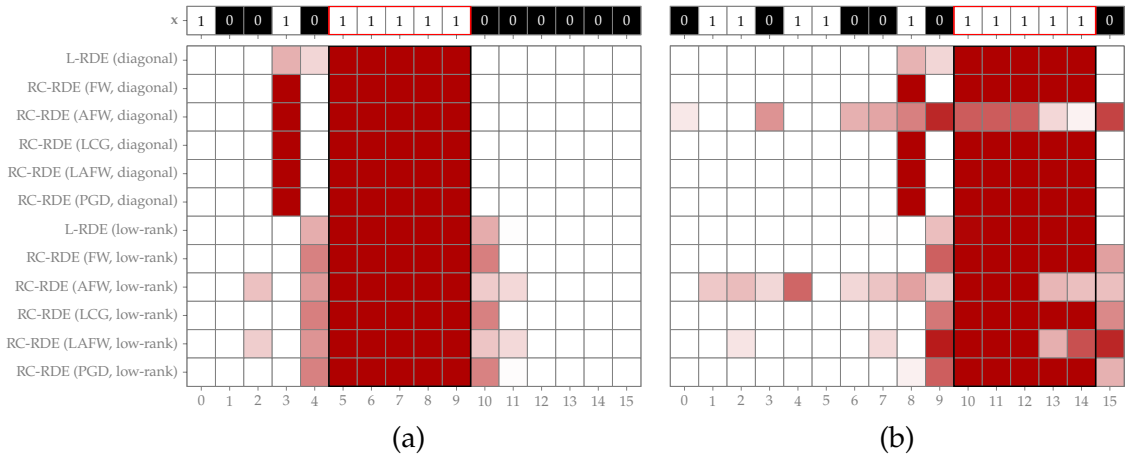


Figure B.1: **Synthetic Binary Strings – Relevance Maps.** Relevance mappings generated by several methods for two binary strings. The left string (a) contains one block of five consecutive ones whereas the right string (b) contains one complete and one incomplete block. This complements Figure 4.3.

methods mark the discriminative image region containing the stem as relevant. Low-rank solutions tend to additionally mark regions surrounding the stem as relevant, which is less pronounced for the diagonal variants. Table B.1 shows a quantitative evaluation of the performance of the corresponding relevance mapping methods with respect to two different similarity measures averaged over 12 examples from the dataset (one for each class). The diagonal FW, LCG, and PGD variants of (RC-RDE) achieve the best results with regard to the Pearson correlation coefficient. The diagonal FW and LCG variants also perform best with respect to the Jaccard index. All RDE methods clearly outperform the other comparison baseline relevance mapping methods in Table 4.1.

B.2.3 MNIST Dataset

Figure B.3 shows a comparison of results obtained using the different FW variants for solving (RC-RDE) and (MR-RDE) for the MNIST experiment example image from Figure 4.7. For clarity and due to the advantageous performance of the diagonal over the low-rank ADF approximation to evaluating the distortion functional $D(\mathbf{s})$ in the An8flower experiment, we only show diagonal RDE results for MNIST.[2] The last row shows the single-rate mappings $\mathbf{\Pi}^{\text{opt}}\mathbf{p}_k$ associated to the SFW solution of (Ord-RDE) as well as a corresponding multi-rate mapping $\frac{1}{n-1}\sum_{k\in[n-1]}\mathbf{\Pi}^{\text{opt}}\mathbf{p}_k$. All FW variants yield similar results and are robust across varying rates, in the sense that solutions for larger rates add additional features to the relevant set without significantly modifying the features that were already considered relevant at smaller rates.

Figure B.4 complements Figure 4.8 and shows the relevance ordering test results for all FW variants for the MNIST experiment.

The quantitative effect of solving (RC-RDE) for different rates is illustrated in Figure B.5. For the sake of clarity, we only show the relevance ordering test results for FW (left)

[2] We have checked for (L-RDE) and (RC-RDE) with vanilla FW that the low-rank variants behave qualitatively and quantitatively comparably to the diagonal variants.

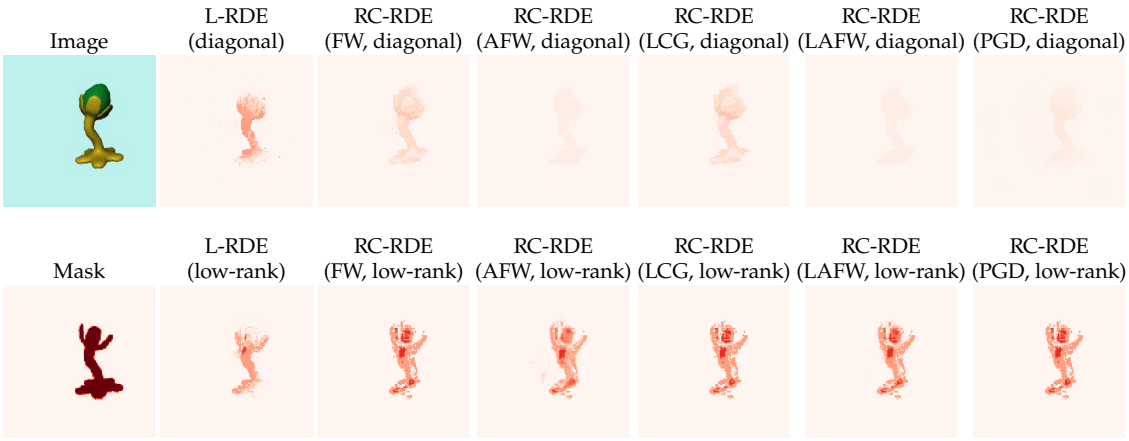


Figure B.2: **An8flower – Relevance Maps.** Relevance mappings generated by several methods for an image from the An8flower benchmark dataset classified as *yellow stem* by our network. The rate constraint for RC-RDE is $k = 1000$ ($n = 49152$). This complements Figure 4.5

Table B.1: **An8flower – Correlation Comparison.** 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). The rate constraint for RC-RDE is $k = 1000$ ($n = 49152$). This complements Table 4.1.

	L-RDE (diagonal)	RC-RDE (FW, diagonal)	RC-RDE (AFW, diagonal)	RC-RDE (LCG, diagonal)	RC-RDE (LAFW, diagonal)	RC-RDE (PGD, diagonal)
Pearson Corr.	0.23 ± 0.16	**0.35 ± 0.20**	0.32 ± 0.19	**0.35 ± 0.20**	0.33 ± 0.17	**0.35 ± 0.20**
Jaccard Index	0.14 ± 0.09	**0.25 ± 0.16**	0.24 ± 0.16	**0.25 ± 0.16**	0.22 ± 0.13	0.20 ± 0.15

	L-RDE (low-rank)	RC-RDE (FW, low-rank)	RC-RDE (AFW, low-rank)	RC-RDE (LCG, low-rank)	RC-RDE (LAFW, low-rank)	RC-RDE (PGD, low-rank)
Pearson Corr.	0.27 ± 0.16	0.30 ± 0.13	0.30 ± 0.13	0.30 ± 0.13	0.31 ± 0.13	0.30 ± 0.12
Jaccard Index	0.16 ± 0.08	0.17 ± 0.09	0.17 ± 0.09	0.16 ± 0.08	0.17 ± 0.09	0.16 ± 0.07

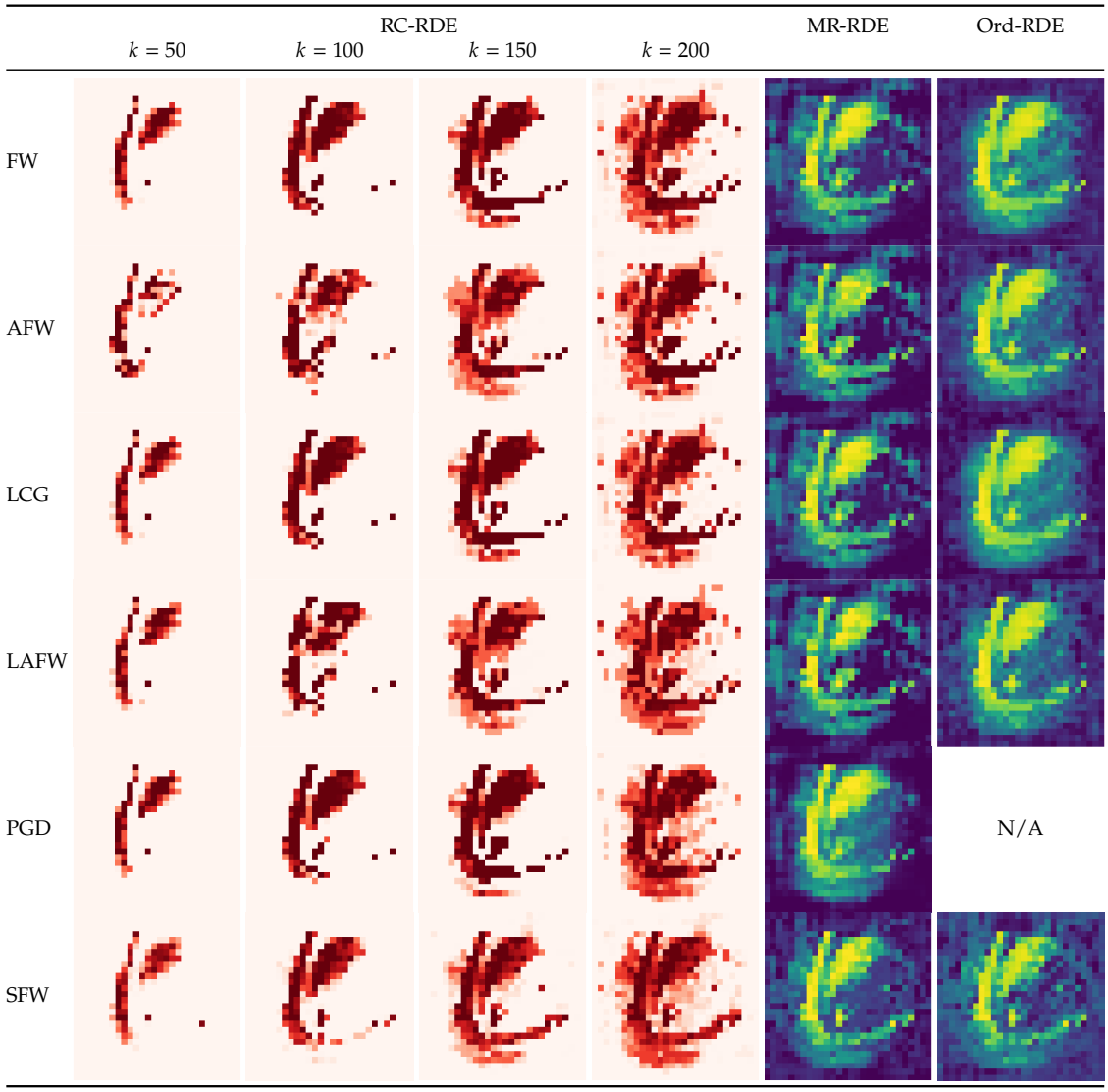


Figure B.3: **MNIST – Relevance Maps.** Relevance mappings generated by different FW variants for an image from the MNIST dataset classified as *digit six* by our network. Multi-rate (MR-RDE) and ordering (Ord-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow). This complements Figure 4.7.

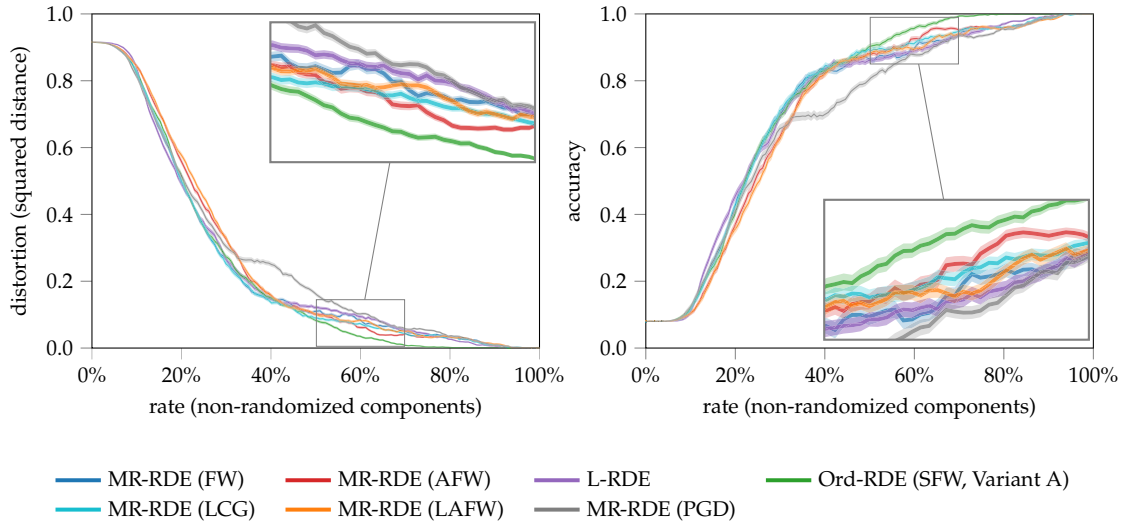


Figure B.4: **MNIST – Ordering Comparison.** Relevance ordering test results for different FW variants for the MNIST dataset using squared distance (left) and classification accuracy (right) as performance measure. An average result over 50 images from the test set (5 images per class randomly selected) and 512 random input samples per image is shown (shaded regions mark ± standard deviation). This complements Figure 4.8.

and PGD (right). The results for AFW, LCG, and LAFW are comparable. The (RC-RDE) mappings achieve a low distortion at the rates for which they were optimized but are suboptimal at other rates. The combined (MR-RDE) solution approximates the lower envelope of the individual curves and performs best overall.

A comparison of runtimes for the MNIST experiment and the corresponding numbers of iterations that are taken until the termination criterion $\langle \mathbf{s}_t - \mathbf{v}_t, \nabla D(\mathbf{s}_t) \rangle < \varepsilon = 10^{-7}$ is reached for the different FW variants is shown in Figure B.6(a). We observe that AFW converges fastest for small rates k, while all variants perform similarly at higher rates (with slight advantages of FW and LCG over AFW and LAFW). This is due to a reduced number of iterations of AFW and LAFW at small rates. On the other hand, the increased runtime of the active-set methods AFW and LAFW can be explained by the fact that each vertex in the active set is a sparse vector with more and more non-zero components as the rate increases. Hence, active set operations require more arithmetic operations at higher rates. All methods reach the maximum number of $T = 2000$ iterations before satisfying the termination criterion at high rates. This can be explained by the fact that the termination threshold $\varepsilon = 10^{-7}$ is chosen quite conservatively to ensure convergence of all methods on all instances. However, we observe that all methods typically converge much faster to a satisfactory solution. Figure B.6(b) shows (RC-RDE) solutions at a single exemplary rate $k = 150$ for the MNIST image from Figure 4.7 after $T = 50$, 100, 150, and 200 iterations. The solutions are visually indistinguishable for all methods, confirming that they were already mostly converged after only 50 iterations. We do not show a comparable analysis of runtimes and iteration counts for the STL-10 experiment. Due to higher computational costs, the STL-10 mappings were computed in parallel on different machines. Hence, no runtimes that are directly comparable between the different methods are available.

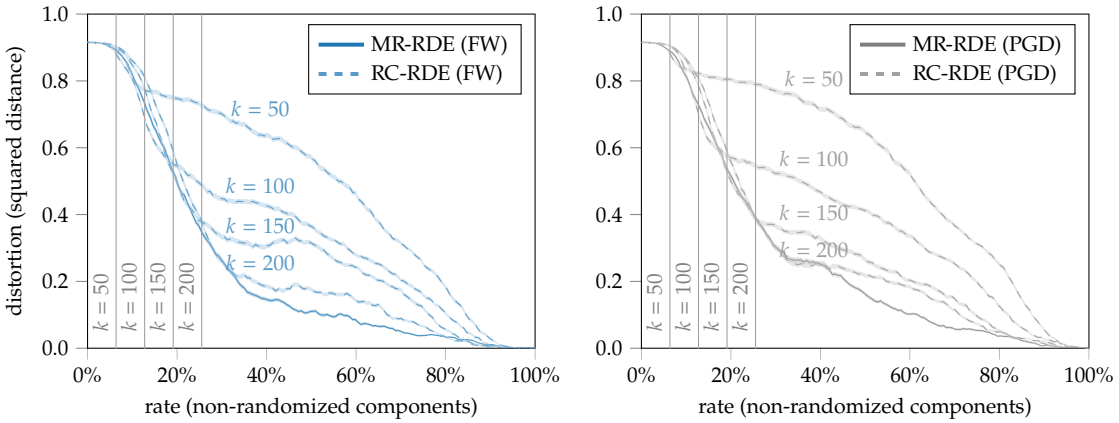


Figure B.5: **MNIST – Ordering Comparison.** Relevance ordering test results for FW and PGD solutions of (RC-RDE) for MNIST at various rates. Vertical lines show the rates k at which the mappings were optimized. The combined (MR-RDE) solutions approximate the respective lower envelopes of the individual curves. An average result over 50 images from the test set (5 images per class) and 512 noise input samples per image is shown (shaded regions mark ± standard deviation).

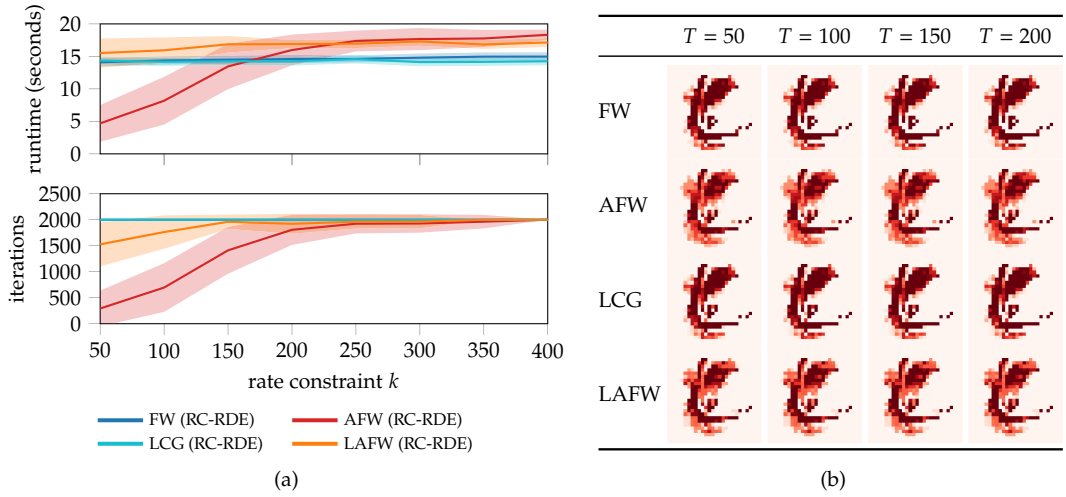


Figure B.6: **MNIST – RDE Convergence** (a) Average runtimes (top) and number of iterations until convergence (bottom) of the considered FW variants for (RC-RDE) on MNIST at different rates. An average result over 50 images from the test set is shown (shaded regions mark ± standard deviation). (b) Relevance mappings obtained at rate $k = 150$ after different maximal numbers of iterations. Results for the same MNIST image from Figure 4.7 classified as *digit six* by the network are shown. All methods were converged effectively already after only $T = 50$ iterations.

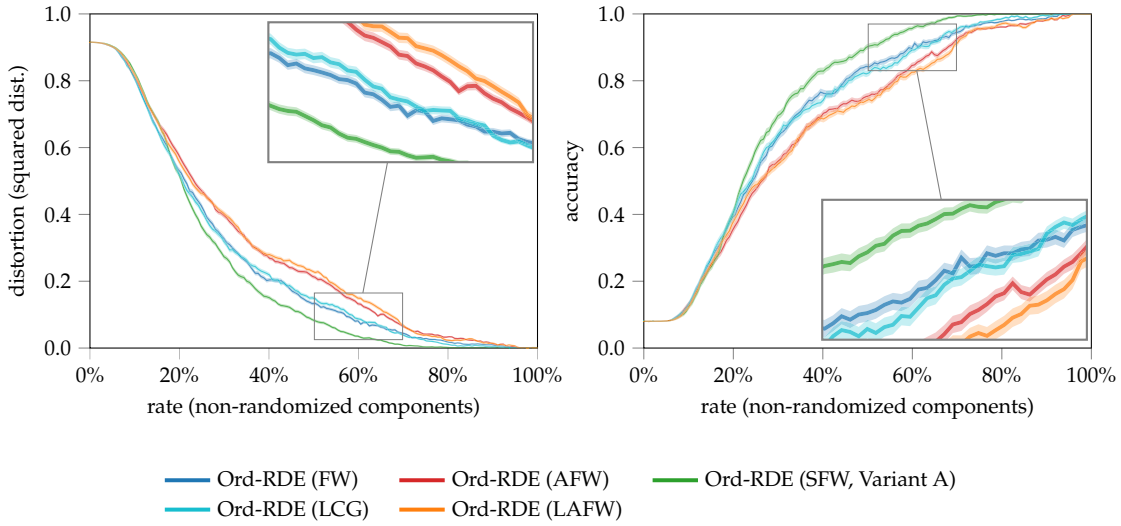


Figure B.7: **MNIST – Ordering Comparison.** Relevance ordering test results for (Ord-RDE) on the MNIST dataset for all considered FW variants. An average result over 50 images from the test set (5 images per class) and 512 noise input samples per image is shown (shaded regions mark ± standard deviation). This complements Figure B.4. Additional SFW variants are shown in Figure B.8.

The relevance-ordering problem (Ord-RDE) can be solved with a stochastic Frank-Wolfe algorithm or with deterministic Frank-Wolfe algorithms if the number n of terms in the objective function is not too large. For the MNIST dataset, both approaches are feasible. We compare the relevance-ordering comparison test results of all FW variants in Figure B.7. We observe that all variants perform well and similarly for very low and high rates. For rates between 20% and 80% of the total number of components SFW has an advantage over FW and LCG which in turn perform slightly better than AFW and LAFW.

Further, we compare different hyperparameter configurations for SFW regarding the batch sizes b_t and momentum factors ρ_t. Hazan and Luo suggest a linearly increasing[3] batch size [HL16] and Mokhtari, Hassani, and Karbasi propose a momentum factor $\rho_t = 1 - 4/(8+t)^{\frac{2}{3}}$ approaching one [MHK20]. We consider the following combinations:

	Variant A	Variant B	Variant C	Variant D	Variant E	Variant F
momentum	$\rho_t = 0$	$\rho_t = 0$	$\rho_t = \frac{1}{2}$	$\rho_t = \frac{1}{2}$	$\rho_t = 1 - \frac{4}{(8+t)^{\frac{2}{3}}}$	$\rho_t = 1 - \frac{4}{(8+t)^{\frac{2}{3}}}$
batch size	$b_t = 40$	$b_t = \min\{40+t, 100\}$	$b_t = 40$	$b_t = \min\{40+t, 100\}$	$b_t = 40$	$b_t = \min\{40+t, 100\}$

The relevance-ordering comparison test results are shown in Figure B.8. For reference, Variant A corresponds to the SFW result also shown in Figures B.4 and B.7. We observe that all variants perform well and similarly across all rates. The batch size has negligible effect in this experiment, while momentum yields no advantage. In particular, both configurations without any momentum perform best. Hence, we show only Variant A (no momentum, constant batch size) in all other experiments.

[3] We limit the batch size growth up to a maximal size $b_{\max} = 100$ in our experiments.

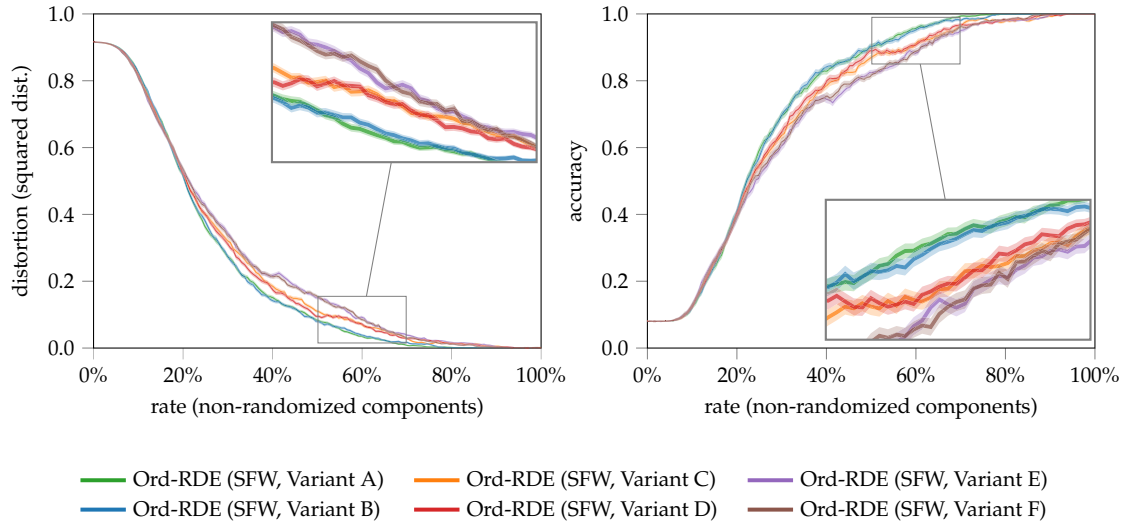


Figure B.8: **MNIST – Ordering Comparison.** Relevance ordering test results for (Ord-RDE) on MNIST for different variants of SFW. An average result over 50 images from the test set (5 images per class) and 512 noise input samples per image is shown (shaded regions mark ± standard deviation). This complements Figure B.7.

B.2.4 STL-10 Dataset

Figure B.9 complements Figure 4.12 and shows the relevance ordering test results for all FW variants for the STL-10 experiment. As in the MNIST experiment, for clarity and due to the advantageous performance of the diagonal over the low-rank ADF approximation to evaluating the distortion functional $D(\mathbf{s})$ in the An8flower experiment, we only show diagonal RDE results for STL-10.[4]

Further, we complement Figures 4.10 and 4.11 by Figures B.10 to B.19 showing additional examples of relevance maps for images of different classes from the STL-10 dataset as well as additional results for all FW variants for the images from Figures 4.10 and 4.11.

B.3 Specifications of the Synthetic Binary Strings Experiment

B.3.1 Network Architecture

Recall that the underlying binary classifier is given by the Boolean function

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

that checks binary strings of length n for the existence of a block of k consecutive ones. A ReLU network with two hidden layers that interpolates Ψ can be constructed as

$$\Phi(\mathbf{x}) = \mathbf{W}_3 \varrho\left(\mathbf{W}_2 \varrho(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2\right) + \mathbf{b}_3$$

[4] We have checked for (L-RDE) and (RC-RDE) with vanilla FW that the low-rank variants behave qualitatively and quantitatively comparably to the diagonal variants.

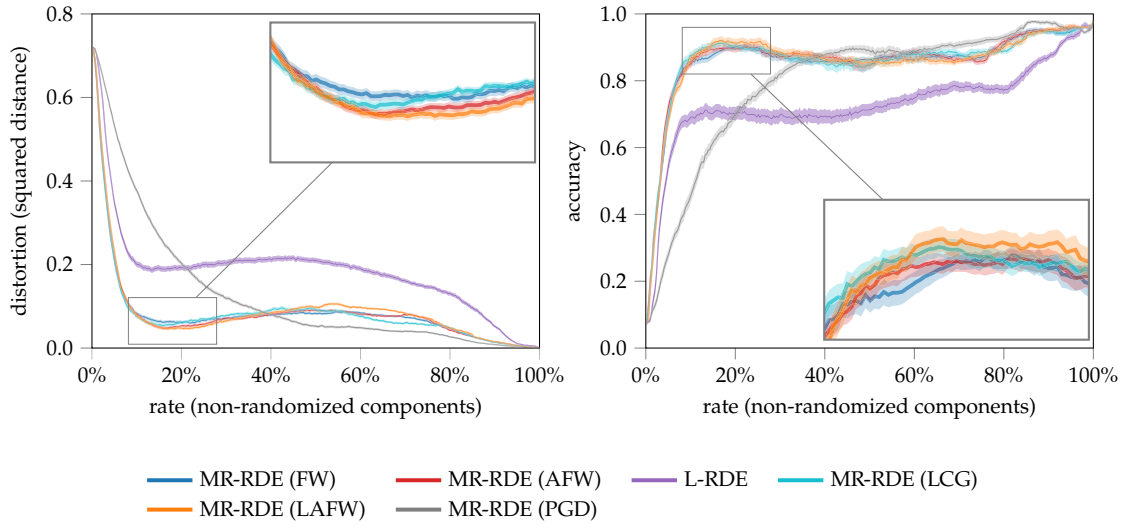


Figure B.9: **STL-10 – Ordering Comparison.** Relevance ordering test results for different FW variants for for the STL-10 dataset using squared distance (left) and classification accuracy (right) as performance measure. An average result over 50 images from the test set (5 images per class randomly selected) and 64 random input samples per image is shown (shaded regions mark ± standard deviation). This complements Figure 4.12.

with

$$\begin{aligned}
\mathbf{W}_1 &= \left[\sum_{j=i}^{i+k-1} \mathbf{e}_j^T\right]_{i=1}^{n-k+1} \in \mathbb{R}^{(n-k+1)\times n} && \text{and} \quad \mathbf{b}_1 = -(k-1)\cdot \mathbf{1}_{n-k+1} \in \mathbb{R}^{n-k+1},\\
\mathbf{W}_2 &= -\mathbf{1}_{n-k+1}^\top \in \mathbb{R}^{1\times(n-k+1)} && \text{and} \quad \mathbf{b}_2 = 1 \in \mathbb{R}^1,\\
\mathbf{W}_3 &= -1 \in \mathbb{R}^{1\times 1} && \text{and} \quad \mathbf{b}_3 = 1 \in \mathbb{R}^1,
\end{aligned}$$

where $\mathbf{e}_j$ is the j-th unit vector in $\mathbb{R}^n$. This network is purely constructed and not trained on any data. We use $n = 16$ and $k = 5$ in our experiment.

B.3.2 RDE Optimization

For the (RC-RDE) optimization we consider FW, AFW, LCG, LAFW, and PGD with the initial guess chosen as $\mathbf{s} = \mathbf{0}_n$. For the (L-RDE) optimization we solve the resulting box-constrained problem via L-BFGS-B [Byr+95]. The initial guess for $\mathbf{s}$ is chosen as the mean of $\mathcal{U}([0,1]^n)$ in this case. As reference distribution $\mathcal{V}$ we use the Gaussian distribution with mean and variance equal to the mean and variance of $\mathcal{U}([0,1]^n)$ for both (RC-RDE) and (L-RDE). For all low-rank variants we use the full rank $r = 16$. To estimate a good value for the (L-RDE) regularization parameter λ we solve the optimization problem for values $\lambda = 10^q$ with ten values of q spaced evenly in the interval $[-5, 0]$. We compare the results visually and empirically select $\lambda = 1.67 \cdot 10^{-3}$, which results in relevance maps with a sparsity that corresponds well to the true block size $k = 5$.

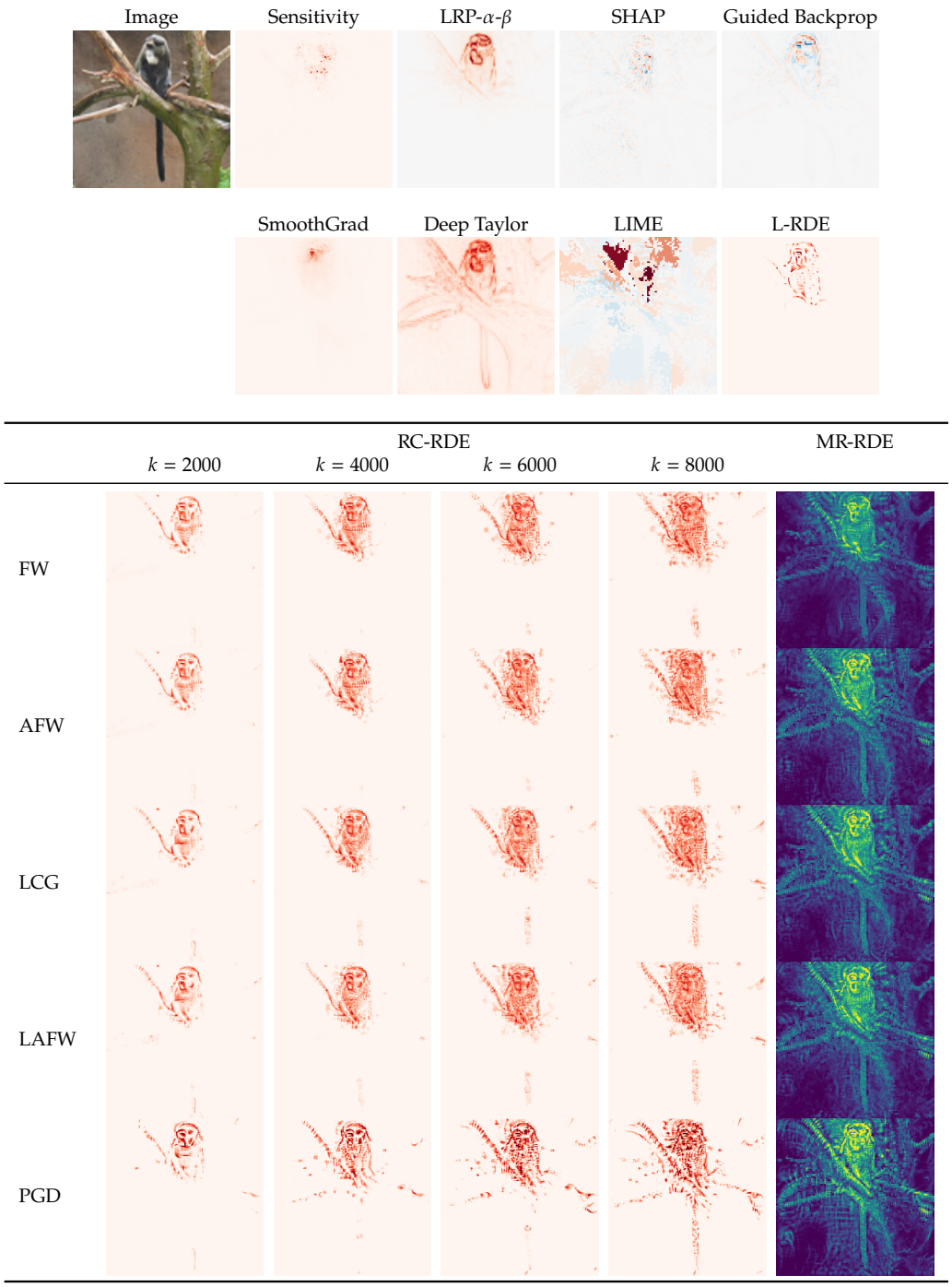


Figure B.10: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *monkey* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow). This complements Figure 4.10.

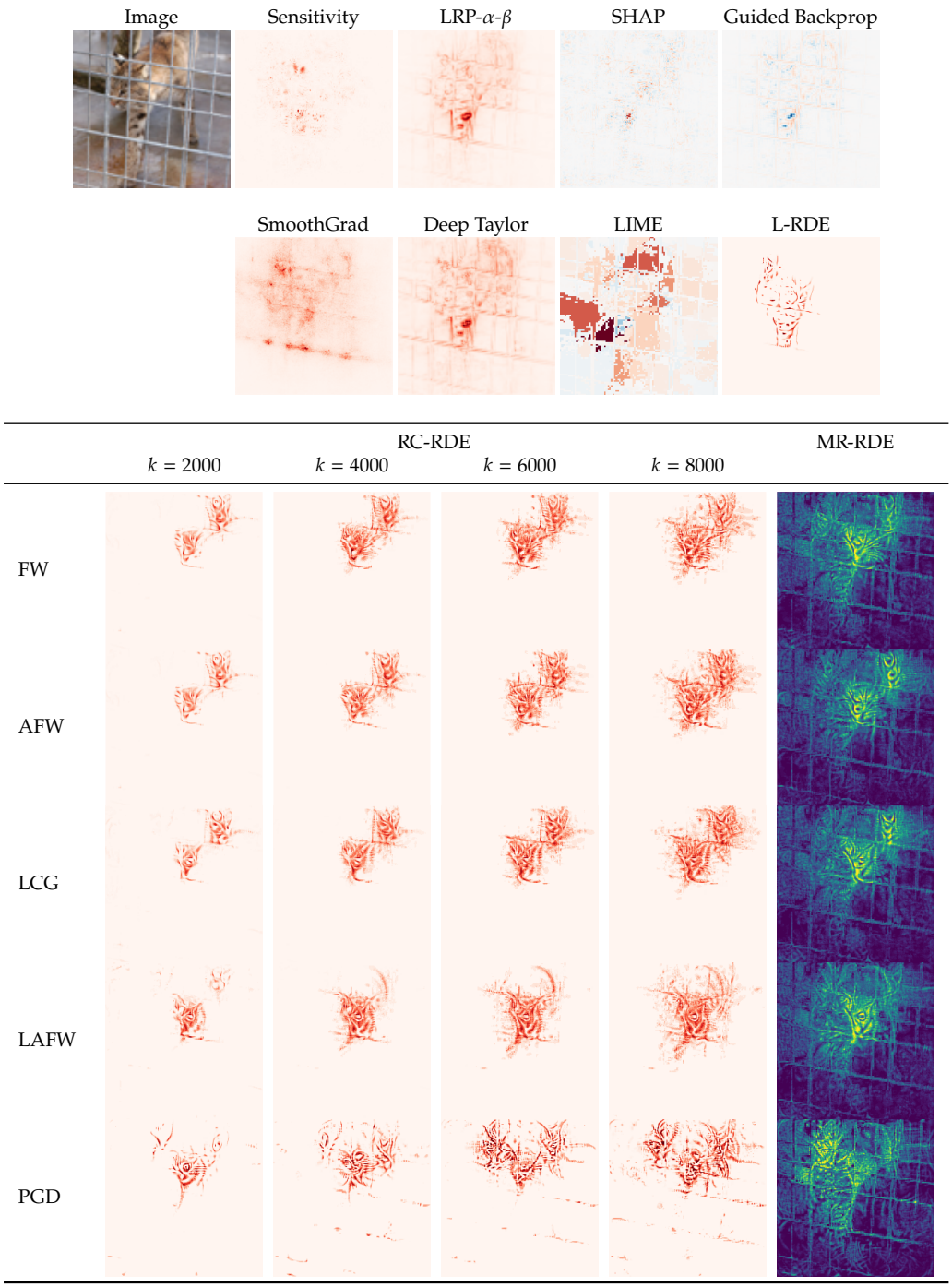


Figure B.11: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *cat* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow).

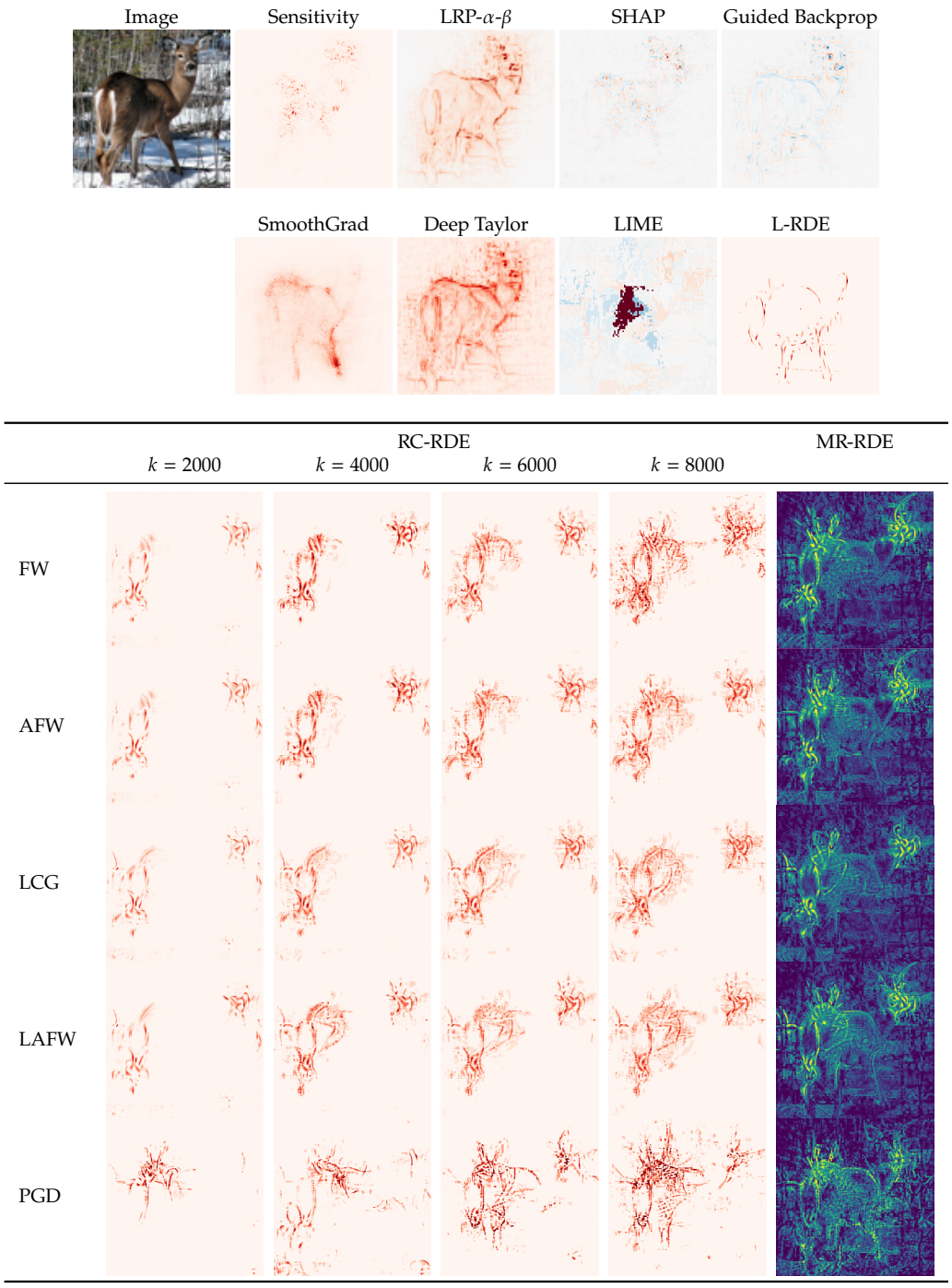


Figure B.12: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *deer* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow).

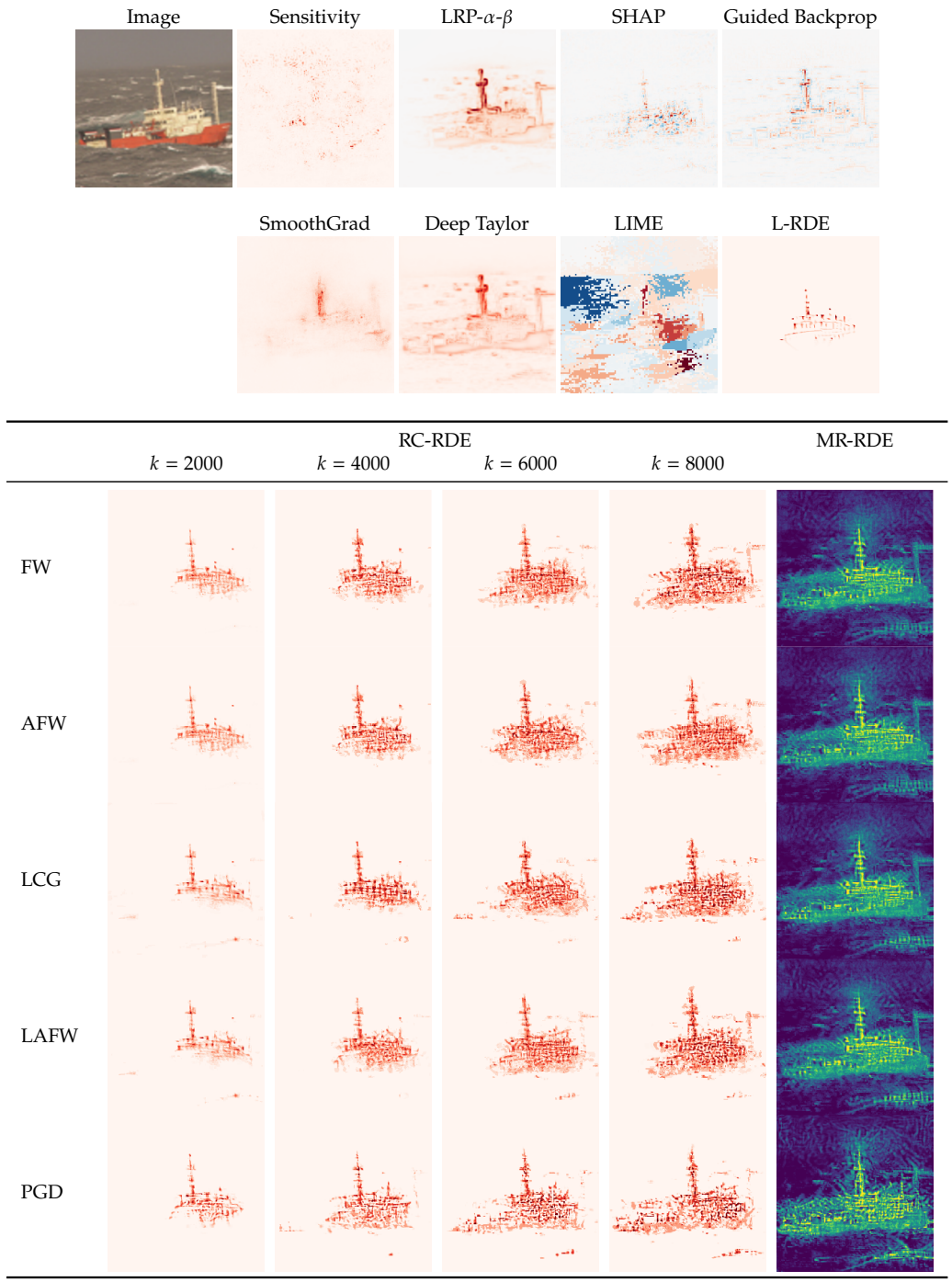


Figure B.13: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *ship* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow).

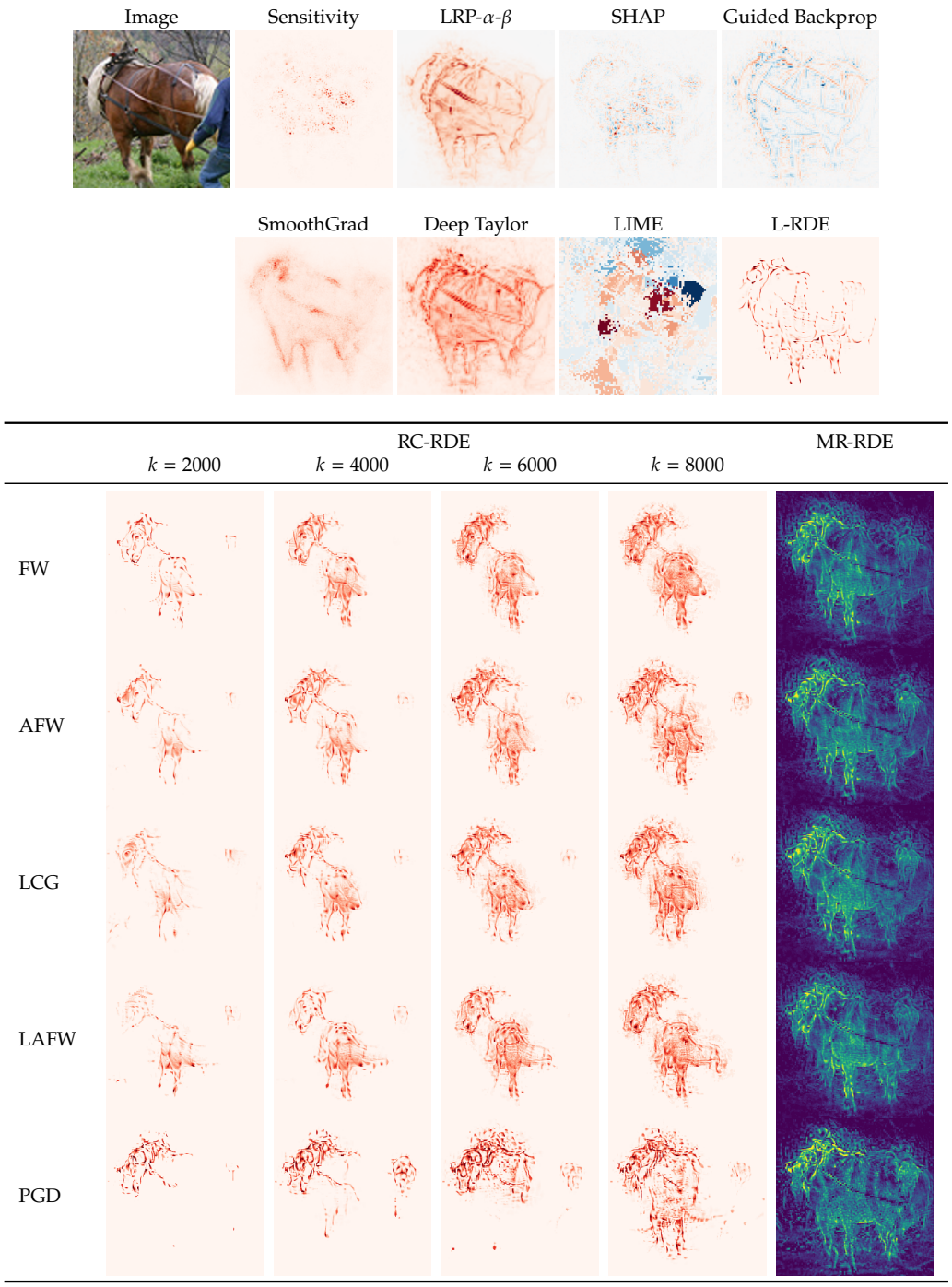
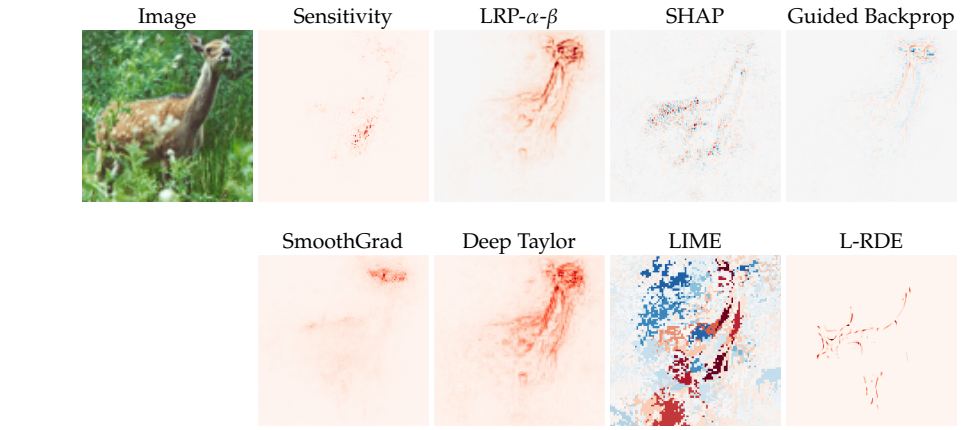


Figure B.14: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *horse* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow). This complements Figure 4.11.



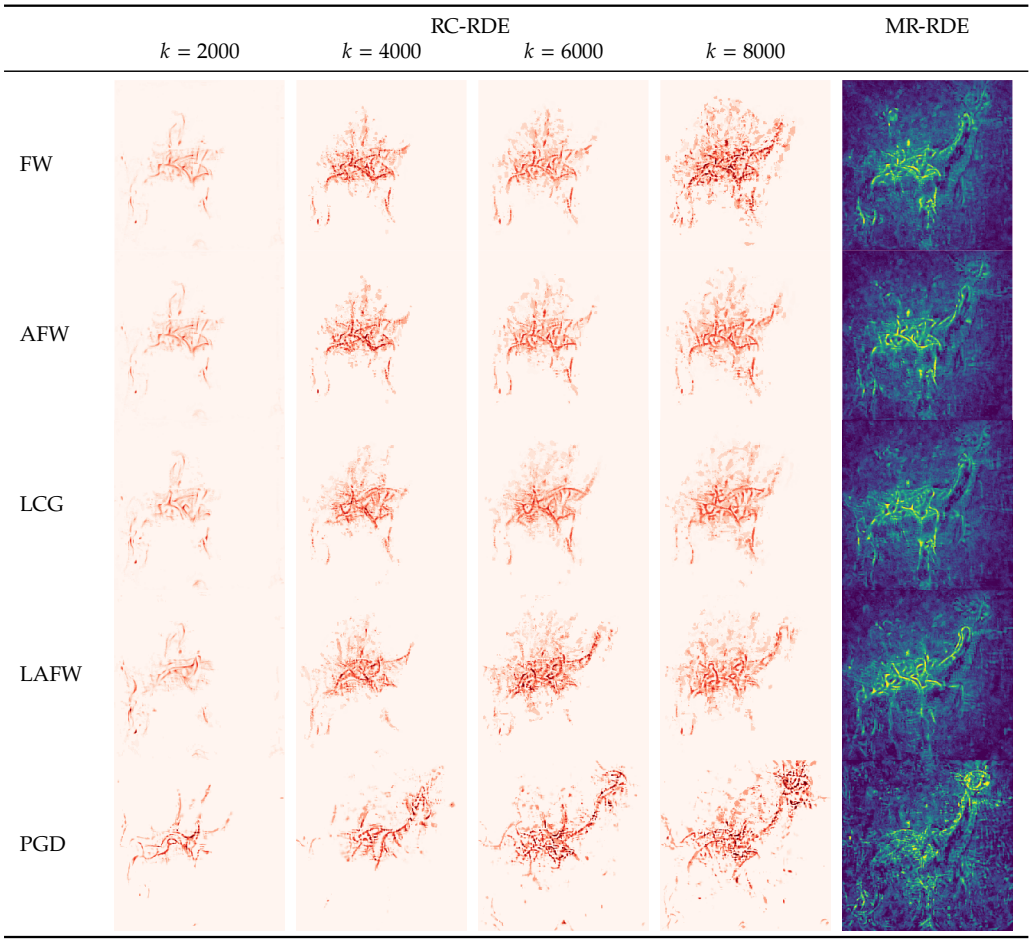


Figure B.15: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *deer* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow).

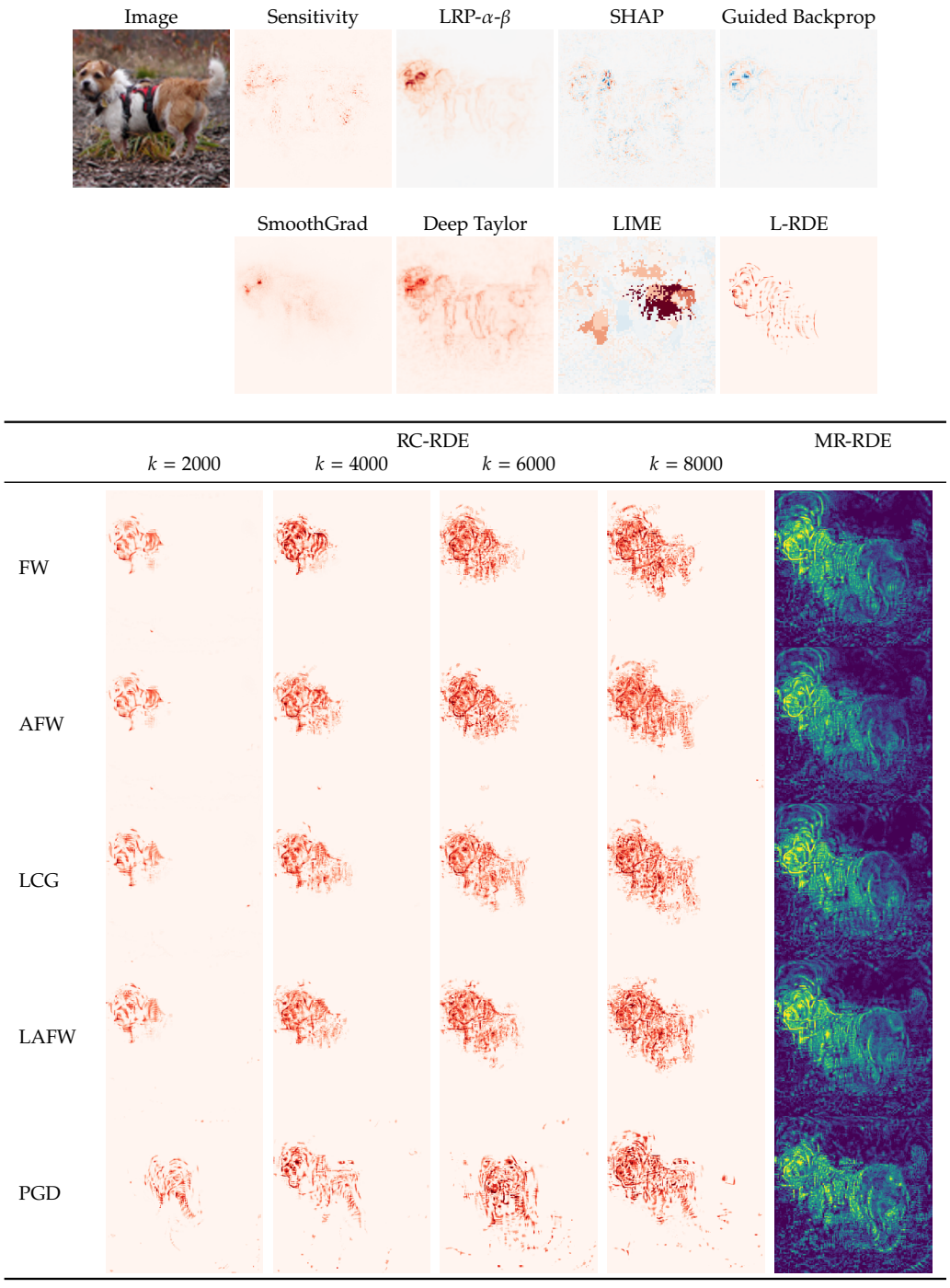


Figure B.16: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *dog* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow).

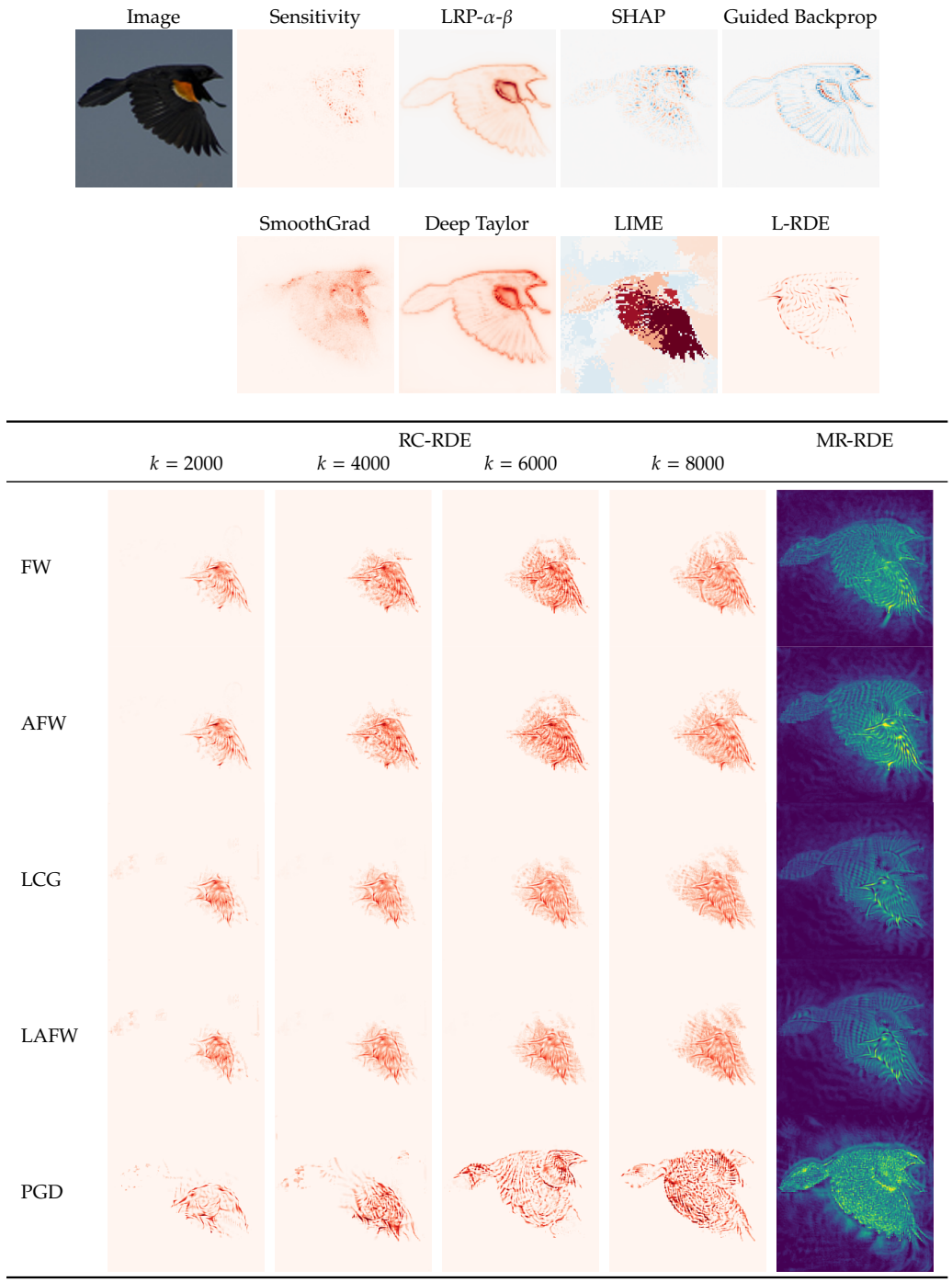


Figure B.17: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *bird* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow).

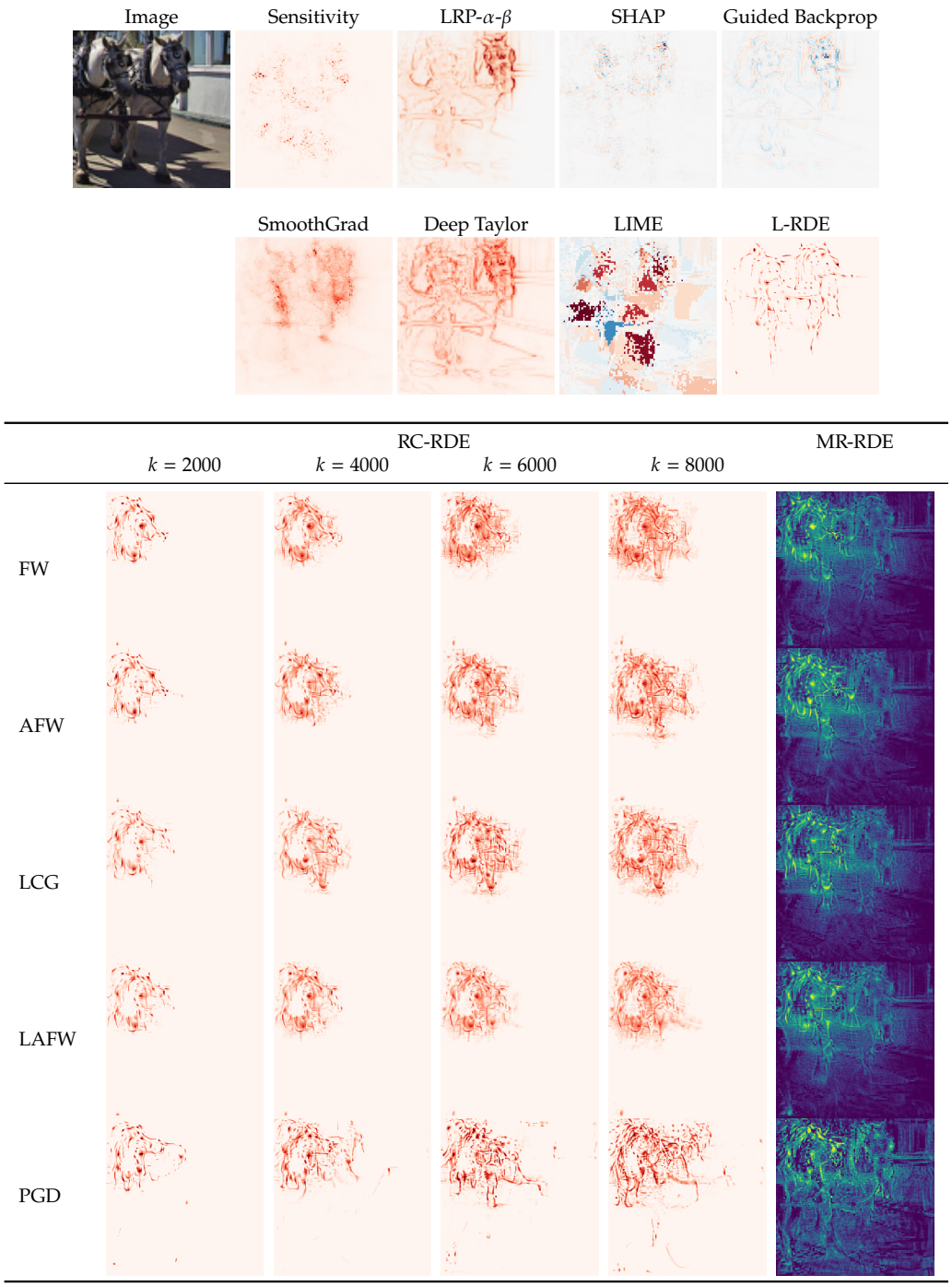


Figure B.18: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *horse* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow).

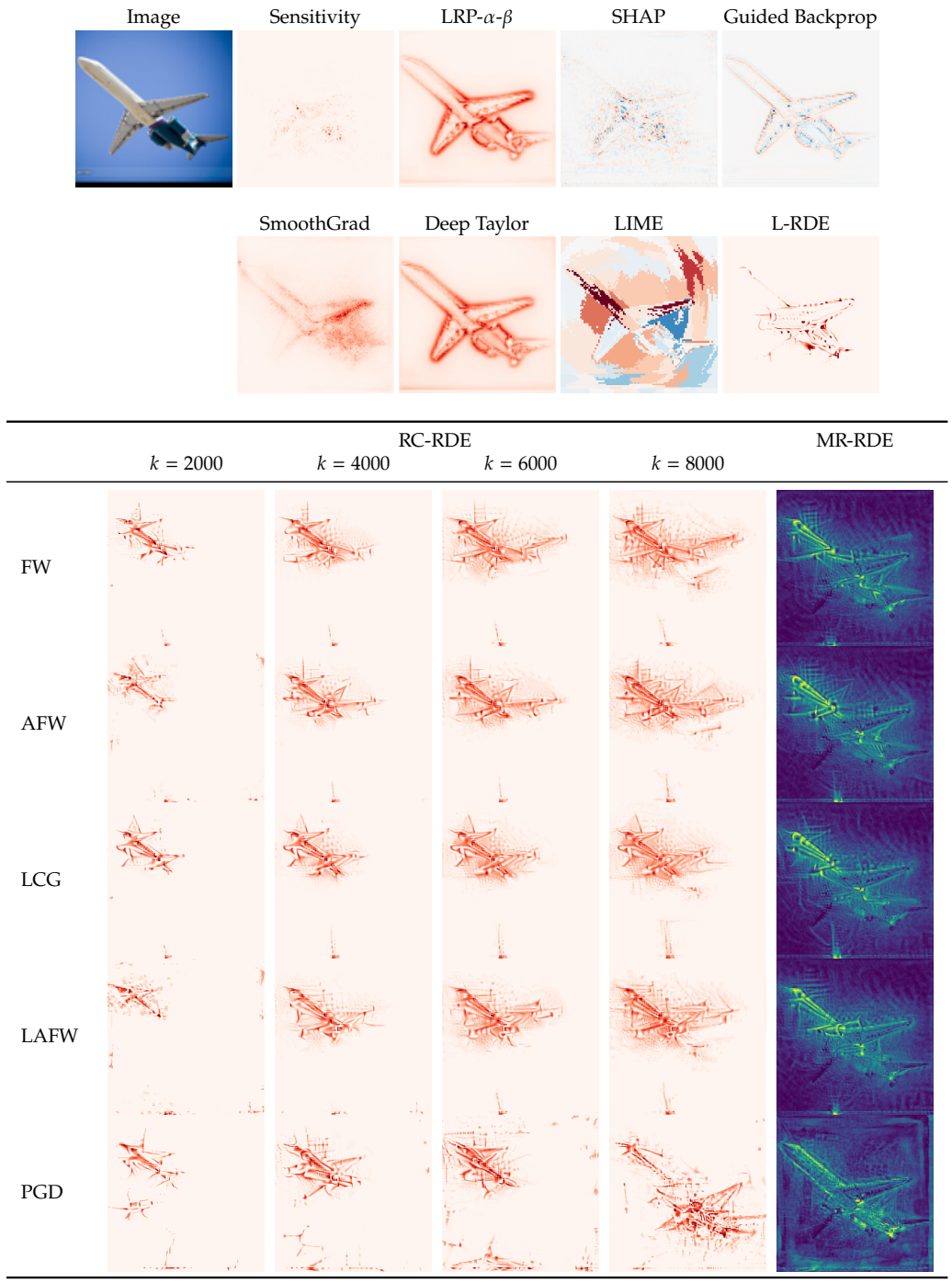


Figure B.19: **STL-10 – Relevance Maps.** Relevance mappings generated by several methods for an image from the STL-10 dataset classified as *airplane* by our network. The colormap indicates positive relevances as red and negative relevances as blue. Multi-rate (MR-RDE) solutions are shown in a different colormap to highlight the fact, that they are not to be viewed as sparse relevance maps but represent component orderings from least relevant (blue) to most relevant (yellow).

B.3.3 Comparison Methods

We use the Innvestigate toolbox for generating relevance mappings according to SmoothGrad [Smi+17] with a noise scale of 0.5 and 64 noise samples. We use the SHAP toolbox to generate relevance mappings according to SHAP [LL17] and use the `DeepExplainer` method for deep network models with 1024 reference inputs drawn randomly from $\mathcal{U}([0,1]^n)$. Finally, we use the LIME toolbox to generate relevance mappings according to LIME [RSG16]. We use the local explanations of the `LimeTabularExplainer` method with 1024 reference inputs drawn randomly from $\mathcal{U}([0,1]^n)$.

B.4 Specifications of the An8flower Experiment

B.4.1 Network Architecture and Training

For the An8flower experiments we use a convolutional neural network with three convolution layers each followed by average-pooling and finally two fully-connected layers and softmax output, see Table B.2 for a detailed description of the architecture. We train the randomly initialized network end-to-end for 100 epochs to a final test accuracy of 0.99. We use 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 is used.

B.4.2 RDE Optimization

We use 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 normalize the distortion function by $\Phi(\mathbf{x})$. For the (RC-RDE) optimization we consider FW, AFW, LCG, LAFW, and PGD with the initial guess chosen as $\mathbf{s} = \mathbf{0}_n$. For the (L-RDE) optimization we solve the resulting box-constrained problem via L-BFGS-B [Byr+95]. The initial guess is chosen as $\mathbf{s} = 0.5 \cdot \mathbf{1}_n$ in this case. As reference distribution $\mathcal{V}$ we use the Gaussian distribution with mean and variance estimated from the training data for both (RC-RDE) and (L-RDE). We estimate good values for the (L-RDE) regularization parameter λ and the rank r of the low-rank variants as follows: We use a single randomly chosen image signal from the test set and solve the (L-RDE) optimization problem for values $\lambda = 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}$ to get a first rough estimate. We compare the results visually and determine 10^0 and 10^1 as good candidates. We refine the search using values $\lambda = 1, 2, \ldots, 10$ and finally empirically select $\lambda = 5$. Once the regularization parameter is determined we proceed similarly for the rank and test the values $r = 5, 10, 30, 50, 100$. We observe that choosing ranks larger than 10 has negligible effect on the relevance mappings and select $r = 10$ as a promising option for all low-rank variants.

B.4.3 Comparison Methods

We use the Innvestigate toolbox for generating relevance mappings according to Layer-wise Relevance Propagation (LRP) [Bac+15], Deep Taylor decompositions [MSM18], Sensitivity Analysis [SVZ13], SmoothGrad [Smi+17], and Guided Backprop [Spr+15]. We use LRP-α-β with the parameter preset `SequentialPresetAFlat` recommended for convolutional networks and numerical stability parameter $\epsilon = 0.2$. We use the bounded Deep Taylor

Table B.2: **An8flower – Network Architecture.** 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	-	-	-

method with lower and upper bounds given by $\min_j x_j$ and $\max_j x_j$ respectively. For SmoothGrad we use a noise scale of $0.3 \cdot (\max_j x_j - \min_j x_j)$ and 64 noise samples. We use the SHAP toolbox to generate relevance mappings according to SHAP [LL17] and use 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 use the LIME toolbox to generate relevance mappings according to LIME [RSG16]. We use the local explanations of the `LimeImageExplainer` method recommended for image data.

B.5 Specifications of the Relevance Ordering Test Experiment for MNIST

B.5.1 Network Architecture and Training

For the MNIST experiment we use a convolutional neural network with three convolution layers each followed by average-pooling and finally two fully-connected layers and softmax output, see Table B.3 for a detailed description of the architecture. We trained the randomly initialized network end-to-end for 100 epochs to a final test accuracy of 0.99. We use 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 is used. We augment 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-process all data by sample-wise mean centering.

B.5.2 RDE Optimization

We use 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 normalize the distortion function by $\Phi(\mathbf{x})$. For the (RC-RDE) optimization we consider FW, AFW, LCG, LAFW, and PGD with the initial guess chosen as $\mathbf{s} = \mathbf{0}_n$. For the (Ord-RDE) optimization we consider FW, AFW, LCG, LAFW, and SFW with the initial guess chosen as $\mathbf{\Pi} = \mathbf{I}_{n \times n}$. For

the (L-RDE) optimization we solve the resulting box-constrained problem via PGD with a momentum term with factor 0.85 (chosen based on previous experiences and not further tuned). The initial guess is chosen as $\mathbf{s} = 0.25 \cdot \mathbf{1}_n$ in this case. As reference distribution $\mathcal{V}$ we use the Gaussian distribution with mean and variance estimated from the training data for both (RC-RDE) and (L-RDE). We estimate good values for the (L-RDE) regularization parameter λ as follows: We use a single randomly chosen image signal from the test set and solve the (L-RDE) optimization problem for values $\lambda = 10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}$ to get a first rough estimate. We compare the results visually as well as using the relevance ordering test described below and determine 10^{-1} and 10^{0} as good candidates. We refine the search using values $\lambda = 0.1, 0.2, \ldots, 1.0$ and finally empirically select $\lambda = 0.5$.

B.5.3 Comparison Methods

We use the Innvestigate toolbox for generating relevance mappings according to Layer-wise Relevance Propagation (LRP) [Bac+15], Deep Taylor decompositions [MSM18], Sensitivity Analysis [SVZ13], SmoothGrad [Smi+17], and Guided Backprop [Spr+15]. We use LRP-α-β with the parameter preset `SequentialPresetAFlat` recommended for convolutional networks and numerical stability parameter $\epsilon = 0.2$. We use the bounded Deep Taylor method with lower and upper bounds given by $\min_j x_j$ and $\max_j x_j$ respectively. For SmoothGrad we use a noise scale of $0.3 \cdot (\max_j x_j - \min_j x_j)$ and 64 noise samples. We use the SHAP toolbox to generate relevance mappings according to SHAP [LL17] and use 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 use the LIME toolbox to generate relevance mappings according to LIME [RSG16]. We use the local explanations of the `LimeImageExplainer` method recommended for image data.

B.5.4 Relevance Ordering Comparison Test

For the relevance ordering based comparison test we sort components according to their relevance scores (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. Here, we increase the set of fixed components always in groups of size 5 (resulting in 157 steps until all 784 components are fixed) and use 512 random samples drawn from $\mathcal{U}([0,1]^n)$ per step. This procedure is repeated and the results are averaged over 50 different input signals spaced evenly over the test data set (5 images for each of the 10 classes).

B.6 Specifications of the Relevance Ordering Test Experiment for STL-10

B.6.1 Network Architecture and Training

For the STL-10 experiment we use a VGG-16 network [SZ14] pre-trained on the Imagenet dataset as a baseline, see Table B.4 for a detailed description of the architecture. We refine the network for the STL-10 dataset in three stages. First, we fix the convolutional blocks of the Imagenet network and train only the fully-connected layers for 500 epochs. Then we train the complete network end-to-end for another 500 epochs. Finally, after replacing all

Table B.3: **MNIST – Network Architecture.** 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	-	-	-

max-pooling layers by average-pooling layers, we again train end-to-end for another 500 epochs to a final test accuracy of 0.935. We us 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 use a combined ℓ_2- and ℓ_1-regularization term with regularization parameters $5 \cdot 10^{-4}$ and $5 \cdot 10^{-5}$ respectively. The standard training/validation/testing split of the data set is used. We augment 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.

B.6.2 RDE Optimization

We use 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 normalize the distortion function by $\Phi(\mathbf{x})$.

For the (RC-RDE) optimization we consider FW, AFW, LCG, LAFW, and PGD with the initial guess chosen as $\mathbf{s} = \mathbf{0}_n$. For the (L-RDE) optimization we solve the resulting box-constrained problem via PGD with a momentum term with factor 0.85 (chosen based on previous experiences and not further tuned). The initial guess is chosen as $\mathbf{s} = 0.25 \cdot \mathbf{1}_n$ in this case. We estimate good values for the (L-RDE) regularization parameter λ as follows: We use a single randomly chosen image signal from the test set and solve the (L-RDE) optimization problem for values $\lambda = 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}$ to get a first rough estimate. We compare the results visually as well as using the relevance ordering test described below and determine 10^{-1} and 10^{0} as good candidates. We refine the search using values $\lambda = 0.1, 0.2, \ldots, 1.0$ and finally empirically select $\lambda = 0.1$.

B.6.3 Comparison Methods

See above. We use the same settings as in the MNIST experiment for all methods.

Table B.4: **STL10 – Network Architecture.** 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	-	-	-

B.6.4 Relevance Ordering Comparison Test

For the relevance ordering based comparison test we sort components according to their relevance scores (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. Here, we increase the set of fixed components always in groups of size 512 (resulting in 294 steps until all 150528 components are fixed) and use 128 random samples drawn from $\mathcal{U}([0,1]^n)$ per step. This procedure is repeated and the results are averaged over 50 different input signals spaced evenly over the test data set (5 images for each of the 10 classes).

B.7 Statistics from Streaming Data

In this section we describe algorithms for obtaining various sample statistics from batches of a stream of data. This allows the processing of data sets that are too large to be handled at once (due to memory constraints) or that are not immediately completely available (online processing of streaming data). This is used, for example, for the estimation of parameters of the noise distribution $\mathcal{V}$ from training data in the RDE approach.

More precisely, assume that we are given data samples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^n$ where $N \in \mathbb{N}$ is

very large or a priory unknown. Hence, the full data matrix $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \dots & \mathbf{x}_N\end{bmatrix} \in \mathbb{R}^{n\times N}$ can not be processed in its entirety. Instead we assume that it is partitioned into $T \in \mathbb{N}$ batches $\mathbf{X} = \begin{bmatrix}\mathbf{X}_1 & \dots & \mathbf{X}_T\end{bmatrix}$ with $\mathbf{X}_t \in \mathbb{R}^{n\times N_t}$ and $\sum_t N_t = N$. Our goal is to compute statistics of $\mathbf{X}$, such as the sample mean, sample variance, or sample covariance matrix, from the batched data. When N and therefore typically also T is large, it is desirable to obtain these statistics from a single pass through the data batches (if possible). Each algorithm presented in the following is split into a main routine (collecting results from individual batches) and a subroutine that processes a single batch $\mathbf{X}_t$ of data.

B.7.1 Sample Mean, Variance & Covariance

We start with the most simple case of computing the sample mean

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_j,$$

which can easily be rewritten in terms of the batched data as

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{t=1}^{T}\sum_{j=1}^{N_t}\mathbf{x}_{t,j},$$

where $\mathbf{x}_{t,j}$ denotes the j-th column (j-th data sample) in the t-th batch $\mathbf{X}_t$. This leads to Algorithms 7 and 8.

The straight-forward approach to computing the sample variance

$$\mathbf{v} = \frac{1}{N-1}\sum_{j=1}^{N}\left(\mathbf{x}_j - \overline{\mathbf{x}}\right)^2 = \frac{1}{N-1}\left(\sum_{j=1}^{N}\mathbf{x}_j^2 - \overline{\mathbf{x}}^2\right)$$

can not easily be rewritten in a single pass batched form. Instead, it would require two passes through the data: A first pass computes the sample mean $\overline{\mathbf{x}}$ and a second pass sums the squared deviations from that mean.

However, it is possible to combine this into a single pass algorithm[5], as proposed by Chan, Golub, and LeVeque in [CGL79]. They introduce a correction term at each step that compensates a shift of the current estimate of $\overline{\mathbf{x}}$ as new batches of data are being processed. This is presented in Algorithms 9 and 10. Note that Algorithm 9 also computes the sample mean as a byproduct, so that we obtain a combined computation of the sample mean and variance in a single pass through the data batches.

The same correction term idea of Chan, Golub, and LeVeque can also be generalized to computing the sample covariance matrix

$$\mathbf{Q} = \frac{1}{N-1}(\mathbf{X} - \overline{\mathbf{x}}\mathbf{1}_N^\top)(\mathbf{X} - \overline{\mathbf{x}}\mathbf{1}_N^\top)^\top = \frac{1}{N-1}\left(\mathbf{X}\mathbf{X}^\top - \overline{\mathbf{x}}\overline{\mathbf{x}}^\top\right)$$

with a single pass through the batches of data. This results in Algorithms 11 and 12. As before, Algorithm 11 also computes the mean as a byproduct.

[5] Here, single pass refers to the fact that each data batch is processed only once. Within each batch we can use a straight-forward two pass approach, see Algorithm 10. This could also be further refined by recursively applying the batched single pass approach to smaller sub-batches of $\mathbf{X}_t$ if N_t is still large.

Algorithm 7 Compute Mean

Input: data batches $\mathbf{X}_1, \ldots, \mathbf{X}_T$
Output: sample mean $\overline{\mathbf{x}}$

$\boldsymbol{\mu} \leftarrow \mathbf{0}_n$ ▷ initialize accumulator
$c \leftarrow 0$ ▷ initialize counter
for $t \leftarrow 1$ **to** T **do**
    $\boldsymbol{\mu}, c \leftarrow \textsc{BatchMean}(\mathbf{X}_t, N_t, \boldsymbol{\mu}, c)$
end for
return $\frac{1}{c}\boldsymbol{\mu}$ ▷ normalize

Algorithm 8 Process Single Batch (Mean)

function $\textsc{BatchMean}(\mathbf{X}_t, N_t, \boldsymbol{\mu}, c)$
    for $j \leftarrow 1$ **to** N_t **do**
        $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \mathbf{x}_{t,j}$
    end for
    $c \leftarrow c + N_t$
    return $\boldsymbol{\mu}, c$
end function

Algorithm 9 Compute Variance

Input: data batches $\mathbf{X}_1, \ldots, \mathbf{X}_T$
Output: sample variance $\mathbf{v}$

$\boldsymbol{\mu}, \mathbf{v} \leftarrow \mathbf{0}_n, \mathbf{0}_n$ ▷ initialize accumulators
$c \leftarrow 0$ ▷ initialize counter
for $t \leftarrow 1$ **to** T **do**
    $\boldsymbol{\mu}, \mathbf{v}, c \leftarrow \textsc{BatchVar}(\mathbf{X}_t, N_t, \boldsymbol{\mu}, \mathbf{v}, c)$
end for
return $\frac{1}{c-1}\mathbf{v}$ ▷ normalize

Algorithm 10 Process Single Batch (Variance)

function BatchVar($\mathbf{X}_t$, N_t, $\boldsymbol{\mu}$, $\mathbf{v}$, c)
  $\boldsymbol{\mu}_t, \mathbf{v}_t \leftarrow \mathbf{0}_n, \mathbf{0}_n$
  for $j \leftarrow 1$ **to** N_t **do**
    $\boldsymbol{\mu}_t \leftarrow \boldsymbol{\mu}_t + \mathbf{x}_{t,j}$
  end for
  for $j \leftarrow 1$ **to** N_t **do**
    $\mathbf{v}_t \leftarrow \mathbf{v}_t + (\mathbf{x}_{t,j} - \boldsymbol{\mu}_t)^2$
  end for
  $\boldsymbol{\Delta} \leftarrow \frac{1}{c}\boldsymbol{\mu} - \frac{1}{N_t}\boldsymbol{\mu}_t$
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \boldsymbol{\mu}_t$
  $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{v}_t + \frac{cN_t}{c+N_t}\boldsymbol{\Delta}^2$
  $c \leftarrow c + N_t$
  return $\boldsymbol{\mu}$, $\mathbf{v}$, c
end function

Algorithm 11 Compute Covariance

Input: data batches $\mathbf{X}_1, \ldots, \mathbf{X}_T$
Output: sample covariance $\mathbf{Q}$

$\boldsymbol{\mu}, \mathbf{Q} \leftarrow \mathbf{0}_n, \mathbf{0}_{n\times n}$ ▷ initialize accumulators
$c \leftarrow 0$ ▷ initialize counter
for $t \leftarrow 1$ **to** T **do**
  $\boldsymbol{\mu}, \mathbf{Q}, c \leftarrow$ BatchCov($\mathbf{X}_t$, N_t, $\boldsymbol{\mu}$, $\mathbf{Q}$, c)
end for
return $\frac{1}{c-1}\mathbf{Q}$ ▷ normalize

Algorithm 12 Process Single Batch (Covariance)

function BatchCov($\mathbf{X}_t$, N_t, $\boldsymbol{\mu}$, $\mathbf{Q}$, c)
  $\boldsymbol{\mu}_t, \mathbf{Q}_t \leftarrow \mathbf{0}_n, \mathbf{0}_{n\times n}$
  for $j \leftarrow 1$ **to** N_t **do**
    $\boldsymbol{\mu}_t \leftarrow \boldsymbol{\mu}_t + \mathbf{x}_{t,j}$
  end for
  for $j \leftarrow 1$ **to** N_t **do**
    $\mathbf{Q}_t \leftarrow \mathbf{Q}_t + (\mathbf{x}_{t,j} - \boldsymbol{\mu}_t)(\mathbf{x}_{t,j} - \boldsymbol{\mu}_t)^\top$
  end for
  $\boldsymbol{\Delta} \leftarrow \frac{1}{c}\boldsymbol{\mu} - \frac{1}{N_t}\boldsymbol{\mu}_t$
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \boldsymbol{\mu}_t$
  $\mathbf{Q} \leftarrow \mathbf{Q} + \mathbf{Q}_t + \frac{cN_t}{c+N_t}\boldsymbol{\Delta}\boldsymbol{\Delta}^\top$
  $c \leftarrow c + N_t$
  return $\boldsymbol{\mu}$, $\mathbf{Q}$, c
end function

B.7.2 Low-Rank Approximations of Covariance

The best rank $r \in \mathbb{N}$ ($r < n$) approximation of $\mathbf{X} \in \mathbb{R}^{n \times N}$ (with respect to the Frobenius norm) is given by a truncated singular value decomposition (SVD). Denote the full SVD as

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

with orthogonal matrices $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{V} \in \mathbb{R}^{N \times N}$ and diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{n \times N}$ containing the singular values of $\mathbf{X}$ ordered from largest to smallest as diagonal entries. The truncated SVD of rank (at most) r, also denoted r-SVD from now on, is obtained by setting all but the r largest singular values in $\mathbf{\Sigma}$ to zero, or equivalently

$$\mathbf{X} \approx \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^\top,$$

where $\widetilde{\mathbf{U}} \in \mathbb{R}^{n \times r}$, $\widetilde{\mathbf{V}} \in \mathbb{R}^{N \times r}$ are the first r columns of $\mathbf{U}$ and $\mathbf{V}$ respectively and $\widetilde{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$ is the upper-left $r \times r$ sub-matrix of $\mathbf{\Sigma}$. Unlike the computation of the mean, variance, and covariance matrix, such a low-rank approximation can generally not be computed exactly from streaming batches of data. In other words, combining the individual low-rank approximations of two batches $\mathbf{X}_1$ and $\mathbf{X}_2$ is not equal to the best low-rank approximation of the combined data $\begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}$. However, it can still be a good approximation. We present an efficient algorithm due to Řehůřek. It relies on reduced QR-factorizations for merging two SVDs. See [Řeh11] for details and a discussion of the truncation effects of intermediate steps on the final approximation. Given the fact, that the sample covariance matrix is closely related to $\mathbf{X}\mathbf{X}^\top = \mathbf{V}\mathbf{\Sigma}\mathbf{\Sigma}^\top\mathbf{V}^\top$, we will state all steps of the algorithm only in terms of $\mathbf{\Sigma}$ and $\mathbf{V}$. Even though left singular vectors $\mathbf{U}$ are computed as intermediate steps for each data batch, it is not necessary to keep track of them in the merging step of two SVDs. We begin with an algorithm for computing a low-rank approximation of the autocorrelation matrix $\mathbf{X}\mathbf{X}^\top$.

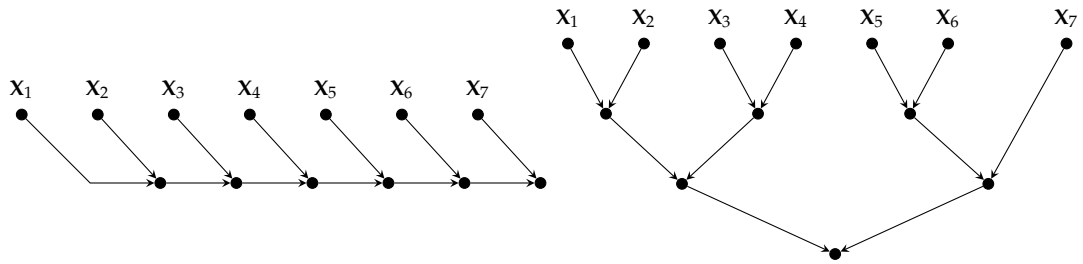
Algorithm 13 Compute Low-Rank Autocorrelation

Input: data batches $\mathbf{X}_1, \ldots, \mathbf{X}_T$, rank r
Output: approximate r-SVD of autocorrelation $\frac{1}{N-1}\mathbf{X}\mathbf{X}^\top$

$\mathbf{\Sigma}, \mathbf{V} \leftarrow \mathbf{0}_{r \times r}, \mathbf{0}_{n \times r}$ ▷ initialize accumulators
$c \leftarrow 0$ ▷ initialize counter
for $t \leftarrow 1$ **to** T **do**
    $\mathbf{\Sigma}, \mathbf{V}, c \leftarrow \textsc{BatchCorr}(\mathbf{X}_t, N_t, \mathbf{\Sigma}, \mathbf{V}, c, r)$
end for
return $\frac{1}{c-1}\mathbf{V}\mathbf{\Sigma}\mathbf{\Sigma}^\top\mathbf{V}$ ▷ normalize

Algorithm 14 Process Single Batch (Autocorrelation)

function $\textsc{BatchCorr}(\mathbf{X}_t, N_t, \mathbf{\Sigma}, \mathbf{V}, c, r)$
    $\mathbf{U}_t, \mathbf{\Sigma}_t, \mathbf{V}_t \overset{r\text{-SVD}}{\leftarrow} \mathbf{X}_t$
    $\mathbf{\Sigma}, \mathbf{V} \leftarrow \textsc{Merge}(\mathbf{\Sigma}, \mathbf{V}, \mathbf{\Sigma}_t, \mathbf{V}_t, r)$
    $c \leftarrow c + N_t$
    return $\mathbf{\Sigma}, \mathbf{V}, c$
end function



(a) (b)

Figure B.20: **Merge Orders.** Illustration of two possible orders in which intermediates results of the low-rank autocorrelation algorithm (example is shown for $T = 7$ batches) can be combined: (a) sequential and (b) hierarchical.

Algorithm 15 Merge Low-Rank SVD

function MERGE($\mathbf{\Sigma}_1$, $\mathbf{V}_1$, $\mathbf{\Sigma}_2$, $\mathbf{V}_2$, r)
  $\mathbf{Z} \leftarrow \mathbf{V}_1^\top \mathbf{V}_2$
  $\mathbf{Q}, \mathbf{R} \overset{\text{reduced-QR}}{\leftarrow} \mathbf{V}_2 - \mathbf{V}_1 \mathbf{Z}$
  $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V} \overset{r\text{-SVD}}{\leftarrow} \begin{bmatrix} \mathbf{\Sigma}_1 & \mathbf{Z}\mathbf{\Sigma}_2 \\ \mathbf{0}_{r \times r} & \mathbf{R}\mathbf{\Sigma}_2 \end{bmatrix}$
  $\mathbf{V} \leftarrow \begin{bmatrix} \mathbf{V}_1 & \mathbf{Q} \end{bmatrix} \mathbf{U}$
  return $\mathbf{\Sigma}$, $\mathbf{V}$
end function

The algorithm as presented above combines the results of processed batches in a sequential "linear" manner where each new batch is merged with a running accumulation of all previous batches, as illustrated in Figure B.20(a). This puts more emphasis on the influence of later batches toward the overall approximation, as discussed in [Řeh11]. A more balanced approach proposed (but not further investigated) in [Řeh11] combines the results of processed batched in a pairwise "binary tree" manner instead, as illustrated in Figure B.20(b). This can be achieved by pushing intermediate results onto a stack (increasing memory requirements compared to the sequential approach) and retrieving and merging them at the appropriate time.

Finally, low-rank approximations to the covariance matrix can be obtained in a two-pass approach: First compute the sample mean $\bar{\mathbf{x}}$ according to Algorithm 7. Second, compute the autocorrelation on the shifted data batches $\mathbf{X}_t - \bar{\mathbf{x}}\mathbf{1}_{N_t}^\top$ according to Algorithm 13. It is not immediately clear if a simultaneous estimation of the mean and low-rank approximation to the covariance matrix can be achieved using a correction term $\mathbf{\Delta}$ at each step similar to Algorithms 10 and 12.

A comparison of sequential and hierarchical low-rank approximations to the covariance matrix of the MNIST dataset is presented in Figure B.21. It shows the relative Frobenius norm errors of the approximations to the exact covariance matrix (left) and the singular values of the approximations at five different selected ranks compared to the exact singular values. In both cases we do not see a significant difference between the sequential and hierarchical approximations for the MNIST dataset. In our experiments in Section 4.4

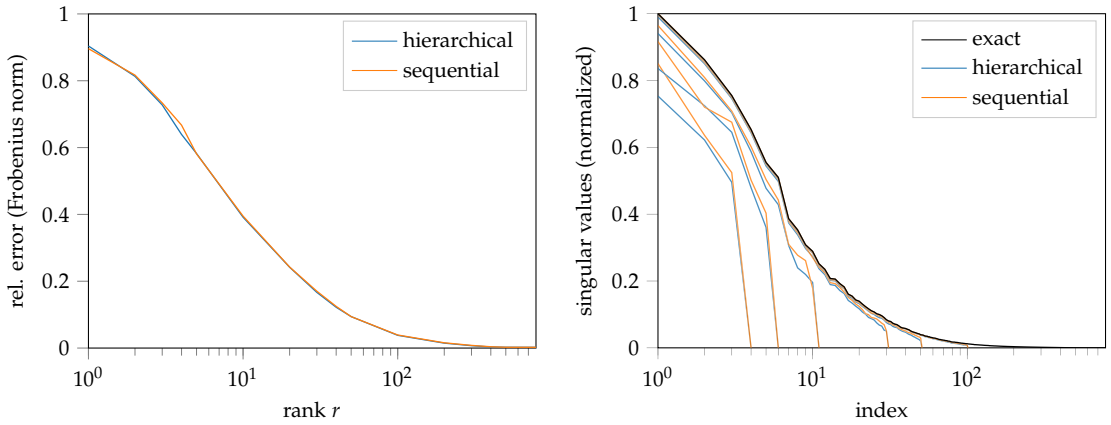


Figure B.21: **MNIST – Low-Rank Approximations.** (a) Relative Frobenius norm errors of low-rank approximations to the MNIST covariance matrix obtained with the hierarchical and the sequential merge order. (b) Singular values of low-rank approximations to the MNIST covariance matrix obtained with the hierarchical and the sequential merge order at selected ranks $r = 3, 5, 10, 30, 50, 100$ in comparison to the exact singular values.

and Appendix B.2 we use the hierarchical variant to estimate the reference distribution $\mathcal{V}$ for the An8flower, MNIST, and STL-10 datasets.

C

Deferred Proofs of Chapter 5

C.1 Metric Spaces and Hausdorff Dimension

In this section we will proof Lemma 5.15. 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 $\mathbb{R}^m$ 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].

Lemma C.1. *Let $(\mathcal{X}, d_\mathcal{X})$ and $(\mathcal{Y}, d_\mathcal{Y})$ be metric spaces, $A, B \subseteq \mathcal{X}$ Borel sets, $(A_i)_{i\in\mathbb{N}}$ a countable collection of Borel sets $A_i \subseteq \mathcal{X}$, and $\Phi\colon \mathcal{X} \to \mathcal{Y}$ (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) $\dim_H(\Phi(\mathcal{X})) \leq \dim_H(\mathcal{X})$.

If $(\mathcal{X}, d_\mathcal{X})$ is a separable space, then the last part can be generalized to locally Lipschitz continuous maps.

Lemma C.2. *Let $(\mathcal{X}, d_\mathcal{X})$ and $(\mathcal{Y}, d_\mathcal{Y})$ be metric spaces, $\Phi\colon \mathcal{X} \to \mathcal{Y}$ locally Lipschitz continuous, and $\mathcal{X}$ separable. Then*

$$\dim_H(\Phi(\mathcal{X})) \leq \dim_H(\mathcal{X}).$$

Proof. For every $\mathbf{x} \in \mathcal{X}$ there exists an open neighborhood $U_\mathbf{x}$ of $\mathbf{x}$ such that Φ restricted to $U_\mathbf{x}$ is Lipschitz continuous. Also $(U_\mathbf{x})_{\mathbf{x}\in\mathcal{X}}$ is an open cover of $\mathcal{X}$. As $\mathcal{X}$ is separable and thus Lindelöf there exists a countable subcover $(U_{\mathbf{x}_i})_{i\in\mathbb{N}}$. From Lemma C.1 we conclude

$$\dim_H(\Phi(U_{\mathbf{x}_i})) \leq \dim_H(U_{\mathbf{x}_i}) \leq \dim_H(\mathcal{X})$$

for all $i \in \mathbb{N}$. Since $\Phi(\mathcal{X}) = \Phi(\bigcup_{i\in\mathbb{N}} U_{\mathbf{x}_i}) = \bigcup_{i\in\mathbb{N}} \Phi(U_{\mathbf{x}_i})$, we can again use Lemma C.1 to conclude

$$\dim_H(\Phi(\mathcal{X})) = \dim_H(\bigcup_{i\in\mathbb{N}} \Phi(U_{\mathbf{x}_i})) \leq \sup_{i\in\mathbb{N}} \dim_H(\Phi(U_{\mathbf{x}_i})) \leq \dim_H(\mathcal{X}). \qquad \square$$

We now come to the special case of the Euclidean space $\mathbb{R}^m$. Every Borel subset of $\mathbb{R}^m$ with non-empty interior has Hausdorff dimension m, again see for example [Edg08]. A direct consequence of this is the following result.

Lemma C.3. *We have* $\dim_H(\mathbb{R}^m) = m$ *and* $\dim_H(\mathbb{R}^m_+) = m$.

We are now ready to prove Lemma 5.15.

Proof of Lemma 5.15. Since $\Omega \subseteq \mathbb{R}^d$ is a separable metric space, Φ is locally Lipschitz continuous, and $\Phi(\Omega) \subseteq \mathbb{R}^m$ has non-empty interior, we can use Lemmas C.2 and C.3 to obtain

$$m = \dim_H(\Phi(\Omega)) \leq \dim_H(\Omega) \leq \dim_H(\mathbb{R}^d) = d. \qquad \square$$

C.2 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 5.28.

We start by 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 to glue all these h_n together we also want to assure $h_n(0) = h_n(1) = \mathbf{0}_n$ for all $n \geq 2$. Thus we define

$$h_n(t) = \begin{cases} 4nt(2g_n(0) - \mathbf{1}_n), & t \in [0, \frac{1}{4}], \\ n(2g_n(2t - \frac{1}{2}) - \mathbf{1}_n), & t \in [\frac{1}{4}, \frac{3}{4}], \\ 4n(1-t)(2g_n(1) - \mathbf{1}_n), & t \in [\frac{3}{4}, 1]. \end{cases}$$

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

$$\Gamma(t) = \begin{cases} h_{\lfloor t \rfloor}(t - \lfloor t \rfloor) \times \{0\}^\infty, & t \geq 2, \\ \{0\}^\infty, & t < 2. \end{cases}$$

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 $h_n(0) = h_n(1) = \mathbf{0}_n$ 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

$$\mathbb{R}^\infty = \bigcup_{n \geq 2} [-n, n]^n \times \{0\}^\infty = \bigcup_{n \geq 2} \Gamma([n, n+1]).$$

C.3 Restricting the Set of Weight Matrices

In this section we briefly discuss some variations of characterizations 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.

C.3.1 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 5.3.1. For all other matrix restrictions it remains to be investigated whether our approach can be adapted.

C.3.2 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 finite collections $\mathcal{F}$ instead.

We begin with the extreme case, in which the collection contains only a single measurable function $\mathcal{F} = \{\Phi \colon \mathbb{R}^n \to \mathbb{R}^n\}$, for example a ReLU layer $\Phi(\mathbf{x}) = \varrho(\mathbf{W}\mathbf{x} + \mathbf{b})$ with a fixed weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$. As in Sections 5.3.1 and 5.3.3 we can construct an invariant family of distributions starting with a prototype distribution $\mu_0 \in \mathcal{D}(\mathbb{R}^n)$. Next, we iteratively define

$$\mu_k = \Phi_* \mu_{k-1}, \quad k \in \mathbb{N},$$

and using $\eta(t) = t - \lfloor t \rfloor$ also the intermediate interpolations

$$\mu_t = (1 - \eta(t))\mu_{\lfloor t \rfloor} + \eta(t)\mu_{\lfloor t \rfloor + 1}, \quad t \geq 0. \tag{C.1}$$

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

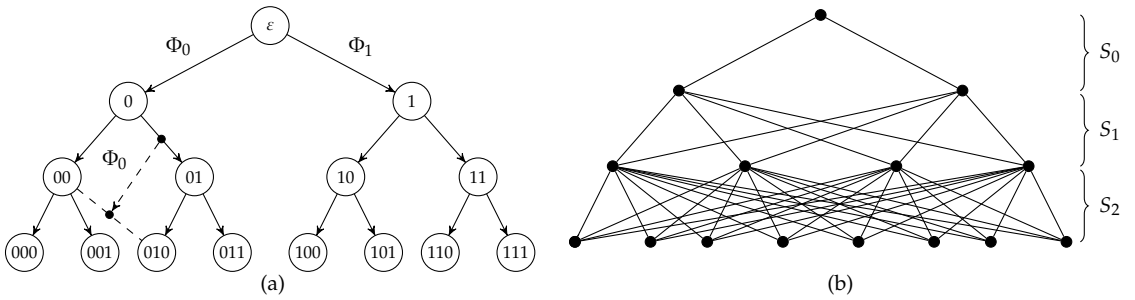


Figure C.1: **Graphical Representation of Interpolated Measures.** (a) Consecutively applying either Φ_0 or Φ_1 to a prototype measure μ results in an infinite tree structure (only the first three levels are shown). Transforming an intermediate (interpolated) point by Φ_0 or Φ_1 (dashed) results in a point outside the tree structure. (b) Adding additional edges to represent also the transformations of all interpolated points results in a layered bipartite graph in which all vertices of consecutive levels are connected. All subgraphs S_k induced by two consecutive levels of vertices are complete bipartite graphs.

distributions is always bounded by one. First, we consider the case $\lfloor t_2 \rfloor = \lfloor t_1 \rfloor = k$ for some $k \in \mathbb{N}$. Then

$$\begin{aligned} d_P(\mu_{t_2}, \mu_{t_1}) &\leq \|\mu_{t_2} - \mu_{t_1}\|_{\mathrm{TV}} \\ &\leq \|(\eta(t_2) - \eta(t_1))(\mu_{k+1} - \mu_k)\|_{\mathrm{TV}} \\ &\leq |\eta(t_2) - \eta(t_1)| \|\mu_{k+1} - \mu_k\|_{\mathrm{TV}} \\ &\leq |t_2 - t_1|(\|\mu_{k+1}\|_{\mathrm{TV}} + \|\mu_k\|_{\mathrm{TV}}) \\ &= 2|t_2 - t_1|, \end{aligned}$$

where $\|\cdot\|_{\mathrm{TV}}$ is the total variation norm whose induced metric upper bounds the Prokhorov metric, see for example [Hub81, Chp. 2].

Second, we consider the case $\lfloor t_2 \rfloor = \lfloor t_1 \rfloor + 1 = k$. Then

$$\mu_{t_2} - \mu_{t_1} = (1 - \eta(t_2) - \eta(t_1))\mu_k + \eta(t_2)\mu_{k+1} - (1 - \eta(t_1))\mu_{k-1}$$

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 $\Phi_* \mu_t = \mu_{t+1}$ for any $t \geq 0$, which means that it is Φ-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 $\mathcal{F} = \{\Phi_0, \Phi_1\}$ of two functions. Again, we start by choosing an arbitrary prototype measure, which we will simply denote μ in this case. For any finite binary string $\mathbf{z} \in \{0,1\}^*$ of arbitrary length $k \in \mathbb{N} \cup \{0\}$, we define

$$\mu_{\mathbf{z}} = (\Phi_{z_k} \circ \cdots \circ \Phi_{z_1})_* \mu,$$

with the convention $\mu_\varepsilon = \mu$ for the empty string ε. The procedure of obtaining measures this way can be associated to a perfect binary tree of infinite depth with root ε. A binary string is the child of another string in this tree if it extends the other by exactly one digit 0 or 1, see Figure C.1. Note that different strings could result in the same measure so that the set of measures $\{\mu_\mathbf{z}\}_{\mathbf{z}\in\{0,1\}^*}$ is not in one-to-one correspondence to the vertices of the tree. However, each measure $\mu_\mathbf{z}$ is represented by at least one vertex in the tree.

In order to find a parametrization 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 parametrization will extend the interpolation idea from (C.1) and results from a walk through all nodes of the graph.

Let $\mathrm{len}(\mathbf{z})$ denote the length of a binary string $\mathbf{z}$. We define the graph $G = (V, E)$ with vertices $V = \{0,1\}^*$ and edges $E = \{(\mathbf{z}_1, \mathbf{z}_2) \in V \times V \,:\, |\mathrm{len}(\mathbf{z}_1) - \mathrm{len}(\mathbf{z}_2)| = 1\}$, i.e., every vertex is a binary string and all strings of consecutive lengths are connected by an edge. The resulting graph is a bipartite graph where the vertices corresponding to binary strings of even or odd length respectively are independent sets. Let us define subgraphs S_k of G for $k \in \mathbb{N} \cup \{0\}$ that are induced by the vertex sets

$$V_k = \{0,1\}^k \cup \{0,1\}^{k+1},$$

where the convention $\{0,1\}^0 = \{\varepsilon\}$ for $k = 0$ (with the empty string ε) is used. The resulting subgraphs are complete bipartite graphs, see Figure C.1. 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 a walk can pass through an edge multiple times this is easy to construct: Each subgraph S_k 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_k starts and ends in $\mathbf{0}_k$ (the string of k zeros) and denote it $\mathcal{W}_k$. Let $\sqcap$ symbolize the concatenation of two walks. We set

$$\mathcal{W} = \bigsqcap_{k=0}^{\infty} \mathcal{W}_k,$$

which is possible since in G the two vertices $\mathbf{0}_k$ and $\mathbf{0}_{k+1}$ are connected by an edge for every $k \in \mathbb{N}$. Denoting the i-th vertex visited by $\mathcal{W}$ as $\mathcal{W}(i)$ we can now define

$$\mu_t = (1 - \eta(t))\mu_{\mathcal{W}(\lfloor t \rfloor)} + \eta(t)\mu_{\mathcal{W}(\lfloor t \rfloor + 1)}, \tag{C.2}$$

with $\eta(t)$ as before.

To see that this fulfills our criteria consider the following. All distributions in the parametrized family are of the form

$$(1-s)\mu_{\mathbf{z}_1} + s\mu_{\mathbf{z}_2}, \quad \text{with} \quad \mathbf{z}_1 \in \{0,1\}^k, \mathbf{z}_2 \in \{0,1\}^{k+1}, k \in \mathbb{N} \cup \{0\}, s \in [0,1].$$

Transforming it by the function Φ_j for $j \in \{0,1\}$ results in the pushforward measure

$$(1-s)\mu_{\mathbf{z}_1'} + s\mu_{\mathbf{z}_2'}, \quad \text{with} \quad \mathbf{z}_1' = (\mathbf{z}_1, j), \mathbf{z}_2' = (\mathbf{z}_2, j).$$

But since $\mathbf{z}_1'$ and $\mathbf{z}_2'$ still have length difference one, they share an edge in S_{k+1}. The walk $\mathcal{W}$ passes through all edges, so we can define $i \in \mathbb{N}$ as the smallest number where either

$\mathcal{W}(i) = \mathbf{z}'_1$ and $\mathcal{W}(i+1) = \mathbf{z}'_2$ or $\mathcal{W}(i) = \mathbf{z}'_2$ and $\mathcal{W}(i+1) = \mathbf{z}'_1$. Let us without loss of generality assume that the former holds. Then

$$(1-s)\mu_{\mathbf{z}'_1} + s\mu_{\mathbf{z}'_2} = \mu_{i+s},$$

which shows $t \mapsto \mu_t$ is $\mathcal{F}$-invariant. The Lipschitz continuity follows analogously to the case with only a single function above.

The same argument can be used for more than two functions in $\mathcal{F}$. In this case the binary tree is simply replaced by an $|\mathcal{F}|$-ary tree. It can even be extended to countably many different measurable functions. In this case, 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

$$\mathcal{W} = \prod_{i=1}^{\infty} \mathcal{W}_i.$$

It follows that every edge of every graph in the sequence will eventually be reached. Defining the parametrization as in (C.2), ensures that both $\mathcal{F}$-invariance as well as Lipschitz continuity hold.

D

Additions to Chapter 6

Table D.1: **Scenario A1 – CS with 1D Signals.** A numerical representation of the results of Figure 6.4(e), including the additional methods UNetFL and Tira. The smallest relative error per noise level is highlighted in bold.

rel. noise – adversarial		0.0%	0.1%	0.5%	1.0%	2.0%	4.0%	6.0%
TV[η]	rel. ℓ_2-err. [%]	**0.00±0.00**	**0.32±0.08**	1.66±0.42	3.36±0.86	6.63±1.67	12.26±2.57	17.21±2.98
UNet	rel. ℓ_2-err. [%]	2.53±1.97	2.67±2.01	3.36±2.11	4.49±2.31	7.29±2.52	13.15±3.24	18.27±3.58
UNetFL	rel. ℓ_2-err. [%]	2.01±1.70	2.14±1.73	2.82±1.84	3.95±2.01	6.46±2.21	11.91±2.54	16.98±2.72
Tira	rel. ℓ_2-err. [%]	1.22±1.15	1.33±1.18	1.95±1.33	3.05±1.64	5.90±2.23	11.97±3.13	17.18±3.27
TiraFL	rel. ℓ_2-err. [%]	0.98±0.88	1.10±0.92	1.73±1.15	2.74±1.42	5.32±1.89	**11.07±2.82**	16.43±3.42
ItNet	rel. ℓ_2-err. [%]	0.45±0.18	0.52±0.18	**0.93±0.17**	**1.80±0.80**	**4.50±1.90**	11.62±3.67	**16.42±3.70**

Table D.2: **Scenario A1 – CS with 1D Signals.** A numerical representation of the results of Figure 6.4(f), including the additional methods UNetFL and Tira. The smallest relative error per noise level is highlighted in bold.

rel. noise – Gaussian		0.0%	0.1%	0.5%	1.0%	2.0%	4.0%	6.0%
TV[η]	rel. ℓ_2-err. [%]	**0.00±0.00**	**0.18±0.06**	0.90±0.30	1.79±0.60	3.61±1.18	6.84±2.15	9.74±2.64
UNet	rel. ℓ_2-err. [%]	2.53±1.97	2.58±1.99	2.79±2.03	3.09±2.07	3.82±2.11	5.67±2.25	7.70±2.49
UNetFL	rel. ℓ_2-err. [%]	2.01±1.70	2.05±1.71	2.24±1.75	2.53±1.79	3.23±1.91	4.99±2.05	7.03±2.21
Tira	rel. ℓ_2-err. [%]	1.22±1.15	1.26±1.16	1.43±1.21	1.71±1.25	2.35±1.43	4.15±1.79	6.42±2.23
TiraFL	rel. ℓ_2-err. [%]	0.98±0.88	1.02±0.90	1.20±0.95	1.48±1.05	2.10±1.24	3.86±1.63	5.98±2.19
ItNet	rel. ℓ_2-err. [%]	0.45±0.18	0.47±0.18	**0.59±0.17**	**0.80±0.17**	**1.38±0.50**	**2.91±0.99**	**5.33±2.15**

Table D.3: **Scenario A2 – CS with MNIST.** A numerical representation of the results of Figure 6.9(c), including the additional methods UNetFL and Tira. The smallest relative error per noise level is highlighted in bold.

rel. noise – adversarial		0.0%	0.5%	1.0%	3.0%	5.0%	7.5%	10.0%
TV[η]	rel. ℓ_2-err. [%]	15.32±10.13	18.18±9.83	20.50±9.60	28.68±8.77	35.92±8.45	43.87±7.95	50.85±7.35
UNet	rel. ℓ_2-err. [%]	9.79±2.14	10.24±2.17	10.71±2.19	12.96±2.37	15.71±2.58	20.23±2.91	25.08±3.15
UNetFL	rel. ℓ_2-err. [%]	7.88±1.42	8.23±1.42	8.60±1.42	10.23±1.42	12.13±1.45	14.97±1.47	18.28±1.51
Tira	rel. ℓ_2-err. [%]	8.56±1.77	8.95±1.78	9.37±1.79	11.21±1.81	13.50±1.80	16.87±1.95	20.66±2.11
TiraFL	rel. ℓ_2-err. [%]	7.64±1.38	7.99±1.37	8.36±1.36	9.99±1.34	11.94±1.34	14.91±1.28	18.25±1.30
ItNet	rel. ℓ_2-err. [%]	**2.47±0.58**	**2.96±0.60**	**3.53±0.60**	**6.26±0.59**	**9.35±0.72**	**13.62±1.31**	**18.06±1.77**

Table D.4: **Scenario A2 – CS with MNIST.** A numerical representation of the results of Figure 6.9(d), including the additional methods UNetFL and Tira. The smallest relative error per noise level is highlighted in bold.

rel. noise – Gaussian		0.0%	0.5%	1.0%	3.0%	5.0%	7.5%	10.0%
TV[η]	rel. ℓ_2-err. [%]	15.32±10.13	16.55±9.71	17.48±9.31	21.52±8.19	25.53±7.75	30.00±7.48	34.20±7.18
UNet	rel. ℓ_2-err. [%]	9.79±2.14	9.87±2.15	9.96±2.14	10.36±2.13	10.86±2.18	11.54±2.16	12.37±2.11
UNetFL	rel. ℓ_2-err. [%]	7.88±1.42	7.88±1.42	7.89±1.42	7.99±1.41	8.16±1.40	8.49±1.38	8.92±1.36
Tira	rel. ℓ_2-err. [%]	8.56±1.77	8.56±1.77	8.57±1.76	8.67±1.75	8.85±1.72	9.17±1.69	9.59±1.65
TiraFL	rel. ℓ_2-err. [%]	7.64±1.38	7.70±1.38	7.77±1.37	8.12±1.35	8.52±1.36	9.18±1.35	9.88±1.35
ItNet	rel. ℓ_2-err. [%]	**2.47±0.58**	**2.58±0.59**	**2.72±0.58**	**3.60±0.58**	**4.65±0.66**	**6.00±0.73**	**7.32±0.80**

Table D.5: **Scenario B1 – Fourier Measurements with Ellipses.** A numerical representation of the results of Figure 6.10(c), including the additional methods UNetFL and Tira. The best relative error/PSNR/SSIM per noise level is highlighted in bold. Note that the high SSIM values for TV[η] for 5% and 8% can be explained by the fact that adversarial perturbations for TV[η] cause point-like artifacts, see the zoomed region in Figure 6.11. In contrast to the PSNR, the SSIM seems to be less sensitive to such types of errors.

rel. noise – adversarial		0.0%	0.5%	1.0%	2.0%	3.0%	5.0%	8.0%
TV[η]	rel. ℓ_2-err. [%]	**0.44±0.11**	**2.51±0.24**	4.34±0.35	7.35±0.45	9.96±0.46	14.19±0.44	20.72±0.63
	PSNR	**60.00±3.26**	**44.73±2.20**	39.98±2.16	35.38±2.00	32.74±1.91	29.66±1.82	26.37±1.93
	SSIM	**1.00±0.00**	0.99±0.00	0.98±0.01	0.96±0.02	0.94±0.03	**0.93±0.03**	**0.93±0.02**
UNet	rel. ℓ_2-err. [%]	2.94±0.63	4.27±0.55	5.70±0.53	8.38±0.52	10.88±0.53	15.20±0.70	20.41±0.96
	PSNR	43.52±2.06	40.15±1.79	37.61±1.72	34.24±1.67	31.97±1.66	29.07±1.69	26.51±1.78
	SSIM	0.99±0.01	0.98±0.01	0.97±0.01	0.96±0.01	0.94±0.02	0.91±0.02	0.85±0.03
UNetFL	rel. ℓ_2-err. [%]	2.72±0.50	4.12±0.44	5.57±0.43	8.35±0.43	10.97±0.43	15.58±0.67	21.03±1.02
	PSNR	44.13±2.29	40.46±1.96	37.80±1.87	34.28±1.76	31.90±1.73	28.85±1.76	26.25±1.82
	SSIM	0.99±0.00	0.99±0.00	0.98±0.01	0.97±0.01	0.95±0.02	0.91±0.03	0.85±0.04
Tira	rel. ℓ_2-err. [%]	1.74±0.37	3.33±0.33	4.85±0.37	7.73±0.42	10.42±0.53	15.01±0.74	20.39±0.99
	PSNR	48.05±2.50	42.27±1.93	39.01±1.85	34.94±1.80	32.35±1.76	29.18±1.76	26.52±1.83
	SSIM	1.00±0.00	0.99±0.00	**0.99±0.00**	0.97±0.01	0.95±0.02	0.91±0.03	0.87±0.04
TiraFL	rel. ℓ_2-err. [%]	1.75±0.39	3.42±0.34	4.94±0.41	7.82±0.44	10.54±0.51	15.14±0.69	20.55±0.95
	PSNR	48.05±2.58	42.05±1.93	38.85±1.86	34.85±1.83	32.24±1.80	29.10±1.75	26.45±1.81
	SSIM	1.00±0.00	**0.99±0.00**	0.99±0.01	0.97±0.01	0.95±0.02	0.91±0.03	0.87±0.04
ItNet	rel. ℓ_2-err. [%]	1.45±0.29	2.81±0.28	**4.21±0.32**	**6.87±0.37**	**9.37±0.40**	**13.65±0.49**	**18.98±0.65**
	PSNR	49.63±1.80	43.76±1.62	**40.23±1.64**	**35.97±1.63**	**33.27±1.67**	**30.00±1.64**	**27.13±1.65**
	SSIM	0.99±0.00	0.99±0.00	0.98±0.00	**0.97±0.01**	**0.96±0.01**	0.92±0.02	0.88±0.03

Table D.6: **Scenario B1 – Fourier Measurements with Ellipses.** A numerical representation of the results of Figure 6.10(d), including the additional methods UNetFL and Tira. The best relative error/PSNR/SSIM per noise level is highlighted in bold.

rel. noise – Gaussian		0.0%	0.5%	1.0%	2.0%	3.0%	5.0%	8.0%
TV[η]	rel. ℓ_2-err. [%]	**0.44±0.11**	**0.80±0.12**	**1.18±0.15**	1.93±0.25	2.64±0.35	3.90±0.50	5.52±0.58
	PSNR	**60.00±3.26**	**54.73±2.39**	**51.29±2.44**	47.08±2.46	44.33±2.47	40.94±2.25	37.91±2.17
	SSIM	**1.00±0.00**	**1.00±0.00**	0.99±0.00	0.98±0.01	0.96±0.02	0.96±0.02	0.92±0.04
UNet	rel. ℓ_2-err. [%]	2.94±0.63	2.94±0.63	2.95±0.63	3.00±0.63	3.07±0.62	3.28±0.61	3.72±0.61
	PSNR	43.52±2.06	43.50±2.06	43.47±2.06	43.33±2.07	43.12±2.07	42.51±2.09	41.41±2.12
	SSIM	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.01	0.99±0.00	0.98±0.00
UNetFL	rel. ℓ_2-err. [%]	2.72±0.50	2.73±0.50	2.74±0.50	2.80±0.51	2.88±0.51	3.13±0.53	3.61±0.55
	PSNR	44.13±2.29	44.12±2.29	44.07±2.29	43.90±2.28	43.64±2.28	42.92±2.26	41.65±2.20
	SSIM	0.99±0.00	0.99±0.00	0.99±0.00	0.99±0.00	0.99±0.00	0.99±0.00	0.99±0.00
Tira	rel. ℓ_2-err. [%]	1.74±0.37	1.75±0.37	1.77±0.37	1.83±0.38	1.92±0.38	2.19±0.40	2.70±0.45
	PSNR	48.05±2.50	48.02±2.50	47.95±2.49	47.65±2.48	47.20±2.46	46.03±2.41	44.19±2.36
	SSIM	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	0.99±0.00	0.99±0.00
TiraFL	rel. ℓ_2-err. [%]	1.75±0.39	1.76±0.39	1.77±0.39	1.83±0.40	1.92±0.40	2.19±0.42	2.70±0.46
	PSNR	48.05±2.58	48.02±2.57	47.94±2.57	47.66±2.55	47.20±2.54	46.04±2.47	44.21±2.39
	SSIM	1.00±0.00	1.00±0.00	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**0.99±0.00**
ItNet	rel. ℓ_2-err. [%]	1.45±0.29	1.46±0.29	1.47±0.29	**1.53±0.30**	**1.62±0.32**	**1.92±0.37**	**2.50±0.45**
	PSNR	49.63±1.80	49.60±1.80	49.52±1.82	**49.19±1.89**	**48.65±1.98**	**47.18±2.18**	**44.89±2.27**
	SSIM	0.99±0.00	0.99±0.00	0.99±0.00	0.99±0.00	0.99±0.00	0.99±0.00	0.99±0.00

Table D.7: **Case Study C – fastMRI.** A numerical representation of the results of Figure 6.16(c), including the additional methods UNetFL and Tira. The best relative error/PSNR/SSIM per noise level is highlighted in bold.

rel. noise – adversarial		0.0%	0.2%	0.5%	1.0%	1.5%	2.0%	2.5%
TV[η]	rel. ℓ_2-err. [%]	8.39±1.38	8.89±1.40	9.35±1.41	10.34±1.41	11.35±1.42	12.35±1.43	12.96±1.44
	PSNR	31.70±1.47	31.18±1.40	30.73±1.34	29.85±1.22	29.02±1.10	28.28±1.02	27.85±0.98
	SSIM	0.78±0.04	0.77±0.04	0.76±0.04	0.74±0.04	0.72±0.04	0.70±0.04	0.68±0.04
UNet	rel. ℓ_2-err. [%]	8.18±1.27	8.29±1.27	8.41±1.27	8.67±1.26	8.99±1.25	9.38±1.22	9.84±1.18
	PSNR	31.90±1.38	31.79±1.37	31.66±1.35	31.38±1.30	31.06±1.23	30.69±1.15	30.26±1.06
	SSIM	0.80±0.04	0.80±0.03	0.79±0.03	0.79±0.03	0.78±0.03	0.78±0.03	0.77±0.03
UNetFL	rel. ℓ_2-err. [%]	8.23±1.28	8.35±1.28	8.47±1.28	8.75±1.27	9.10±1.25	9.51±1.21	10.01±1.17
	PSNR	31.85±1.39	31.72±1.37	31.59±1.34	31.30±1.29	30.96±1.22	30.56±1.13	30.10±1.04
	SSIM	0.79±0.04	0.79±0.04	0.79±0.04	0.78±0.03	0.78±0.03	0.77±0.03	0.77±0.03
Tira	rel. ℓ_2-err. [%]	7.97±1.26	8.10±1.26	8.24±1.26	8.58±1.25	9.00±1.21	9.52±1.17	10.16±1.12
	PSNR	32.13±1.41	31.99±1.39	31.84±1.36	31.48±1.30	31.05±1.19	30.54±1.09	29.97±0.97
	SSIM	0.80±0.03	0.80±0.03	0.80±0.03	0.79±0.03	0.79±0.03	0.78±0.03	0.77±0.03
TiraFL	rel. ℓ_2-err. [%]	7.98±1.27	8.11±1.27	8.26±1.27	8.60±1.27	9.03±1.24	9.55±1.20	10.19±1.15
	PSNR	32.12±1.42	31.98±1.40	31.82±1.37	31.46±1.31	31.03±1.22	30.52±1.11	29.95±1.00
	SSIM	0.80±0.03	0.80±0.03	0.80±0.03	0.79±0.03	0.78±0.03	0.78±0.03	0.77±0.03
ItNet	rel. ℓ_2-err. [%]	**7.08±1.20**	**7.21±1.20**	**7.35±1.19**	**7.67±1.17**	**8.08±1.13**	**8.59±1.10**	**9.20±1.07**
	PSNR	**33.18±1.52**	**33.02±1.49**	**32.85±1.45**	**32.45±1.35**	**31.99±1.23**	**31.45±1.12**	**30.84±1.02**
	SSIM	**0.82±0.04**	**0.82±0.04**	**0.81±0.04**	**0.81±0.03**	**0.80±0.03**	**0.79±0.03**	**0.78±0.03**

Table D.8: **Case Study C – fastMRI.** A numerical representation of the results of Figure 6.16(d), including the additional methods UNetFL and Tira. The best relative error/PSNR/SSIM per noise level is highlighted in bold.

rel. noise – Gaussian		0.0%	0.2%	0.5%	1.0%	1.5%	2.0%	2.5%
TV[η]	rel. ℓ_2-err. [%]	8.39±1.38	8.39±1.38	8.40±1.38	8.44±1.37	8.49±1.36	8.57±1.35	8.65±1.34
	PSNR	31.70±1.47	31.69±1.47	31.68±1.47	31.65±1.46	31.58±1.44	31.51±1.41	31.42±1.38
	SSIM	0.78±0.04	0.78±0.04	0.78±0.04	0.78±0.04	0.78±0.04	0.77±0.04	0.77±0.04
UNet	rel. ℓ_2-err. [%]	8.18±1.27	8.18±1.27	8.18±1.27	8.20±1.26	8.22±1.26	8.24±1.26	8.27±1.26
	PSNR	31.90±1.38	31.90±1.38	31.90±1.38	31.89±1.38	31.86±1.37	31.84±1.37	31.80±1.36
	SSIM	0.80±0.04	0.80±0.04	0.80±0.04	0.80±0.04	0.80±0.04	0.80±0.04	0.79±0.04
UNetFL	rel. ℓ_2-err. [%]	8.23±1.28	8.24±1.28	8.24±1.28	8.25±1.28	8.26±1.28	8.29±1.27	8.31±1.27
	PSNR	31.85±1.39	31.84±1.39	31.84±1.38	31.83±1.38	31.81±1.38	31.79±1.37	31.76±1.37
	SSIM	0.79±0.04	0.79±0.04	0.79±0.04	0.79±0.04	0.79±0.04	0.79±0.04	0.79±0.04
Tira	rel. ℓ_2-err. [%]	7.97±1.26	7.97±1.26	7.98±1.26	7.99±1.26	8.01±1.26	8.04±1.25	8.07±1.25
	PSNR	32.13±1.41	32.13±1.41	32.13±1.41	32.11±1.40	32.09±1.40	32.05±1.39	32.02±1.38
	SSIM	0.80±0.03	0.80±0.03	0.80±0.03	0.80±0.03	0.80±0.03	0.80±0.03	0.80±0.03
TiraFL	rel. ℓ_2-err. [%]	7.98±1.27	7.98±1.27	7.99±1.27	8.00±1.27	8.02±1.27	8.05±1.26	8.08±1.26
	PSNR	32.12±1.42	32.12±1.42	32.12±1.42	32.10±1.41	32.08±1.41	32.05±1.40	32.01±1.40
	SSIM	0.80±0.03	0.80±0.03	0.80±0.03	0.80±0.03	0.80±0.03	0.80±0.03	0.80±0.04
ItNet	rel. ℓ_2-err. [%]	**7.08±1.20**	**7.08±1.20**	**7.08±1.20**	**7.10±1.20**	**7.13±1.19**	**7.17±1.19**	**7.22±1.18**
	PSNR	**33.18±1.52**	**33.18±1.52**	**33.17±1.52**	**33.15±1.51**	**33.12±1.50**	**33.07±1.48**	**33.01±1.47**
	SSIM	**0.82±0.04**	**0.82±0.04**	**0.82±0.04**	**0.82±0.04**	**0.82±0.04**	**0.81±0.04**	**0.81±0.04**

Table D.9: **Reconstruction Tasks – Network Architectures.** Detailed description of hyper-parameters for all considered neural network architectures. The convolution kernel sizes are 3 or 3×3, the max-pooling sizes are 2 or 2×2, and the activation functions are rectified linear units (ReLUs) for all networks.

		Piecewise Constant	MNIST	Ellipses (Fourier)	Ellipses (Radon)	fastMRI (radial)	fastMRI (challenge)
UNet	inversion	Tikhonov (0.02)	Tikhonov (0.02)	$\mathbf{A}^*$	FBP (Hann-Filter)	$\mathbf{A}^*$	–
	U-Net levels	5	5	5	5	5	
	channels per level	(64, 128, 256, 512, 1024)	(64, 128, 256, 512, 1024)	(32, 64, 128, 256, 512)	(36, 72, 144, 288, 576)	(24, 48, 96, 192, 384)	
UNetFL	inversion	learned, init Tikh. (0.02)	learned, init Tikh. (0.02)	learned	–	learned	–
	U-Net levels	5	5	5		5	
	channels per level	(64, 128, 256, 512, 1024)	(64, 128, 256, 512, 1024)	(32, 64, 128, 256, 512)		(24, 48, 96, 192, 384)	
Tira	inversion	Tikhonov (0.02)	Tikhonov (0.02)	$\mathbf{A}^*$	–	$\mathbf{A}^*$	$\mathbf{A}^*$
	Tiramisu levels	5	5	5		5	5
	dense blocks per level	(5, 7, 9, 12, 15)	(5, 7, 9, 12, 15)	(5, 7, 9, 12, 15)		(5, 7, 9, 12, 15)	(6, 8, 10, 12, 14)
	initial channels	16	16	16		12	16
	channel growth rate	16	16	16		12	12
	bottleneck layers	25	25	20		18	20
TiraFL	inversion	learned, init Tikh. (0.02)	learned, init Tikh. (0.02)	learned	–	learned	–
	Tiramisu levels	5	5	5		5	
	dense blocks per level	(5, 7, 9, 12, 15)	(5, 7, 9, 12, 15)	(5, 7, 9, 12, 15)		(5, 7, 9, 12, 15)	
	initial channels	16	16	16		12	
	channel growth rate	16	16	16		12	
	bottleneck layers	25	25	20		18	
ItNet	inversion	Tikhonov (0.02)	Tikhonov (0.02)	$\mathbf{A}^*$	–	$\mathbf{A}^*$	–
	U-Net levels	5	5	5		5	
	channels per level	(64, 128, 256, 512, 1024)	(64, 128, 256, 512, 1024)	(32, 64, 128, 256, 512)		(24, 96, 192, 384)	
	iterations	8	8	8		8	
ConvNet	convolutional layers	–	4	–	–	–	–
	channels per layer		(32, 32, 64, 64)				
	fully connected layers		3				
	features per layer		(200, 200, 10)				
	remarks		dropout ($p = 0.5$) between fc layers				

Table D.10: **Reconstruction Tasks – Network Training.** Detailed description of hyper-parameters for all neural network trainings. All networks are trained in 1 or 2 phases and respective parameters are shown per training phase. Default parameters for the Adam optimizer are used except for the ε parameter, which is reported for all networks and training phases.

		Piecewise Constant	MNIST	Ellipses (Fourier)	Ellipses (Radon)	fastMRI (radial)	fastMRI (challenge)
UNet	epochs	(200, 75)	(200, 20)	(100, 10)	(155, 5)	(100)	–
	mini-batch size	(40, 40)	(40, 40)	(40, 40)	(40, 40)	(40)	
	learning rate	$(8 \cdot 10^{-5}, 5 \cdot 10^{-5})$	$(8 \cdot 10^{-5}, 5 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$	$(2 \cdot 10^{-4})$	
	weight decay	$(5 \cdot 10^{-3}, 5 \cdot 10^{-3})$	$(5 \cdot 10^{-3}, 5 \cdot 10^{-3})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(5 \cdot 10^{-4}, 1 \cdot 10^{-4})$	$(1 \cdot 10^{-5})$	
	Adam ε-parameter	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-4}, 1 \cdot 10^{-4})$	$(1 \cdot 10^{-4})$	
	gradient accumulation	(1, 200)	(1, 200)	(1, 200)	(1, 200)	(1)	
	jittering level	2	4	10	500	150	
UNetFL	epochs	(250, 50)	(200, 75)	(100, 10)	–	(45)	–
	mini-batch size	(40, 40)	(40, 40)	(40, 40)		(40)	
	learning rate	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$		$(8 \cdot 10^{-5})$	
	weight decay	$(5 \cdot 10^{-4}, 5 \cdot 10^{-4})$	$(5 \cdot 10^{-3}, 5 \cdot 10^{-3})$	$(1 \cdot 10^{-3}, 5 \cdot 10^{-4})$		$(1 \cdot 10^{-5})$	
	Adam ε-parameter	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$		$(1 \cdot 10^{-4})$	
	gradient accumulation	(1, 200)	(1, 200)	(1, 200)		(1)	
	jittering level	2	4	10		150	
	remarks					init **U** from UNet	
Tira	epochs	(200, 50)	(200, 50)	(50, 10)	–	(50, 16)	(40)
	mini-batch size	(40, 40)	(40, 40)	(8, 5)		(6, 6)	(4)
	learning rate	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$		$(8 \cdot 10^{-5}, 5 \cdot 10^{-5})$	$(1 \cdot 10^{-4})$
	weight decay	$(5 \cdot 10^{-4}, 5 \cdot 10^{-4})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$		$(1 \cdot 10^{-6}, 1 \cdot 10^{-6})$	$(1 \cdot 10^{-5})$
	Adam ε-parameter	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$		$(2 \cdot 10^{-4}, 2 \cdot 10^{-4})$	$(1 \cdot 10^{-4})$
	gradient accumulation	(1, 200)	(1, 100)	(1, 200)		(1, 1)	(1)
	jittering level	2	4	10		150	10
TiraFL	epochs	(200, 50)	(200, 50)	(30, 7)	–	(50, 20)	–
	mini-batch size	(40, 40)	(40, 40)	(10, 5)		(6, 6)	
	learning rate	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$		$(8 \cdot 10^{-5}, 5 \cdot 10^{-5})$	
	weight decay	$(5 \cdot 10^{-4}, 5 \cdot 10^{-4})$	$(5 \cdot 10^{-4}, 5 \cdot 10^{-4})$	$(1 \cdot 10^{-4}, 1 \cdot 10^{-5})$		$(1 \cdot 10^{-6}, 1 \cdot 10^{-6})$	
	Adam ε-parameter	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 1 \cdot 10^{-5})$		$(2 \cdot 10^{-4}, 2 \cdot 10^{-4})$	
	gradient accumulation	(1, 200)	(1, 200)	(1, 200)		(1, 1)	
	jittering level	2	4	10		150	
	remarks			init **T** from Tira			
ItNet	epochs	(100, 5)	(100, 10)	(35, 6)	–	(15, 8)	–
	mini-batch size	(40, 40)	(40, 40)	(15, 15)		(10, 10)	
	learning rate	$(5 \cdot 10^{-5}, 2 \cdot 10^{-5})$	$(8 \cdot 10^{-5}, 5 \cdot 10^{-5})$	$(5 \cdot 10^{-5}, 5 \cdot 10^{-5})$		$(5 \cdot 10^{-5}, 5 \cdot 10^{-5})$	
	weight decay	$(5 \cdot 10^{-4}, 5 \cdot 10^{-4})$	$(1 \cdot 10^{-3}, 1 \cdot 10^{-3})$	$(1 \cdot 10^{-4}, 1 \cdot 10^{-4})$		$(1 \cdot 10^{-6}, 1 \cdot 10^{-6})$	
	Adam ε-parameter	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$	$(2 \cdot 10^{-4}, 2 \cdot 10^{-4})$		$(1 \cdot 10^{-4}, 1 \cdot 10^{-4})$	
	gradient accumulation	(1, 200)	(1, 200)	(1, 200)		(1, 1)	
	jittering level	2	4	10		150	
	remarks			init **U** from UNet		init **U** from UNet	
ConvNet	epochs	–	(20, 10)	–	–	–	–
	mini-batch size		(40, 40)				
	learning rate		$(2 \cdot 10^{-4}, 5 \cdot 10^{-5})$				
	weight decay		$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$				
	Adam ε-parameter		$(1 \cdot 10^{-5}, 1 \cdot 10^{-5})$				
	gradient accumulation		(1, 1)				
	jittering level		no				

Table D.11: **Reconstruction Tasks – Adversarial Perturbations.** Detailed description of hyper-parameters for finding adversarial perturbations. The parameters reported for Net apply equally to all network types.

		Piecewise Constant	MNIST	Ellipses (Fourier)	Ellipses (Radon)	fastMRI (radial)	fastMRI (challenge)
TV[η]	ADMM iterations k_{rec}	50000	50000	5000	200	5000	5000
	ADMM iterations k_{grad}	2000	2000	200	20	200	150
	Adam iterations	30	30	250	15	250	50
	step size	$5 \cdot 10^0$	$5 \cdot 10^0$	$5 \cdot 10^0$	$5 \cdot 10^0$	$5 \cdot 10^0$	$5 \cdot 10^0$
	random initializations	200	100	6	6	6	6
Net	Adam iterations	300	100	1000	500	250	200
	step size	$5 \cdot 10^0$	$5 \cdot 10^0$	$5 \cdot 10^0$	$5 \cdot 10^0$	$5 \cdot 10^0$	$5 \cdot 10^0$
	random initializations	200	100	6	6	6	6
ConvNet $\circ$ TV[η]	ADMM iterations k_{rec}	–	50000	–	–	–	–
	ADMM iterations k_{grad}		2000				
	Adam iterations		100				
	step size		$5 \cdot 10^{-1}$				
	random initializations		100				
ConvNet $\circ$ Net	Adam iterations	–	100	–	–	–	–
	step size		$5 \cdot 10^{-1}$				
	random initializations		100				

E

Additions to Chapter 7

Table E.1: **AAPM Challenge – Network Architectures.** Detailed description of hyper-parameters for the neural network architectures. The convolution kernel sizes are 3×3, the max-pooling sizes are 2×2, and the activation functions are rectified linear units (ReLUs) for all networks.

		AAPM Sparse-View CT Challenge
UNet	inversion	estimated FBP (Hann-Filter)
	U-Net levels	5
	channels per level	(32, 64, 128, 256, 512)
Tira	inversion	estimated FBP (Hann-Filter)
	Tiramisu levels	5
	dense blocks per level	(4, 5, 7, 10, 12)
	initial channels	18
	channel growth rate	18
	bottleneck layers	16
ItNet	inversion	estimated FBP (Hann-Filter)
	U-Net levels	5
	channels per level	(32, 64, 128, 256, 512)
	iterations	4
ItNet-post	inversion	estimated FBP (Hann-Filter)
	U-Net levels	5
	channels per level	(32, 64, 128, 256, 512)
	iterations	5
LPD	inversion	estimated FBP (Hann-Filter)
	iterations	8
	channels per iteration (primal)	(32, 32, 32, 32, 32, 32, 32)
	channels per iteration (dual)	(32, 32, 32, 32, 32, 32, 32)

Table E.2: **AAPM Challenge – Network Training.** Detailed description of hyper-parameters for the neural network trainings. Networks are trained in 1, 2, or 3 phases. Respective parameters are shown per training phase. The Adam optimizer with default settings was used except for the ε-parameter, which is 10^{-5} for all networks and phases.

		AAPM Sparse-View CT Challenge
UNet	epochs	400
	mini-batch size	4
	learning rate	$2 \cdot 10^{-4}$
	weight decay	$1 \cdot 10^{-3}$
Tira	epochs	(150, 250)
	mini-batch size	(2, 2)
	learning rate	$(2 \cdot 10^{-4}, 8 \cdot 10^{-5})$
	weight decay	$(1 \cdot 10^{-3}, 1 \cdot 10^{-3})$
ItNet	epochs	(250, 250)
	mini-batch size	(2, 2)
	learning rate	$(8 \cdot 10^{-5}, 3 \cdot 10^{-5})$
	weight decay	$(1 \cdot 10^{-4}, 1 \cdot 10^{-4})$
	remarks	init $\mathbf{U}$ from UNet, weight sharing across iterations
ItNet-post	epochs	(300, 300, 300)
	mini-batch size	(2, 2, 2)
	learning rate	$(5 \cdot 10^{-5}, 2 \cdot 10^{-5}, 1 \cdot 10^{-5})$
	weight decay	$(1 \cdot 10^{-4}, 1 \cdot 10^{-4}, 1 \cdot 10^{-4})$
	remarks	init $\mathbf{U}$ from ItNet, no weight sharing across iterations, train only iterations 4 and 5
LPD	epochs	400
	mini-batch size	4
	learning rate	$2 \cdot 10^{-4}$
	weight decay	$1 \cdot 10^{-3}$