CONNECTING MATHEMATICAL MODELS FOR IMAGE PROCESSING AND NEURAL NETWORKS

A Dissertation Submitted Towards
the Degree Doctor of Natural Sciences (Dr. rer. nat.)
of the Faculty of Mathematics and Computer Science
of Saarland University

submitted by
TOBIAS ALT-VEIT

Saarbrücken, 2022

DAY OF COLLOQUIUM:
15.12.2022

DEAN OF FACULTY:
Prof. Dr. Jürgen Steimle

CHAIR OF THE COMMITTEE:
Prof. Dr. Eddy Ilg

REVIEWERS:
Prof. Dr. Joachim Weickert
Prof. Dr. Lars Ruthotto

ACADEMIC ASSISTANT:
Dr. Pascal Peter

to Anna

SHORT ABSTRACT

This thesis deals with the connections between mathematical models for image processing and deep learning. While data-driven deep learning models such as neural networks are flexible and well performing, they are often used as a black box. This makes it hard to provide theoretical model guarantees and scientific insights. On the other hand, more traditional, model-driven approaches such as diffusion, wavelet shrinkage, and variational models offer a rich set of mathematical foundations. Our goal is to transfer these foundations to neural networks. To this end, we pursue three strategies. First, we design trainable variants of traditional models and reduce their parameter set after training to obtain transparent and adaptive models. Moreover, we investigate the architectural design of numerical solvers for partial differential equations and translate them into building blocks of popular neural network architectures. This yields criteria for stable networks and inspires novel design concepts. Lastly, we present novel hybrid models for inpainting that rely on our theoretical findings. These strategies provide three ways for combining the best of the two worlds of model- and data-driven approaches. Our work contributes to the overarching goal of closing the gap between these worlds that still exists in performance and understanding.

KURZZUSAMMENFASSUNG

Gegenstand dieser Arbeit sind die Zusammenhänge zwischen mathematischen Modellen zur Bildverarbeitung und Deep Learning. Während datengetriebene Modelle des Deep Learning wie z.B. neuronale Netze flexibel sind und gute Ergebnisse liefern, werden sie oft als Black Box eingesetzt. Das macht es schwierig, theoretische Modellgarantien zu liefern und wissenschaftliche Erkenntnisse zu gewinnen. Im Gegensatz dazu bieten traditionellere, modellgetriebene Ansätze wie Diffusion, Wavelet Shrinkage und Variationsansätze eine Fülle von mathematischen Grundlagen. Unser Ziel ist es, diese auf neuronale Netze zu übertragen. Zu diesem Zweck verfolgen wir drei Strategien. Zunächst entwerfen wir trainierbare Varianten von traditionellen Modellen und reduzieren ihren Parametersatz, um transparente und adaptive Modelle zu erhalten. Außerdem untersuchen wir die Architekturen von numerischen Lösern für partielle Differentialgleichungen und übersetzen sie in Bausteine von populären neuronalen Netzwerken. Daraus ergeben sich Kriterien für stabile Netzwerke und neue Designkonzepte. Schließlich präsentieren wir neuartige hybride Modelle für Inpainting, die auf unseren theoretischen Erkenntnissen beruhen. Diese Strategien bieten drei Möglichkeiten, das Beste aus den beiden Welten der modell- und datengetriebenen Ansätzen zu vereinen. Diese Arbeit liefert einen Beitrag zum übergeordneten Ziel, die Lücke zwischen den zwei Welten zu schließen, die noch in Bezug auf Leistung und Modellverständnis besteht.

ABSTRACT

The goal of this thesis is to identify connections between models relying on partial differential equations (PDEs) and convolutional neural networks (CNNs) for image and signal processing.

The rise of deep learning and CNNs has drastically reshaped many areas of computer science. The availability of large amounts of data and powerful computing hardware has made modern neural networks highly flexible, performant, and thus ubiquitous. However, this comes at the cost of intransparent models which often do not provide any mathematical guarantees. On the other hand, traditional mathematical models for image processing such as diffusion, wavelet shrinkage, and variational models benefit from a long history of research on their theoretical properties and thus offer valuable concepts such as stability, well-posedness, and rotation invariance. Yet, they are outperformed by CNNs. Our goal is to identify fundamental links between the two worlds to come up with improved models that combine the performance of learning-based approaches with the transparency and mathematical foundations of model-based ones. To this end, we pursue three directions.

In a first step, we equip traditional models with few trainable components, making them more adaptive than their predecessors. After training the novel models, we reduce the resulting parameter set by re-modelling it through compact and physically plausible dynamics. This illustrates how mathematical models can be enhanced through concepts from machine learning without sacrificing theoretical guarantees.

In a second step, we extensively investigate the links between numerical algorithms for PDEs and neural network architectures by translating the essential numerical operations into neural building components. This allows us to identify design concepts for neural networks that inherit properties such as stability directly from their PDE-based counterparts. We find that many popular CNNs share an architectural similarity with well-known numerical algorithms for solving PDE-related problems. This second part provides a foundation for understanding the power of neural networks from the viewpoint of numerics for PDEs.

Finally, we show that our considerations are also relevant from a practical point of view by presenting two models for use in inpainting-based image compression. Both approaches take inspiration from our previous theoretical findings and practical experiences, allowing us to combine speed and adaptivity in hybrid models for image inpainting.

The three parts offer three ways for combining the best of both model- and data-driven worlds from different viewpoints: mathematical modelling, numerical analysis, and practical image processing applications. This allows us to come up with models that follow transparent and insightful modelling decisions, fulfil desirable mathematical properties such as stability and invariances, or use deep learning in a tightly controlled framework with interpretable results. These findings are intended as a contribution to the overarching goal of closing the gap between traditional mathematical models and deep learning that exists in both understanding and performance.

ZUSAMMENFASSUNG

Ziel dieser Arbeit ist es, Verbindungen zwischen Modellen für die Bild- und Signalverarbeitung, die auf partiellen Differentialgleichungen (PDEs, partial differential equations) beruhen, und faltenden neuronalen Netzen (CNNs, convolutional neural networks) zu identifizieren.

Der Aufstieg von Deep Learning und CNNs hat viele Bereiche der Informatik drastisch verändert. Die Verfügbarkeit von großen Datenmengen und leistungsfähiger Computerhardware haben dazu geführt, dass moderne neuronale Netze sehr flexibel, leistungsfähig, und damit allgegenwärtig sind. Dies hat jedoch den Nachteil, dass Modelle intransparent sind und oft keine mathematischen Garantien bieten. Demgegenüber profitieren traditionelle mathematische Modelle für die Bildverarbeitung wie Diffusion, Wavelet Shrinkage und Variationsansätze von einer langen Forschungsgeschichte zu ihren theoretischen Eigenschaften und bieten daher wertvolle Konzepte wie Stabilität, Gutgestelltheit und Rotationsinvarianz. Dennoch fehlt ihnen die Leistung, die neuronale Netze bieten können. Unser Ziel ist es, grundlegende Verbindungen zwischen den beiden Welten zu identifizieren, um verbesserte Modelle zu entwickeln, die die Leistung lernbasierter Ansätze mit der Transparenz und den mathematischen Grundlagen von modellbasierten Ansätzen kombinieren. Zu diesem Zweck verfolgen wir drei verschiedene Ansätze.

In einem ersten Schritt statten wir traditionelle Modelle mit einigen wenigen trainierbaren Komponenten aus und machen sie dadurch adaptiver als ihre Vorgänger. Nach dem Training der neuen Modelle reduzieren wir den resultierenden Parametersatz, indem wir sie durch kompakte und physikalisch plausible Dynamiken neu modellieren. Das zeigt wie mathematische Modelle durch Konzepte des Machine Learnings verbessert werden können, ohne dabei ihre theoretischen Garantien zu opfern.

In einem zweiten Schritt untersuchen wir ausführlich die Verbindungen zwischen numerischen Algorithmen für PDEs und neuronalen Netzwerkarchitekturen durch die Übersetzung von grundlegenden numerischen Operationen in neuronale Bausteine. Dies erlaubt uns, Designkonzepte für neuronale Netze zu identifizieren, die Eigenschaften wie z.B. Stabilität direkt von ihren PDE-basierten Gegenstücken übernehmen. Wir stellen fest, dass viele populäre CNNs eine architektonische Ähnlichkeit mit bekannten numerischen Algorithmen zur Lösung von PDE-Problemen aufweisen. Der zweite Teil bietet eine Grundlage für das Verständnis der Leistungsfähigkeit neuronaler Netze unter dem Gesichtspunkt der Numerik für PDEs.

Abschließend zeigen wir, dass unsere Überlegungen auch aus praktischer Sicht relevant sind, indem wir zwei Modelle vorstellen, welche nützlich für Inpainting-basierte Bildkompression sind. Beide Ansätze sind inspiriert von unseren vorherigen theoretischen Einsichten und praktischen Erfahrungen und erlauben uns, Geschwindigkeit und Adaptivität in hybriden Modellen für das Inpainting von Bildern zu kombinieren.

Die drei Teile bieten drei Möglichkeiten aus verschiedenen Blickwinkeln, das Beste aus beiden modell- und datengetriebenen Welten zu kombinieren: mathematische Modellierung, Numerik und praktische Anwendungen. Dies ermöglicht uns, Modelle zu entwickeln, die transparenten und erkenntnisreichen Modellierungsentscheidungen folgen, wünschenswerte mathematische Eigenschaften wie Stabilität und Invarianz erfüllen, oder Deep Learning in einem streng kontrollierten Rahmen mit interpretierbaren Ergebnissen nutzen. Diese Erkenntnisse sollen einen Beitrag zu dem übergreifenden Ziel liefern, die Lücke zwischen traditionellen mathematischen Modellen und Deep Learning zu schließen, die sowohl im Modellverständnis als auch der Leistung existiert.

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my advisor *Prof. Joachim Weickert*. Not only was he always available when asked for help, but his guidance and support were also indispensable for this research. I appreciated that he provided a clear vision for our projects, while still giving me the freedom on how to approach the goals.

Moreover, I thank *Prof. Lars Ruthotto* for fruitful discussions and serving as a second reviewer for this thesis.

This work would not have been possible without the financial support of the *European Research Council (ERC)* under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 741215, ERC Advanced Grant INCOVID). This is gratefully acknowledged.

I would also like to thank my collaborators: *Dr. Pascal Peter* for his expertise in inpainting-based image compression, valuable feedback on various manuscripts, and his encouragement; *Karl Schrader* for his extensive knowledge and practical experience on deep learning, which led to several productive collaborations; *Dr. Matthias Augustin* for checking intricate mathematical problems with thorough detail; and finally, *Sarah Andris* and *Rahul Mohideen Kaja Mohideen* for allowing me to broaden my knowledge by supporting them in their projects.

I thank *Sarah Andris*, *Johannes Bund*, *Karl Schrader*, *Dr. Pascal Peter*, and *Prof. Joachim Weickert* for proofreading this thesis and offering valuable feedback.

My thanks are extended to further current and former members of the mathematical image analysis (MIA) chair for creating a comfortable and inspiring working environment: *Dr. Leif Bergerhoff, Dr. Kireeti Bodduna, Pinak Bheed, Dr. Marcelo Cárdenas, Vassillen Chizhov, Michael Ertel, Peter Franke, Ferdinand-Dennis Jost, Niklas Kämper, Hyoseung Kang, Dr. Sabine Müller, Kristina Schaefer, Jón Arnar Tómasson, Aaron Wewior*, and *Ellen Wintringer*.

Last but not least, I would like to thank *Christiane and Kurt Alt* as well as *Regine and Georg Veit* for their past and present aid, and my wife *Anna Veit* for her unconditional love and support.

CONTENTS

1 Introduction 1
1.1 Contributions . . . 3
1.2 Organisation of the Thesis . . . 4
2 Mathematical Preliminaries 5
2.1 Derivative Operators . . . 5
2.2 Convolution . . . 6
2.3 Taylor Expansion . . . 7
2.4 Norms . . . 8
2.5 Signals and Images . . . 11
2.6 Error Measures . . . 12
3 Related Work 15
3.1 Diffusion . . . 16
3.2 Variational Methods . . . 36
3.3 Wavelet Shrinkage . . . 38
3.4 Deep Learning and Neural Networks . . . 41
3.5 Mathematical Foundations of Deep Learning . . . 62
3.6 Applications . . . 64

I Improving Mathematical Models through Learning
4 Trainable Adaptive Wavelet Shrinkage 73
4.1 Review: Shift-Invariant Wavelet Shrinkage . . . 75
4.2 Adaptive Wavelet Shrinkage . . . 79
4.3 Experiments . . . 82
4.4 Conclusions . . . 88
5 Trainable Integrodifferential Diffusion 91
5.1 Useful Reformulation of EED . . . 93
5.2 Integrodifferential Diffusion . . . 94
5.3 Finding Scale-adaptive Parameter Functions . . . 96
5.4 Extension to Inpainting . . . 104
5.5 Conclusions . . . 108

II Mathematically Founded Neural Networks
6 Mathematical Models and Residual Networks 113
6.1 Review: Basic Approaches . . . 115
6.2 Translation into Residual Networks . . . 116
6.3 Dictionary of Activation Functions . . . 122
6.4 Conclusions . . . 123
7 Numerical Algorithms and Neural Architectures 129
7.1 Review: Generalised One-dimensional Diffusion . . . 131
7.2 From Diffusion to Symmetric Residual Networks . . . 132
7.3 The Value of Skip Connections . . . 137
7.4 Review: Multigrid Solvers and U-nets . . . 142

7.5 From Multigrid to U-nets . . . 144
7.6 Experimental Evaluations . . . 147
7.7 Conclusions . . . 158
8 Rotationally Invariant Neural Networks 161
8.1 Two Views on Rotational Invariance . . . 163
8.2 Towards Rotationally Invariant Networks . . . 164
8.3 Discussion . . . 175
8.4 Experiments . . . 176
8.5 Conclusions . . . 181

III Image Inpainting with Hybrid Models

9 Inpainting with Anisotropic Shepard Interpolation 185
9.1 Review: Isotropic Shepard Interpolation . . . 187
9.2 Anisotropic Shepard Interpolation . . . 188
9.3 Inpainting Experiments . . . 193
9.4 Application to Compression . . . 198
9.5 Compression Experiments . . . 200
9.6 Conclusions . . . 204
10 Learning Sparse Masks for Diffusion Inpainting 207
10.1 Review: Data Optimisation for Inpainting . . . 209
10.2 Sparse Masks with Surrogate Inpainting . . . 210
10.3 Experiments . . . 213
10.4 Conclusions . . . 221
11 Conclusions and Outlook 223
11.1 Conclusions . . . 223
11.2 Outlook . . . 225
A Rotationally Invariant Wavelet Shrinkage 227
B Stability of Du Fort–Frankel Schemes 231
C Stability of Multiscale Architectures 237
D Contributions and Publications 239
D.1 Further Contributions . . . 239
D.2 List of Publications . . . 240
E Bibliography 243
F Glossary 283
G List of Symbols 285
H List of Figures 291
I List of Tables 293

1 INTRODUCTION

The rapid progress in artificial intelligence over the last decade has drastically reshaped many areas of computer science. The advances in *neural networks*, combined with the availability of performant parallel computing hardware have transformed *deep learning* from a niche research field to a universally applicable methodology for a large class of tasks [152, 223, 225, 330].

Neural networks are highly flexible tools: Given large amounts of data consisting of inputs and desired outputs, so-called ground truth, supervised networks learn to solve a task by comparing their predictions against the ground truth and adapting their parameters such the two are close. This allows to adapt models to various tasks without the need for researchers to craft models by hand.

While this brings enormous flexibility, this approach is not without any downsides. One can identify two common flaws which serve as a motivation of this thesis. First, most neural networks suffer from an unintuitive behaviour in the sense that they do not mimic the mammalian brain structure, unlike their original biological motivation [132, 192, 253] would suggest. It is possible that an image recognition system based on a neural network identifies e.g. cat images correctly in the majority of cases. However, some images which a human would undeniably recognise, are unexpectedly classified e.g. as a piano. While the cause of these *adversarial examples* [154] is known, designing models which are robust against this phenomenon is non-trivial and subject to ongoing research [167, 229, 304, 404].

Why should one care whether a cat becomes a piano? In a controlled research environment, these cases are often within the small margin of overall classification error and do not seem to make a difference in the grand scheme of things. However, when thinking of practical applications of image recognition systems such as in automated driving environments, it is of utmost importance that these systems work correctly in all cases, or at least provide a measure of uncertainty to be able to refer the decision to a human operator. Moreover, these weak points can even be abused with malicious intent by purposefully designing misleading fringe cases [120]. This shows that enforcing mathematical guarantees in such models may seem restrictive in the first place, but must not be disregarded in safety-critical applications.

Whereas this unintuitive behaviour is of practical concern, the second flaw is of more philosophical nature. As neural networks usually contain huge amounts of parameters in the order of several millions, they often work as a black box. A loss function incentivises

that the network outputs come close to the desired ground truth, and the network finds the way towards a good parameter set automatically. How individual parameters interact with each other, and why it is exactly this combination that works well, is often infeasible to investigate. This prohibits researchers from gaining insights into these models, which in turn makes scientific progress difficult.

In fact, the benefits of overparameterisation [40] in neural networks suggest that they even seem to violate the principle of *Occam's razor*, which states that simple models are usually to be preferred over complicated ones. We argue that even though it may be convenient to solve a problem with the help of deep learning, it should be done in such a way that the model is suitable to the problem and can be understood well.

The aforementioned problems have prompted researchers to provide mathematical foundations for deep learning. Since around 2015, many works have used various mathematical frameworks to analyse neural networks in terms of their stability [167], expressibility [303], generalisation potential [408], robustness against adversarial examples [140], and their connections to *partial* and *ordinary differential equations (PDEs and ODEs)* [73, 75, 238, 319].

We contribute to the mathematical foundations of deep learning by considering classical, well-connected image processing models. Approaches that rely on concepts such as *diffusion* [195, 279, 369], *wavelets* [108, 165, 248], and *variational methods* [27, 354, 391] have a long history of research and are mathematically well-founded. While many of these models cannot match the performance of deep neural networks, they provide advantages in terms of strict mathematical guarantees, transparent modelling, and interpretable outcomes.

This dichotomy is the fundamental motivation of this thesis: We want to connect neural networks and mathematical concepts in such a way that we preserve the best of both worlds within performant, but mathematically well-founded models.

One can approach this task in two ways. The goal of an *analytic* strategy is to assign a compact mathematical representation to a neural network. However, this option is challenging due to many design possibilities within neural networks and the large amount of trainable parameters involved. Thus, we pursue a *synthetic* strategy that takes well-founded mathematical concepts as a baseline and uses them to create models that inherit the theoretical guarantees, while benefiting from the strengths of learning. Even though this strategy does not cover the full range of neural networks, it can provide fundamental design criteria that allow for well-performing but mathematically founded components.

1.1 CONTRIBUTIONS

To this end, we identify three visions that this research avenue entails [371].

VISION 1: IMPROVING MATHEMATICAL MODELS BY LEARNING We start by equipping diffusion- and wavelet-based models with fundamental ideas of learning. This preserves the tight mathematical framework of the underlying approaches, but still introduces additional flexibility by allowing parameter adaptivity. However, in contrast to similar works, we reduce the learned parameter set by identifying the relations of the learned parameters to a feasible, minimal set of parameters. For the test application of denoising, this yields significant gains over comparable methods and shows how tightly controlled learning aspects can improve traditional mathematical models.

VISION 2: MATHEMATICALLY FOUNDED NEURAL NETWORKS In a second, more theoretical step, we analyse numerical schemes for models such as nonlinear diffusion, wavelet shrinkage, and variational energies, and connect their essential building blocks to popular neural network architectures. We start with a setting where all involved parameters are fixed, allowing us to focus on the connections between the central nonlinear design choices of diffusivities, shrinkage functions, variational penalisers, and neural activation functions. We extend these considerations to models with trainable filters and identify crucial design criteria for guaranteeing mathematical concepts such as well-posedness, Euclidean stability, and rotation invariance of specific networks. Moreover, we connect several numerical solution strategies for PDEs to popular neural architectures and show that the resulting networks can perform well while saving parameters.

VISION 3: DEVELOPING HYBRID MODELS The previous two parts form the foundation of the final vision where we use the established insights to create hybrid models. We present two models that aim at improving inpainting-based image compression by combining successful concepts to find a balance between adaptivity and efficiency. In once instance thereof, we actually use neural networks as black boxes, however, we design our loss function different from typical data-driven ones by prescribing the residual of a PDE as a loss for the network. Thus, the network finds the way to a solution of the PDE and acts as a surrogate solver which is fully decoupled from the model itself.

These visions are intended as three different options for bridging the gap between model- and data-driven approaches.

1.2 ORGANISATION OF THE THESIS

Before we present our contributions, we review mathematical preliminaries in Chapter 2, and provide an overview of related work in Chapter 3.

The rest of the thesis is structured into three parts that coincide with the aforementioned visions. Part I consists of Chapters 4 and 5, wherein we come up with trainable models for image denoising which rely on wavelet shrinkage and anisotropic diffusion, respectively. Both chapters follow a similar methodology: The classical models are equipped with trainable parameters which are optimised for the task at hand. Afterwards, we manually inspect the parameter evolutions and substitute them with analytical models that reduce the parameter set to a minimum.

Part II presents the theoretical connections between neural networks and numerical solution strategies for diffusion, wavelet shrinkage, and variational models. In Chapter 6 we consider the four aforementioned approaches in the one-dimensional setting and provide an overview of their connections, with a focus on nonlinear design functions. Chapter 7 extends these considerations by focusing on generalised diffusion and neural networks. Lastly, Chapter 8 investigates the two-dimensional case and the associated problem of guaranteeing rotation invariance for the proposed neural architectures.

Finally, Part III of the thesis containing Chapters 9 and 10 is devoted to image inpainting with hybrid models. Chapter 9 constitutes an exception as it does not involve any learning, still it is concerned with a hybrid model combining efficiency and performance for image inpainting and compression. In Chapter 10 we harness the full power of deep learning in a model which learns masks for image inpainting and solves the inpainting problem simultaneously.

In Chapter 11 we present our conclusions as well as an outlook on open questions and future research directions. The Appendices A, B, and C present various proofs for rotation invariance and stability guarantees referring to Chapters 4, 7, and 8, respectively. At the end of the thesis, we list further information such as a list of own publications, the bibliography, a glossary, as well as lists of figures, tables, and symbols.

2

MATHEMATICAL PRELIMINARIES

In this chapter, we introduce basic mathematical definitions that appear throughout the thesis.

2.1 DERIVATIVE OPERATORS

Since this thesis is concerned with partial differential equations (PDEs), the first concept we define are *partial derivatives*. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. We define the partial derivative w.r.t. the i-th argument at a position $\boldsymbol{x} \in \mathbb{R}^n$ as

$$\frac{\partial f}{\partial_{x_i}}(\boldsymbol{x}) = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_i, \ldots, x_n)}{h}. \quad (2.1)$$

If this limit exists, the function is *partially differentiable*. A general assumption which we make in this thesis is that all functions are sufficiently often differentiable for our purposes. We additionally use the expressions $\partial_{x_i} f$ and f_{x_i} as shorthand notations for the partial derivative.

GRADIENT AND DIVERGENCE Two fundamental derivative operators that use partial derivatives are the *gradient* and the *divergence*. For a scalar function $f : \mathbb{R}^n \to \mathbb{R}$, the gradient is defined as

$$\boldsymbol{\nabla} f(\boldsymbol{x}) = \left(\partial_{x_1} f(\boldsymbol{x}), \ldots, \partial_{x_n} f(\boldsymbol{x})\right)^\top. \quad (2.2)$$

The gradient is a vector containing all partial derivatives of the function f and points in the direction of its steepest ascent.

Its counterpart is the divergence operator, which for a vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ with $\boldsymbol{f}(\boldsymbol{x}) = \left(f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\right)^\top$ is given by

$$\operatorname{div}(\boldsymbol{f}(\boldsymbol{x})) = \boldsymbol{\nabla}^\top(\boldsymbol{f}(\boldsymbol{x})) = \sum_{i=1}^{n} \partial_{x_i} f_i(\boldsymbol{x}). \quad (2.3)$$

The divergence sums up partial derivatives of the components of $\boldsymbol{f}$ w.r.t. the respective dimension of the component. We predominantly use the alternative notation $\boldsymbol{\nabla}^\top$ for the divergence operator.

LAPLACE OPERATOR The last derivative operator we need to define is the *Laplace operator*, also called the *Laplacian*. The Laplacian of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\Delta f = \operatorname{div}(\boldsymbol{\nabla} f(\boldsymbol{x})) = \sum_{i=1}^{n} \partial_{x_i x_i} f(\boldsymbol{x}). \quad (2.4)$$

The Laplacian sums up the second derivatives of f in all respective directions. Intuitively, it denotes whether a point is a source or a sink when interpreting the function as a spatial distribution of concentrations.

Sometimes, one also refers to the Laplace operator as the *harmonic operator*. Consequently, its second power Δ^2 is called the *biharmonic operator*.

Partial derivatives, gradient, divergence, and Laplacian play an essential role within this thesis to compute derivatives of signals and images. Derivative information can encode useful image properties such as edges and corners.

Note that we have defined all these derivative operators as spatial operators. Often times, we deal with functions with both spatial components and a temporal one, e.g. $f(\boldsymbol{x}, t)$. In these cases, the derivative operators always refer only to the spatial arguments, i.e. $\boldsymbol{x}$ in this case.

DIRECTIONAL DERIVATIVES To describe boundary conditions of PDEs we make use of *directional derivatives*. For a function $f : \mathbb{R}^n \to \mathbb{R}$, we define the directional derivative in the direction $\boldsymbol{v} \in \mathbb{R}^n$ by

$$\partial_{\boldsymbol{v}} f = \lim_{h \to 0} \frac{f(\boldsymbol{x} + h\boldsymbol{v}) - f(\boldsymbol{x})}{h} \,. \tag{2.5}$$

Thus, the directional derivative measures the change of the function f at the position $\boldsymbol{x}$ in the direction of $\boldsymbol{v}$.

GÂTEAUX DERIVATIVES Minimisers of variational energies are found by means of *Gâteaux derivatives* which generalise derivatives to the calculus of variations [27, 137]. For a functional $F(u)$ which maps a function $u : \mathbb{R}^n \to \mathbb{R}$ to a real value, we define the Gâteaux derivative as

$$\partial_u F(u; r) = \lim_{\varepsilon \to 0} \frac{F(u + \varepsilon r) - F(u)}{\varepsilon} \,. \tag{2.6}$$

Here, $r : \mathbb{R}^n \to \mathbb{R}$ is an arbitrary perturbation function. If this limit exists for all r, the functional F is Gâteaux differentiable. In a similar way as for function derivatives, we abbreviate Gâteaux derivatives as F_u.

2.2 CONVOLUTION

One of the most important operations we encounter in this thesis is the *convolution*. It is essential for continuous image operations as well as for image processing with discrete filters. It is also the fundamental mathematical operation within CNNs.

For two functions $f, g : \mathbb{R}^n \to \mathbb{R}$, the convolution of the two is given by

$$(f * g)(\boldsymbol{x}) = \int_{\mathbb{R}^n} f(\boldsymbol{x} - \boldsymbol{y})\, g(\boldsymbol{y})\, d\boldsymbol{y} \,. \tag{2.7}$$

Intuitively, one of the functions — the so-called *convolution kernel* — is mirrored and continuously moved over the other function. At each point, the value of the convolution is determined by the area under the product of the two functions.

The discrete convolution of two signals $\boldsymbol{f} = (f_i)_{i\in\mathbb{Z}}, \boldsymbol{g} = (g_i)_{i\in\mathbb{Z}}$ is consequently defined as

$$(\boldsymbol{f} * \boldsymbol{g})_i = \sum_{k\in\mathbb{Z}} f_{i-k}\, g_k\,. \tag{2.8}$$

This one-dimensional definition is sufficient for our purposes since we can reformulate digital images in terms of vectors, as we will see in Section 2.5.

The convolution possesses several useful properties. In particular, it is linear and shift-invariant: Shifting the convolution kernel does not change the result of the convolution. This is the core idea in the design of CNNs.

2.3 TAYLOR EXPANSION

The *Taylor expansion* can be used to approximate a sufficiently smooth function at a position by means of a power series. It is an important tool for deriving finite difference approximations and analysing their quality.

For a one-dimensional function $f : \mathbb{R} \to \mathbb{R}$ which is $n+1$ times continuously differentiable with bounded derivatives, the Taylor series in a position $x + h$ is defined as

$$f(x+h) = \sum_{k=0}^{n} \frac{h^k}{k!} f^{(k)}(x) + \mathcal{O}\left(h^{n+1}\right). \tag{2.9}$$

Here, $f^{(k)}(x)$ is the k-th derivative of f at x, and $\mathcal{O}$ is the *Landau notation*. Intuitively, it expresses the fact that there is a remainder which does not grow faster than the function h^{n+1}.

More rigorously, for two real-valued functions $f, g : \mathbb{R} \to \mathbb{R}$ we have $f(x) \in \mathcal{O}(g(x))$ if there exist constants $c > 0$ and x_0 such that

$$|f(x)| \leq C\,|g(x)| \quad \forall x \geq x_0\,. \tag{2.10}$$

Thus, the Taylor expansion (2.9) describes the function f in a vicinity around x by means of its derivatives in x. The larger the distance h becomes, the less accurate the approximation becomes.

For a two-dimensional function $f : \mathbb{R}^2 \to \mathbb{R}$, we analogously define the Taylor expansion in $\boldsymbol{x} + \boldsymbol{h}$ by

$$f(\boldsymbol{x}+\boldsymbol{h}) = \sum_{k=0}^{n} \frac{1}{k!} \langle \boldsymbol{h}, \boldsymbol{\nabla}\rangle^k f(\boldsymbol{x}) + \mathcal{O}\left(|\boldsymbol{h}|^{n+1}\right). \tag{2.11}$$

Instead of one-dimensional derivatives, one uses the scalar product of $\boldsymbol{h} = \left(h_x, h_y\right)^\top$ and the gradient operator $\boldsymbol{\nabla}$, and powers thereof.

2.4 NORMS

Norms are an important mathematical concept within this thesis. They play a role in loss function design as well as several stability analyses. In the following, we define both *vector norms* and *matrix norms*.

VECTOR NORMS A vector norm $\|\cdot\| : V \to \mathbb{R}_0^+$ maps elements from a vector space V to a nonnegative number which can be interpreted as the size of that element. For ease of definition, we only consider vector spaces over the real numbers.

A norm has to fulfil three axioms:

- **Definiteness:** If the norm of an element $\boldsymbol{v}$ is zero, then the element itself is the zero element:

$$\|\boldsymbol{v}\| = \mathbf{0} \Leftrightarrow \boldsymbol{v} = \mathbf{0} \qquad \forall \boldsymbol{v} \in V. \tag{2.12}$$

- **Absolute homogeneity:** Multiplying an element $\boldsymbol{v}$ with a scalar α within the norm is equivalent to multiplying the norm of the element with the absolute value of the scalar:

$$\|\alpha \boldsymbol{v}\| = |\alpha| \, \|\boldsymbol{v}\| \qquad \forall \boldsymbol{v} \in V, \alpha \in \mathbb{R}\,. \tag{2.13}$$

- **Subadditivity:** The norm of the sum of two elements $\boldsymbol{v}, \boldsymbol{w}$ is smaller than the sum of their individual norms:

$$\|\boldsymbol{v} + \boldsymbol{w}\| \leq \|\boldsymbol{v}\| + \|\boldsymbol{w}\| \qquad \forall \boldsymbol{v}, \boldsymbol{w} \in V. \tag{2.14}$$

  This property is also referred to as the *triangle inequality*, as it implies that the sum of the lengths of two sides of a triangle is always larger than the remaining one.

The axioms of absolute homogeneity and subadditivity imply that a norm is also nonnegative, i.e. $\|\boldsymbol{v}\| \geq 0 \; \forall \boldsymbol{v} \in V$.

In particular, we are interested in *p-norms* in Euclidean vector spaces. For a vector $\boldsymbol{v} \in \mathbb{R}^n$, the p-norm for a real-valued p with $1 \leq p < \infty$ is defined as

$$\|\boldsymbol{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}}. \tag{2.15}$$

The most important norm for us is the *Euclidean* or L^2-norm which arises for $p = 2$:

$$\|\boldsymbol{v}\|_2 = \sqrt{\sum_{i=1}^{n} |v_i|^2}. \tag{2.16}$$

Moreover, we encounter some models that make use of the L^1-norm which is obtained for $p = 1$:

$$\|\boldsymbol{v}\|_1 = \sum_{i=1}^{n} |v_i|\,. \tag{2.17}$$

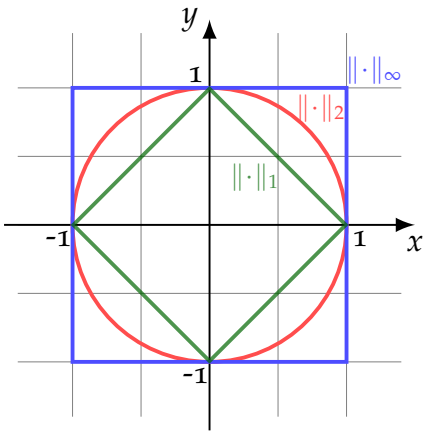


Figure 2.1: Unit balls for the L^1-, L^2-, and maximum-norms. Coloured lines denote those two-dimensional vectors where the respective norm has a value of one. The maximum norm is coloured in blue, the L^2-norm in red, and the L^1-norm in green.

Both norms play an important role in the design of loss functions. While the L^2-norm is sensitive to outliers, the L^1-norm is robust and promotes sparsity. This, however, comes at the price of nondifferentiability of the L^1-norm in the zero position.

A last norm which is important for our stability analysis is the *maximum* or *infinity* norm which is the limit of the p-norm for $p \to \infty$:

$$\|\boldsymbol{v}\|_\infty = \max_i \boldsymbol{v}_i \,. \tag{2.18}$$

In Part II, we analyse PDE and CNN models in terms of their effect on the norm of the signal as it evolves through the steps of the model. We show that the norm of the signal is always nonincreasing. To this end it is important to note that the maximum norm is stricter than the Euclidean norm, which again is stricter than the L^1-norm, since

$$\|\boldsymbol{v}\|_1 \leq \|\boldsymbol{v}\|_2 \leq \|\boldsymbol{v}\|_\infty \qquad \forall \boldsymbol{v} \in V. \tag{2.19}$$

This behaviour can be visualised with the help of unit balls. Figure 2.1 depicts vectors in a two-dimensional plane for which the L^1-, L^2-, and maximum norms attain a value of one. The level lines of the maximum norm are squares containing the circular level lines of the Euclidean norm. These in turn contain the diamond shaped level lines of the L^1-norm.

MATRIX NORMS Matrix norms extend the notion of norms from vectors to matrices. They map matrices to nonnegative real values and follow the same axioms as vector norms.

Of special interest is the *spectral norm*, which is the matrix norm induced by the Euclidean vector norm. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, the spectral norm is defined as

$$\|\boldsymbol{A}\|_2 = \max_{\|\boldsymbol{x}\|_2 = 1} \|\boldsymbol{A}\boldsymbol{x}\|_2 \,, \tag{2.20}$$

i.e. it denotes the strongest elongation that an application of $\boldsymbol{A}$ to any normalised vector $\boldsymbol{x} \in \mathbb{R}^n$ with $\|\boldsymbol{x}\|_2 = 1$ can have.

An equivalent definition in terms of the eigenvalues of $\boldsymbol{A}$ is given by

$$\|\boldsymbol{A}\|_2 = \max\left\{ \sqrt{|\lambda|} \,\middle|\, \lambda \text{ is an eigenvalue of } \boldsymbol{A}^\top \boldsymbol{A} \right\}. \tag{2.21}$$

This definition is useful for showing stability of numerical algorithms for diffusion and their neural network counterparts in Chapter 7.

Certain diffusion models also consider the *Frobenius norm*, which is defined as the root of the sum of squared matrix entries a_{ij} of $\boldsymbol{A}$:

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}. \tag{2.22}$$

It arises from the *trace operator*. The trace of a square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the sum of its diagonal elements:

$$\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^{n} a_{ii}. \tag{2.23}$$

Thus, one obtains

$$\|\boldsymbol{A}\|_F = \operatorname{tr}\left(\boldsymbol{A}^\top \boldsymbol{A}\right). \tag{2.24}$$

We require the notion of matrix norms for our stability analysis of diffusion operators in Chapter 7 and discussions on rotational invariance in Chapter 8.

SPECTRAL RADIUS AND GERSHGORIN'S CIRCLE THEOREM For stability analysis, another important concept which is closely related to the spectral norm is the *spectral radius*. The spectral radius of a square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is given by the largest absolute eigenvalue, i.e.

$$\rho(\boldsymbol{A}) = \max\{|\lambda| \mid \lambda \text{ is an Eigenvalue of } \boldsymbol{A}\}. \tag{2.25}$$

Throughout this thesis, we only consider real matrices. In that case, we have equality between the spectral norm and the spectral radius if $\boldsymbol{A}$ is symmetric. In these cases, we can use the two concepts interchangeably.

Gershgorin's circle theorem [143] helps us to easily estimate the spectral radius of many matrices which we encounter. It states that the eigenvalues of a square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ lie in a union of circles. Each circle is placed in the complex plane with the diagonal entry a_{ii} as the centre, and a radius

$$r_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \left|a_{ij}\right|, \tag{2.26}$$

which corresponds to the sum of the absolute off-diagonal values.

This theorem allows us to elegantly estimate the spectral radius of matrices in our stability analysis in Chapter 7.

STABILITY AND WELL-POSEDNESS Several of our main results are concerned with stability and well-posedness guarantees of neural networks which are inherited from diffusion filters.

We define a sequence of vectors $\boldsymbol{u}^0, \boldsymbol{u}^1, \ldots, \boldsymbol{u}^K$ to be *stable in a p-norm*, if

$$\|\boldsymbol{u}^{k+1}\|_p \leq \|\boldsymbol{u}^k\|_p \qquad (k = 0, \ldots, K-1) \; . \tag{2.27}$$

In particular, we are interested in maximum-minimum stability and Euclidean stability, which represent the choices of $p \to \infty$ and $p = 2$, respectively.

Moreover, a problem is *well-posed* in the sense of Hadamard [169] if three conditions are fulfilled: A solution to the problem exists, this solution is unique, and it depends continuously on the input data to the problem. A problem which is not well-posed is ill-posed.

For example, removing noise from an image is an ill-posed problem: Without additional assumptions, the original value of corrupted image pixels cannot be recovered. Diffusion filters inherently rely on smoothness assumptions with a regularizing effect [326], transforming an ill-posed denoising problem into well-posed one. This is one of the central motivation for using denoising applications as a test bed in this thesis.

2.5 SIGNALS AND IMAGES

We deal with both one-dimensional signals and two-dimensional images. While modelling often takes place in the continuous setting, a practical implementation requires us to define digital counterparts of the continuous data.

CONTINUOUS DEFINITIONS We define a continuous signal as a mapping $f : [a, b] \to \mathbb{R}$ from a one-dimensional domain $[a, b] \subset \mathbb{R}$ with $a < b$ to the real values. In the same manner we define grey value images $f : \Omega \to \mathbb{R}$ as a mapping from a rectangular image domain $\Omega = [a, b] \times [c, d] \subset \mathbb{R}^2$ with $a < b, c < d$.

In some cases, we deal with colour images and signals with multiple channels. In that case, the co-domain is multi-dimensional. For example, an image in the red-green-blue (RGB) colour space is defined as $\boldsymbol{f} : \Omega \to \mathbb{R}^3$ where each component of the co-domain refers to the respective colour.

DISCRETE DEFINITIONS In practice, we discretise the continuous signals and images by means of uniform sampling. Given a grid size h, a discrete signal $\boldsymbol{f} \in \mathbb{R}^n$ with n positions is obtained by sampling the continuous function f at equidistant positions

$$f_i = f\left(a + ih - \frac{h}{2}\right) . \tag{2.28}$$

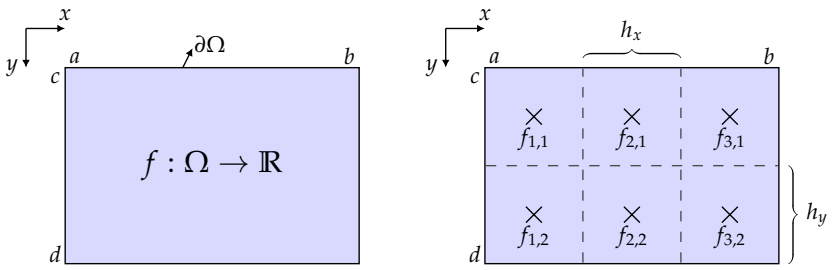


Figure 2.2: Examples for a continuous (left) and discrete (right) rectangular image. The sampling has been performed such that one obtains a discrete image with 2×3 pixels.

Similarly, discrete images arise by sampling the two-dimensional image domain Ω with grid sizes h_x, h_y in the respective directions to obtain a discrete image $\boldsymbol{f} \in \mathbb{R}^{n_x n_y}$ as

$$f_{i,j} = f\left(a + ih_x - \frac{h_x}{2}, c + jh_y - \frac{h_y}{2}\right), \tag{2.29}$$

where the pixel numbers in x- and y-direction are defined by

$$n_x = \frac{b-a}{h_x}, \tag{2.30}$$

$$n_y = \frac{d-c}{h_y}. \tag{2.31}$$

Figure 2.2 displays a continuous rectangular image and its discretised counterpart. The coloured area denotes the image domain, with the x-axis pointing towards the right and the y-axis pointing downward. The discrete image is sampled into six pixels with equal grid size in both directions.

VECTORISATION In the above definition, the naive representation of the image would be a matrix in $\mathbb{R}^{n_x \times n_y}$. Yet, we represent it as a vector in $\mathbb{R}^{n_x n_y}$. This is due to a simple reordering where we write the pixels of the image row-wise into a column vector. For example, the 2×3 image of Figure 2.2 results in a vector $\boldsymbol{f} \in \mathbb{R}^6$ with six components $\boldsymbol{f} = (f_{1,1}, f_{2,1}, f_{3,1}, f_{1,2}, f_{2,2}, f_{3,2})^\top$.

This has the advantage that we can express many operations on the image in terms of matrix-vector multiplications instead of having to deal with more cumbersome tensor operations; see e.g. Section 3.1.2. Multi-channel images are vectorised channel-wise. For example, a discrete RGB image is denoted by $\boldsymbol{f} = (\boldsymbol{f}_R\ \boldsymbol{f}_G\ \boldsymbol{f}_B)^\top$ where each channel is represented by a vector of pixels.

2.6 ERROR MEASURES

Measuring the quality of processed images is an important task to be able to rank different methods in an experimental setting. Throughout

this thesis, we assume that ground truth data is always available. In this setting we can use so-called *full-reference metrics*. The two error measures which we consider are the *mean square error (MSE)* and the *peak signal-to-noise ratio (PSNR)*.

MEAN SQUARE ERROR The mean square error between two images $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^{n_x n_y}$ is defined as

$$\mathrm{MSE}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{n_x n_y} \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 = \frac{1}{n_x n_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left(u_{i,j} - v_{i,j}\right)^2. \tag{2.32}$$

Thus, the MSE averages the squared deviation between each pixel of $\boldsymbol{u}$ and its counterpart in $\boldsymbol{v}$. As a consequence, it is also prone to outliers: Very few extreme deviations increase the MSE more than numerous small deviations. Low MSE values correlate with better quality, and an MSE of zero means that the two images are the same.

PEAK SIGNAL-TO-NOISE RATIO The peak signal-to-noise ratio is closely related to the MSE. Let $c_{\max}$ be the maximal possible grey value of two images $\boldsymbol{u}, \boldsymbol{v}$ with the same range. Then the PSNR is defined as

$$\mathrm{PSNR}(\boldsymbol{u}, \boldsymbol{v}) = 10 \log_{10}\left(\frac{c_{\max}^2}{\mathrm{MSE}(\boldsymbol{u}, \boldsymbol{v})}\right). \tag{2.33}$$

The PSNR is measured in decibels (dB), and higher values indicate better quality. Moreover, its behaviour is logarithmic: A quality increase from 10 dB to 20 dB is less drastic than an increase from 20 dB to 30 dB. For two equal images it is undefined.

The advantage of the PSNR over the MSE is that it is independent of the grey value range, whereas an MSE always needs to be interpreted together with the range at hand.

In some cases, MSE and PSNR can act counterintuitively to the quality perceived by the human eye [366]. This has triggered researchers to come up with so-called *perceptual metrics*, either with purely mathematical [366] or neural [413] design; see e.g. [407] for an overview and evaluation of several metrics. A large number of metrics has been proposed, each with their own advantages and flaws, and so far no metric is universally agreed on. In this thesis, we exclusively use MSE and PSNR as quality metrics as they are easy to compute and widely accepted.

This concludes our review of mathematical preliminaries. In the following chapter, we introduce the fundamental concepts that this thesis is concerned with, and present related work.

3

RELATED WORK

In this chapter we discuss prior work related to this thesis. First, we review four basic approaches that frequently appear throughout the thesis: diffusion in Section 3.1, variational methods in Section 3.2, wavelet shrinkage in Section 3.3, and neural networks in Section 3.4.

Diffusion is a fundamental tool for image processing [195, 279, 369]. We will see in Part II that its numerical realisation shares core connections to popular neural network architectures. Moreover, it allows us to come up with new models for denoising which are built on integrodifferential diffusion in Chapter 4. Lastly, it helps us to advance inpainting-based image compression by combining successful concepts from diffusion with efficient interpolation strategies. This is discussed in Chapter 9.

Variational methods [27, 354, 391] and wavelet shrinkage [108, 165, 248] are well-connected to diffusion models and help us to paint a more complete picture of neural network design in Chapters 6 and 8. Moreover, our first successes for model reduction strategies are built on wavelet shrinkage, as presented in Chapter 4.

Deep learning and neural networks [152, 223, 225, 330] are the remaining essential concepts in this thesis. As they have revolutionised the field of image processing, a plethora of modern image processing models rely on them. To understand their success, we briefly discuss neural network history and give a basic introduction into the most popular modern architectures and their practical implementations.

After having reviewed diffusion, wavelets, variational methods, and neural networks, we then focus on the connections between them in Section 3.5. Such considerations have recently gained a lot of traction. We discuss a subset of relevant works and consequently motivate our contributions to this field.

Lastly, in Section 3.6 we introduce the basic goals of denoising and inpainting as applications which we deal with throughout the thesis. Removing noise from a corrupted image with diffusion filters is the prototype of a well-posed process and thus often serves as a simple test problem for the proposed concepts. Inpainting is a more challenging problem which has the goal of restoring images from only a limited amount of data [46, 251, 378]. This problem goes hand in hand with image compression, as inpainting strategies have emerged as a serious competitor to transform-based ideas [135, 327, 329].

3.1 DIFFUSION

Diffusion is the physical process of equilibration of concentration differences in a closed system. If particles of a certain type are more concentrated in some part of the system, Brownian motion induces a transport process to areas with lower concentrations. Intuitively, the probability that particles move from regions with high concentration to regions with low concentration is higher than vice versa. Thus, at some point in time, the particles will be evenly distributed.

This transport process can be modelled by Fick's law [125]:

$$\boldsymbol{j} = -\boldsymbol{D}\boldsymbol{\nabla} u \qquad \text{on } \Omega \times [0, \infty). \tag{3.1}$$

Here, the function $u : \Omega \times [0, \infty) \to \mathbb{R}$ maps spatial positions from a domain $\Omega \subset \mathbb{R}^n$ at a time $t \in [0, \infty)$ to concentration values. The gradient $\boldsymbol{\nabla} u$ denotes the steepest ascent of u, i.e. it points from low concentrations towards high ones. Therefore, Fick's law states that the *flux* $\boldsymbol{j}$ is oriented opposite to the concentration gradient, inducing an equilibration.

This process can additionally be influenced by the medium. In an *isotropic* medium, a particle is equally likely to move in any direction. In *anisotropic* media, movement along certain directions is more likely than others. This is modelled by the *diffusion tensor* $\boldsymbol{D}$ which is a symmetric positive semi-definite 2×2 matrix

$$\boldsymbol{D} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \tag{3.2}$$

with scalar entries $a, b, c \in \mathbb{R}$. If $a = c$ and $b = 0$, the process is isotropic, otherwise it is anisotropic. It is also possible that the entries depend on the evolving image u.

A second important property of diffusion processes is that mass is preserved: In a closed system, no particles are destroyed or created. This can be modelled by the *continuity equation*

$$\partial_t u = -\boldsymbol{\nabla}^\top \boldsymbol{j} \qquad \text{on } \Omega \times [0, \infty) \tag{3.3}$$

together with the boundary conditions

$$\partial_{\boldsymbol{n}} \boldsymbol{j} = 0 \qquad \text{on } \partial\Omega \times [0, \infty). \tag{3.4}$$

The continuity equation states that the temporal change of the concentration is equal to the negated divergence of the flux. Thus, at a steady state when $t \to \infty$ and $\partial_t u \to 0$, the divergence must also attain zero. This means that the flux vector field does not exhibit any sources or sinks.

The boundary conditions imply *reflecting* or *homogeneous Neumann boundary conditions* on the flux. The flux in the direction of the *outer*

normal vector $\boldsymbol{n}$ on the *domain boundary* $\partial\Omega$ must be zero, indicating that no mass enters or leaves the domain Ω.

In conjunction, this implies that the diffusion process preserves the mass that is initially present in the system over time.

Substituting the flux in the continuity equation (3.3) with the result from Fick's law (3.1) yields the diffusion equation

$$\partial_t u = \boldsymbol{\nabla}^\top (\boldsymbol{D} \boldsymbol{\nabla} u) \qquad \text{on } \Omega \times [0, \infty) \tag{3.5}$$

with boundary conditions

$$\partial_{\boldsymbol{n}} (\boldsymbol{D} \boldsymbol{\nabla} u) = 0 \qquad \text{on } \partial\Omega \times [0, \infty). \tag{3.6}$$

The diffusion equation is one of the fundamental PDEs in physics. In the following, we show how it can be used as a tool for image processing.

DIFFUSION FOR IMAGE PROCESSING For image processing we consider diffusion in the two-dimensional setting. For a family of continuous grey value images $u : \Omega \times [0, \infty) \to \mathbb{R}$, one obtains the initial boundary value problem

$$\partial_t u = \boldsymbol{\nabla}^\top (\boldsymbol{D} \boldsymbol{\nabla} u) \qquad \text{on } \Omega \times [0, \infty), \tag{3.7}$$

$$\partial_{\boldsymbol{n}} (\boldsymbol{D} \boldsymbol{\nabla} u) = 0 \qquad \text{on } \partial\Omega \times [0, \infty), \tag{3.8}$$

$$u(\boldsymbol{x}, 0) = f(\boldsymbol{x}) \qquad \text{on } \Omega. \tag{3.9}$$

Equation (3.7) is the aforementioned general diffusion equation. The reflecting or homogeneous Neumann boundary conditions are expressed by Equation (3.8). Lastly, Equation (3.9) states that the evolving image u at time $t = 0$ is initialised with an input image f.

For image processing, we interpret the concentrations as grey values of an image. This in turn means that the continuity equation implies that the average grey value of u remains constant over the evolution. In fact, one can show that the steady state of diffusion problems of this form is a flat image with the average grey value of the input image f [369].

TERMINOLOGY Before we present a selection of historically important diffusion models, it is helpful to introduce some terminology to describe different aspects of these models. One distinguishes *linearity*, *homogeneity*, and *isotropy* of a diffusion model.

In a linear model, the diffusion tensor $\boldsymbol{D}$ does not depend on the evolving image u. Consequently, a nonlinear model adapts $\boldsymbol{D}(u)$ during the evolution. In a physical interpretation, a nonlinear diffusion process changes the conditions within the medium over time.

Homogeneity describes the spatial configuration of the diffusion tensor $\boldsymbol{D}$. If it varies throughout the domain, the diffusion process is

inhomogeneous, otherwise it is homogeneous. Mathematically speaking, in the inhomogeneous case one uses $\boldsymbol{D} = \boldsymbol{D}(x, y)$, whereas in the homogeneous case $\boldsymbol{D}$ is constant over Ω.

Isotropy denotes whether some directions of movement for a particle are more likely than others. If all directions are equally probable, the diffusion process is isotropic. Otherwise, it is anisotropic. In the isotropic case, $\boldsymbol{D}$ can be replaced by a scalar *diffusivity* value.

Not all combinations of these terminologies are equally prominent. Most nonlinear models are also inhomogeneous, and anisotropic models are usually also nonlinear.

3.1.1 *Popular Diffusion Models*

In the following, we review several popular diffusion models for image processing. If not specified further, all models employ reflecting boundary conditions and are initialised with an input image f.

HOMOGENEOUS LINEAR DIFFUSION The simplest diffusion model is *homogeneous linear diffusion*. While often associated with Witkin's scale space concept [394], Weickert et al. [375] showed that it was first proposed for image processing by Iijima [194–196]. It follows the linear PDE

$$\partial_t u = \Delta u \,. \tag{3.10}$$

It arises from the more general diffusion PDE (3.7) by setting the diffusion tensor $\boldsymbol{D} = \boldsymbol{I}$ to the identity. As this model is homogeneous and isotropic, it performs equal smoothing in all directions at all positions of the image.

While homogeneous diffusion benefits from the fact that it is simple and parameter-free, it does not adapt to the image at hand. It cannot preserve important structural details such as edges, e.g. in a denoising framework.

NONLINEAR ISOTROPIC DIFFUSION To adapt the diffusion process to the local image structure, Perona and Malik [279] presented a nonlinear isotropic diffusion model which creates evolving images according to the PDE

$$\partial_t u = \boldsymbol{\nabla}^\top \left(g\left(|\boldsymbol{\nabla} u|^2 \right) \boldsymbol{\nabla} u \right) . \tag{3.11}$$

The introduction of a nonlinear *diffusivity function* allows to inhibit diffusion around important image structures. In this model, image structures are measured with the gradient magnitude $|\boldsymbol{\nabla} u|^2$ as a fuzzy edge detector.

Note that the Perona–Malik model has been introduced as 'anisotropic'. However, in our terminology, it is isotropic as it uses a scalar

diffusivity function. Thus diffusion may be inhibited at certain positions, but transport at a specific position is still performed equally in all directions.

Typically, one employs nonincreasing, nonnegative, and bounded diffusivities $g(s^2)$. For example, the Charbonnier [70] diffusivity

$$g(s^2) = \frac{1}{\sqrt{1 + \frac{s^2}{\lambda^2}}} \tag{3.12}$$

with a contrast parameter λ dampens diffusion around image edges.

The rational Perona–Malik diffusivity [279]

$$g(s^2) = \frac{1}{1 + \frac{s^2}{\lambda^2}} \tag{3.13}$$

even allows enhancement of image edges due to implicit backward diffusion.

While both the Charbonnier and the Perona–Malik diffusivities seem quite similar at first glance, they have drastically different effects. This becomes apparent when investigating the *flux function* which is associated with the diffusivity via

$$\Phi(s) = g(s^2)\, s\,. \tag{3.14}$$

Figure 3.1 visualises three diffusivities and their associated flux functions. We see that the Charbonnier flux is monotone, while the Perona–Malik flux is nonmonotone. Consequently, the derivative of the latter is negative for $s > \lambda$. As the flux denotes the direction of transport, this means that information is transported *towards* dominant image structures, rather than away from them. A diffusivity which decays even quicker, leading to more implicit backward diffusion is e.g. the exponential Perona–Malik diffusivity [279]

$$g(s^2) = \exp\left(-\frac{s^2}{2\lambda^2}\right). \tag{3.15}$$

The well-known total variation (TV) [16, 313] diffusivity

$$g(s^2) = \frac{1}{|s|} \tag{3.16}$$

constitutes the boundary between forward and backward diffusion. The associated flux is the sign function, resulting in a constant amount of flux independent of the image structure. However, the TV diffusivity is unbounded, requiring sophisticated solution strategies [36, 66] or regularisations [2].

Implicit backward diffusion should not be confused with explicit backward diffusion, where the flux and the diffusivity function become negative. This reverses the diffusion process, thus becoming highly

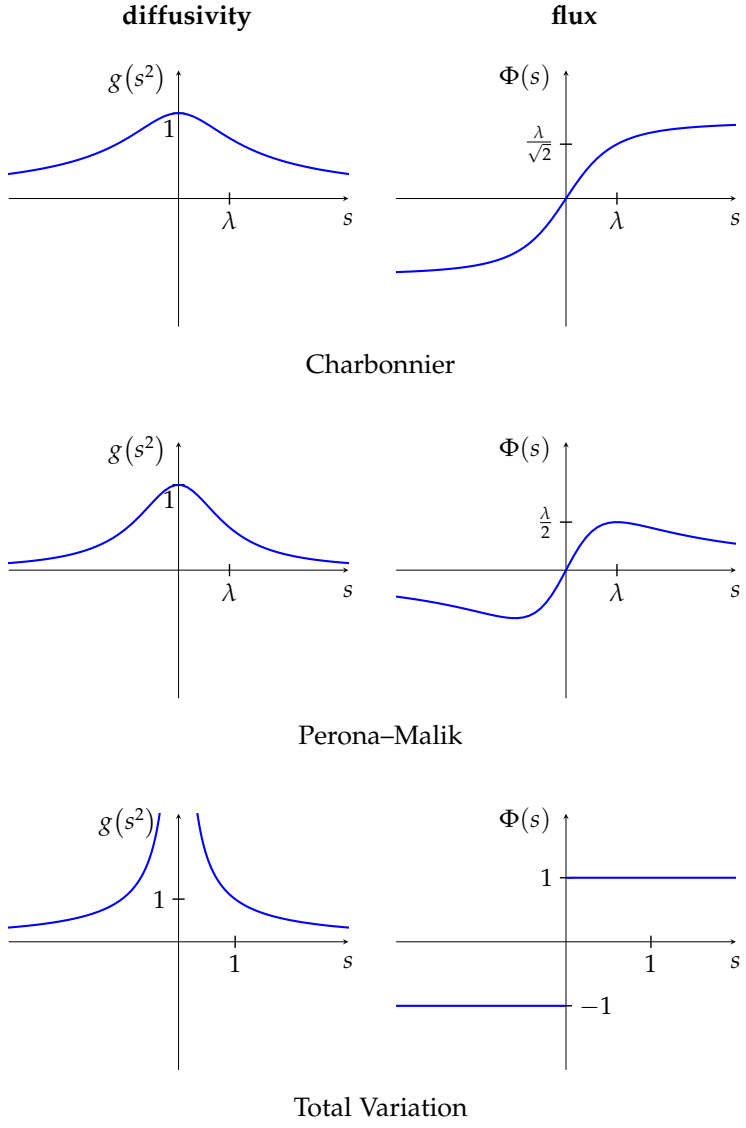


Figure 3.1: Visualisation of Charbonnier, Perona–Malik, and total variation (TV) diffusivities and the associated flux functions. The Charbonnier diffusivity results in a monotone flux function, while the Perona–Malik flux is nonmonotone. The latter allows implicit backward diffusion and leads to edge enhancement. The TV diffusivity is at the boundary between forward and backward diffusion, however, the diffusivity is unbounded.

ill-posed and requiring to come up with sophisticated strategies to circumvent arising stability problems; see e.g. [44, 147, 386].

Diffusivities are one of the central design choices for nonlinear diffusion models. Throughout the years, various diffusivities with individual properties have been proposed. We analyse a popular subset of them in Chapter 6 and connect them to neural network activation functions (see Section 3.4).

REGULARISED NONLINEAR ISOTROPIC DIFFUSION As the continuous Perona–Malik model is not well-posed (see e.g. [119, 211, 410]), several regularisations have been proposed. Catté et al. [65] suggest to replace the gradient operator within the diffusivity argument by a Gaussian-smoothed gradient

$$\partial_t u = \nabla^\top \left(g\left(|\nabla_\sigma u|^2 \right) \nabla u \right), \tag{3.17}$$

where $\nabla_\sigma u = \nabla (K_\sigma * u)$ denotes the gradient of an image which is presmoothed by a convolution with a Gaussian K_σ with standard deviation σ.

This presmoothing yields an infinitely often differentiable result and thus avoids unbounded gradients in the diffusivity argument. However, this comes at the cost of not being able to represent the PDE as the gradient descent of a variational energy [384] (see also Section 3.2).

An alternative regularisation is provided by Niessen et al. [269] who propose to regularise all associated derivatives by

$$\partial_t u = \nabla_\sigma^\top \left(g\left(|\nabla_\sigma u|^2 \right) \nabla_\sigma u \right). \tag{3.18}$$

While this model has a variational formulation [326], it does not preserve any image features which live on a scale smaller than σ as they are smoothed out.

LINEAR INHOMOGENEOUS DIFFUSION An exotic representative for a linear inhomogeneous model was proposed by Fritsch [130] in the context of medical image analysis. Instead of measuring image structures based on the gradient of the evolving image, Fritsch suggests to use the gradient of the initial image f which leads to

$$\partial_t u = \nabla^\top \left(g\left(|\nabla f|^2 \right) \nabla u \right). \tag{3.19}$$

As the diffusivity varies throughout the image domain, this model is inhomogeneous. However, the diffusivity is not changing over the complete evolution, resulting in a linear model. Similar ideas using structural information for colour propagation have been presented by Peter et al. [282] in a linear inhomogeneous anisotropic colourisation model.

HIGHER ORDER DIFFUSION So far, all models rely on PDEs of second order, i.e. the highest involved derivative order is two. Models of higher order benefit from better smoothness conditions, but come at the cost of computational expense.

A simple extension of the linear homogeneous diffusion model of Iijima is obtained when replacing the Laplace operator by the biharmonic operator:

$$\partial_t u = \Delta^2 u = \left(\partial_{xxxx} + 2\,\partial_{xxyy} + \partial_{yyyy}\right) u\,. \tag{3.20}$$

The biharmonic diffusion model is linear, homogeneous, and isotropic, but of order four.

To introduce nonlinearity in the biharmonic diffusion model, You and Kaveh [401] design a Perona–Malik-like process as follows:

$$\partial_t u = -\Delta \left(g\Big((\Delta u)^2 \Big)\, \Delta u \right). \tag{3.21}$$

They replace the gradient and divergence by the Laplace operator, and image structure is measured by the squared Laplacian of the image $(\Delta u)^2$. Note that the squared Laplacian is different from the biharmonic operator: The biharmonic operator contains fourth-order derivatives, while the squared Laplacian only considers products of second-order derivatives.

As this model does not consider mixed derivatives of second order, Lysaker et al. [242] suggest to use the Frobenius norm of the Hessian:

$$\partial_t u = -\mathcal{D}^\top \left(g\big(\|\boldsymbol{H}(u)\|_F^2 \big)\, \mathcal{D} u \right) \tag{3.22}$$

where the differential operator $\mathcal{D}$ induced by the Frobenius norm reads

$$\mathcal{D} = \left(\partial_{xx}, \partial_{xy}, \partial_{yx}, \partial_{yy}\right)^\top. \tag{3.23}$$

This avoids speckle artefacts that can arise in the model of You and Kaveh. Both models are nonlinear, inhomogeneous, isotropic, and of fourth order.

However, PDEs of fourth and higher order are rarely found in the literature; exceptions include [33, 67, 205, 282]. As condition numbers of discrete higher order operators are usually much larger, convergence of numerical solvers is drastically slowed down.

The different options of measuring structure within the diffusivity are discussed more in detail in Chapter 8 where we discuss rotationally invariant design of diffusion models and their connections to CNNs.

NONLINEAR ANISOTROPIC DIFFUSION Up to this point, all presented diffusion models were isotropic. However, anisotropic models can have tremendous advantages as they can transport information along certain directions.

The *edge-enhancing diffusion (EED)* model of Weickert [368] is designed to smooth images along dominant structures, while preventing

diffusion across them. It is the first among the presented models which makes full use of the diffusion tensor by creating evolving images according to the PDE

$$\partial_t u = \nabla^\top (\boldsymbol{D}(\nabla_\sigma u)\, \nabla u)\,. \tag{3.24}$$

The diffusion tensor is constructed from its eigenvalues ν_1, ν_2 and its normalised eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ as follows: The dominant eigenvector $\boldsymbol{v}_1 \parallel \nabla_\sigma u$ is parallel to the regularised gradient of the evolving image, i.e. it points across dominant image structures. Consequently, the second eigenvector $\boldsymbol{v}_2 \perp \nabla_\sigma u$ is orthogonal to it, pointing along structures.

To inhibit the diffusion across image structures, the eigenvalue ν_1 to the dominant eigenvector $\boldsymbol{v}_1$ is computed with the help of a diffusivity $\nu_1 = g(|\nabla_\sigma u|^2)$. The choice of diffusivity strongly depends on the application. For denoising, Weickert [369] proposes a diffusivity which decays even faster than the exponential Perona–Malik diffusivity [279]. For inpainting, the Charbonnier diffusivity is more suited as it does not perform implicit backward diffusion [135].

As diffusion along dominant structures should not be inhibited, the second eigenvalue is set to $\nu_2 = 1$. Thus, the diffusion tensor is fully defined by

$$\boldsymbol{D} = g\Big(|\nabla_\sigma u|^2\Big)\, \boldsymbol{v}_1 \boldsymbol{v}_1^\top + 1\, \boldsymbol{v}_2 \boldsymbol{v}_2^\top. \tag{3.25}$$

As in the model of Catté et al. [65], a smoothing scale σ is involved in the gradient computation. This is a crucial component for creating anisotropy. If the diffusion tensor is constructed without this smoothing, one of its eigenvectors is the image gradient ∇u itself. Then the divergence term collapses:

$$\begin{aligned}
&\nabla^\top (\boldsymbol{D}(\nabla u)\, \nabla u) \\
&\quad = \nabla^\top \left(\left(g\Big(|\nabla u|^2\Big)\, \boldsymbol{v}_1 \boldsymbol{v}_1^\top + 1\, \boldsymbol{v}_2 \boldsymbol{v}_2^\top \right) \nabla u \right) \\
&\quad = \nabla^\top \left(\left(g\Big(|\nabla u|^2\Big)\, \frac{\nabla u}{|\nabla u|} \frac{(\nabla u)^\top}{|\nabla u|} + 1\, \frac{\nabla^\perp u}{|\nabla^\perp u|} \frac{(\nabla^\perp u)^\top}{|\nabla^\perp u|} \right) \nabla u \right) \\
&\quad = \nabla^\top \left(g\Big(|\nabla u|^2\Big)\, \frac{\nabla u}{|\nabla u|} \frac{|\nabla u|^2}{|\nabla u|} \right) \\
&\quad = \nabla^\top \Big(g\Big(|\nabla u|^2\Big)\, \nabla u \Big)\,.
\end{aligned} \tag{3.26}$$

One half of the diffusion tensor does not contribute since the eigenvector $\boldsymbol{v}_2$ is orthogonal to the gradient ∇u and the scalar product of the two vanishes. The first eigenvector coincides with the gradient and the scalar product yields the gradient magnitude, which cancels out with the normalisation of the eigenvectors.

The resulting model is the isotropic Perona–Malik model. This elegantly demonstrates how critical the smoothing scale is for the

Table 3.1: Overview over the properties of popular diffusion models.

model	Eq.	linear	homogeneous	isotropic	order
Iijima	(3.10)	✓	✓	✓	2
Perona–Malik	(3.11)	✗	✗	✓	2
Catté et al.	(3.17)	✗	✗	✓	2
Niessen et al.	(3.18)	✗	✗	✓	2
Fritsch	(3.19)	✓	✗	✓	2
Biharmonic	(3.20)	✓	✓	✓	4
You and Kaveh	(3.21)	✗	✗	✓	4
Lysaker et al.	(3.22)	✗	✗	✓	4
EED, Weickert	(3.24)	✗	✗	✗	2

creation of anisotropy. However, as for the model of Catté et al., this comes at the cost of losing a conventional energy formulation for the EED model [384].

The EED model appears throughout the thesis: It serves as a baseline for our multiscale anisotropic diffusion model in Chapter 5, and we adapt its core ideas to Shepard interpolation for inpainting in Chapter 9. Finally, we learn a mask generation model for EED inpainting [378] in Chapter 10.

This concludes our review of popular diffusion models. While there are several other anisotropic models such as *coherence-enhancing diffusion* [370], *Bi-EED* [282], and others [205, 311, 325, 358, 373], they are out of the scope of this review.

Table 3.1 provides an overview of the properties and orders of the presented diffusion models. Not all combinations of linearity, homogeneity, and isotropy appear, as certain combinations are less useful in practice. For example, an anisotropic but linear and homogeneous process may be able to transport mass along certain directions, but these directions are neither changing during the evolution, nor are they adaptive to the image structure.

Figure 3.2 visualises the effects of the diffusion models from Table 3.1 on the test image *peppers* [342]. For all models, we have chosen a diffusion time of $T = 15$. The contrast parameter λ, and smoothing scale σ have been manually set to highlight the different effects of each model.

The linear isotropic models of homogeneous and biharmonic diffusion introduce a blurring effect that is independent of the image content. The nonlinear isotropic models preserve edges within the image to a different degree, and smooth homogeneous regions.

While the model of Catté does not suffer from staircasing effects as the Perona–Malik model does, it blurs edges in regions with low contrast due to the regularisation. As the model of Niessen regularises

all involved derivatives, it additionally suffers from halo artefacts at edges.

The higher-order models of You and Kaveh as well as Lysaker et al. show a higher amount of smoothness in homogeneous regions. Finally, the EED model is able to enhance edges by smoothing along them, while simultaneously smoothing homogeneous regions.

3.1.2 *Solving Diffusion Problems*

In practice one requires numerical algorithms to solve the diffusion PDEs. To this end, we use finite differences (see e.g. [228]). The core idea is to approximate derivatives by means of differences of sampled image positions. In the following, we give a short introduction to finite differences, more advanced discretisations for anisotropic models, and finally several basic numerical algorithms.

FINITE DIFFERENCES We first show how finite differences can be used to approximate derivatives in the one-dimensional setting. To this end, we obtain a discrete signal $\boldsymbol{f} \in \mathbb{R}^n$ by sampling the continuous signal f with grid size h as described in Section 2.5. To approximate the first order derivative of $\boldsymbol{f}$ at a position i using only f_i and its direct neighbours f_{i+1}, f_{i-1}, one has three options:

$$f_i' \approx \frac{f_{i+1} - f_i}{h} \qquad \text{(forward difference)}, \tag{3.27}$$

$$f_i' \approx \frac{f_i - f_{i-1}}{h} \qquad \text{(backward difference)}, \tag{3.28}$$

$$f_i' \approx \frac{f_{i+1} - f_{i-1}}{2h} \qquad \text{(central difference)}. \tag{3.29}$$

The *forward* and *backward differences* approximate the derivative by means of f_i and its neighbour in the respective direction. The central difference, on the other hand, takes the difference of both neighbours. To compensate the larger distance between the two, the difference is divided by $2h$ instead of h.

How can we judge the quality of these approximations? To this end, we make use of the Taylor expansion (see Section 2.3) at the example of the forward difference. First, we evolve f_{i+1} in the position i:

$$f_{i+1} = f_i + h f_i' + \frac{h^2}{2} f_i'' + \mathcal{O}\left(h^3\right). \tag{3.30}$$

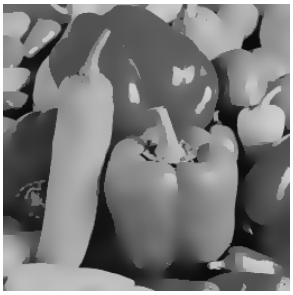
Plugging this expression into the forward difference yields

$$\frac{f_{i+1} - f_i}{h} = f_i' + \frac{h}{2} f_i'' + \mathcal{O}\left(h^2\right). \tag{3.31}$$

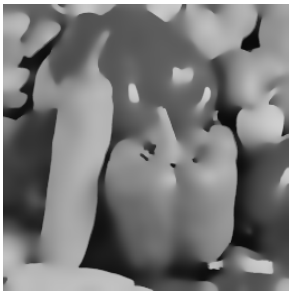
This shows that the forward difference indeed approximates f_i'. For $h \to 0$, all remaining terms vanish and the expression converges to the continuous derivative. Thus, this approximation is *consistent*.

original

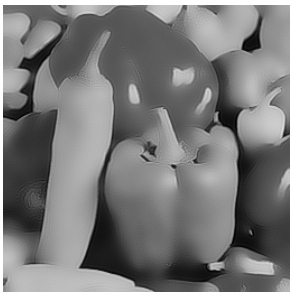
Iijima



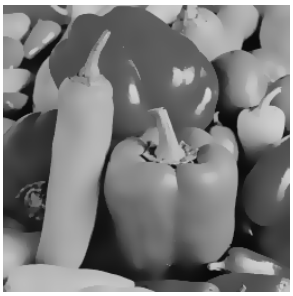
Perona–Malik



Catté et al.



Niessen et al.



Fritsch

Biharmonic

You and Kaveh

Lysaker et al.

EED, Weickert

Figure 3.2: Visualising the effect of the diffusion models from Table 3.1 on the *peppers* image. All models use a diffusion time of $T = 15$ and the exponential Perona–Malik diffusivity. If required, all models except EED use the parameters $\lambda = 10$ and $\sigma = 1$. For EED, we set $\lambda = 1$ and $\sigma = 1$.

Moreover, we find that the first term of the remainder is given by $\frac{h}{2} f_i''$. This *leading error term* shows the *consistency order* of the approximation. As the grid size appears with a power of one in this term, the consistency order of the forward difference is one. Higher consistency orders denote better quality approximations. Analogously, the backward difference also has a consistency order of one. The central difference, however, has a consistency order of two.

There exists a general strategy to derive finite difference expressions for arbitrary derivative operators with any selection of signal positions. We exercise this strategy at the example of the second derivative f_i'' using the points f_{i-1}, f_i, and f_{i+1}.

Our goal is to obtain weights $\alpha_{-1}, \alpha_0, \alpha_1$ for the respective positions such that

$$\alpha_{-1} f_{i-1} + \alpha_0 f_i + \alpha_1 f_{i+1} = 0\, f_i + 0\, f_i' + 1\, f_i''. \tag{3.32}$$

To obtain a system of equations in the coefficients α, we express all signal positions in terms of f_i by means of the Taylor expansion:

$$f_{i-1} = f_i - h f_i' + \frac{h^2}{2} f_i'' + \mathcal{O}\left(h^3\right) \tag{3.33}$$

$$f_i = f_i \tag{3.34}$$

$$f_{i+1} = f_i + h f_i' + \frac{h^2}{2} f_i'' + \mathcal{O}\left(h^3\right) \tag{3.35}$$

Plugging the expressions into (3.32) yields

$$\begin{aligned} &\left(\alpha_{-1} + \alpha_0 + \alpha_1\right) f_i \\ &+h \left(-\alpha_{-1} + \alpha_1\right) f_i' \\ &+\frac{h^2}{2} \left(\alpha_{-1} + \alpha_1\right) f_i'' + \mathcal{O}\left(h^3\right) = 0\, f_i + 0\, f_i' + 1\, f_i''. \end{aligned} \tag{3.36}$$

Dividing by $\frac{h^2}{2}$ and comparing the coefficients in front of the derivatives in f_i yields the linear system

$$\begin{pmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_{-1} \\ \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \frac{2}{h^2} \end{pmatrix}. \tag{3.37}$$

Solving this system results in the derivative approximation

$$f_i'' \approx \frac{f_{i-1} - 2 f_i + f_{i+1}}{h^2}. \tag{3.38}$$

A consistency analysis shows that this approximation has consistency order two:

$$\frac{f_{i-1} - 2 f_i + f_{i+1}}{h^2} = f_i'' + \frac{h^2}{12} f_i'''' + \mathcal{O}\left(h^4\right). \tag{3.39}$$

Larger consistency orders can be achieved by involving more signal positions. A consistent approximation of a derivative of order n requires always at least $n+1$ points. Central differences, i.e. those that involve neighbouring points in a symmetrical manner usually have a higher consistency order as terms with different signs in the Taylor expansion cancel out.

What if one wants to compute a derivative not only in a single position, but in all of them? Here the concept of *stencils* comes in handy. A stencil is a compact representation of the weights arranged by their relative positioning. For example, we introduce the stencil representation of the second order approximation (3.38) as

$$\begin{array}{c} \dfrac{f_{i-1} - 2f_i + f_{i+1}}{h^2} = \dfrac{1}{h^2} f_{i-1} - \dfrac{2}{h^2} f_i + \dfrac{1}{h^2} f_{i+1} \\ \downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow \\ \boxed{\dfrac{1}{h^2}} \quad \boxed{-\dfrac{2}{h^2}} \quad \boxed{\dfrac{1}{h^2}} \end{array} . \tag{3.40}$$

Applying this stencil to a full signal $\boldsymbol{f} \in \mathbb{R}^n$ yields the following matrix-vector multiplication:

$$\boldsymbol{f}'' \approx \frac{1}{h^2} \begin{pmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{pmatrix} \boldsymbol{f} = \boldsymbol{A}\boldsymbol{f} . \tag{3.41}$$

The matrix $\boldsymbol{A} \in R^{n \times n}$ contains the shifted stencil entries in each row. All other entries are zero. This computation can be expressed efficiently by a discrete convolution with the stencil as a convolution kernel.

At the boundaries of the signal, the matrix entries depend on the choice of boundary conditions. Here, we have chosen to mirror the signal such that $f_0 = f_1$ and $f_{n+1} = f_n$. This corresponds to reflecting boundary conditions. Consequently, the neighbour weights in the respective boundary direction coincide with the central entry.

Extending the above approaches to the two-dimensional setting is analogous. All we need is the two-dimensional Taylor expansion. For example, approximating the Laplacian Δf of an image $f : \Omega \to \mathbb{R}$ on a domain $\Omega \subset \mathbb{R}^2$ in the position i, j involving its direct neighbours yields

$$\Delta f \approx \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{h_x^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{h_y^2} . \tag{3.42}$$

The corresponding stencil is given by

$$\begin{array}{|c|c|c|}\hline 0 & \frac{1}{h_y^2} & 0 \\ \hline \frac{1}{h_x^2} & -\frac{2}{h_x^2}-\frac{2}{h_y^2} & \frac{1}{h_x^2} \\ \hline 0 & \frac{1}{h_y^2} & 0 \\ \hline \end{array}. \tag{3.43}$$

As the continuous setting would suggest, the approximation is simply the sum of the one-dimensional derivatives in x and y-direction. Even though the consistency order of this approximation is two, it is unsatisfactory as the leading error term has a directional bias:

$$\begin{aligned}\frac{f_{i-1,j}-2f_{i,j}+f_{i+1,j}}{h_x^2}+\frac{f_{i,j+1}-2f_{i,j}+f_{i,j-1}}{h_y^2}\\ =\Delta f_{i,j}-\frac{h^2}{12}\left(\partial_{xxxx}f_{i,j}+\partial_{yyyy}f_{i,j}\right)+\mathcal{O}\left(h^4\right).\end{aligned} \tag{3.44}$$

The benefit of investing more effort into sophisticated discretisations is often underestimated. However, in many tasks a good discretisation can make the difference between a good and a bad result. To this end, we investigate a general-purpose discretisation for diffusion processes in the following.

DISCRETISING ANISOTROPIC DIFFUSION Transferring mathematical guarantees such as rotation invariance from a continuous model to a discrete implementation is not trivial. Standard discretisations rarely present the best choice when it comes to this task. Especially in anisotropic diffusion models, a good discretisation of the divergence term is critical.

To this end, Weickert et al. [379] present a discretisation for the divergence term in anisotropic diffusion processes of the form

$$\partial_t u = \boldsymbol{\nabla}^\top(\boldsymbol{D}\boldsymbol{\nabla}u) \tag{3.45}$$

with a diffusion tensor

$$\boldsymbol{D}=\begin{pmatrix}a(x,y) & b(x,y)\\ b(x,y) & c(x,y)\end{pmatrix} \tag{3.46}$$

based on nonstandard finite differences [255]. In the following, we briefly review the essential components of this discretisation.

Following Weickert et al. [379], we also introduce shorthand notations

$$\begin{aligned}\boxed{\cdot\!-\!\cdot} &= \frac{u_{i+1,j}-u_{i,j}}{h},\\ \boxed{\cdot\!-\!\cdot} &= \frac{u_{i+1,j+1}-u_{i,j+1}}{h},\\ \boxed{\cdot\,|\,\cdot} &= \frac{u_{i,j+1}-u_{i,j}}{h},\\ \boxed{\cdot\,|\,\cdot} &= \frac{u_{i+1,j+1}-u_{i+1,j}}{h},\end{aligned} \tag{3.47}$$

for finite differences around an off-grid position $i+\frac{1}{2}, j+\frac{1}{2}$.

Weickert et al. [379] derive their discretisation from a gradient descent of an energy. The components of the energy are discretised as follows:

$$\left(u_x^2\right)_{i+\frac{1}{2},j+\frac{1}{2}} \approx \left(1-\alpha_{i+\frac{1}{2},j+\frac{1}{2}}\right)\frac{1}{2}\left(\boxed{}^2+\boxed{}^2\right) + \alpha_{i+\frac{1}{2},j+\frac{1}{2}}\boxed{}\cdot\boxed{}\,, \tag{3.48}$$

$$\left(u_y^2\right)_{i+\frac{1}{2},j+\frac{1}{2}} \approx \left(1-\alpha_{i+\frac{1}{2},j+\frac{1}{2}}\right)\frac{1}{2}\left(\boxed{}^2+\boxed{}^2\right) + \alpha_{i+\frac{1}{2},j+\frac{1}{2}}\boxed{}\cdot\boxed{}\,, \tag{3.49}$$

$$\left(u_x u_y\right)_{i+\frac{1}{2},j+\frac{1}{2}} \approx \frac{1-\beta_{i+\frac{1}{2},j+\frac{1}{2}}}{2}\frac{1}{2}\left(\boxed{}\cdot\boxed{}+\boxed{}\cdot\boxed{}\right) + \frac{1+\beta_{i+\frac{1}{2},j+\frac{1}{2}}}{2}\frac{1}{2}\left(\boxed{}\cdot\boxed{}+\boxed{}\cdot\boxed{}\right). \tag{3.50}$$

A space-varying parameter $\alpha_{i+\frac{1}{2},j+\frac{1}{2}}$ controls the balance between arithmetic and geometric mean in the first two equations. Similarly, a space-varying parameter $\beta_{i+\frac{1}{2},j+\frac{1}{2}}$ steers the contribution of the different options for discretising $\left(u_x u_y\right)_{i+\frac{1}{2},j+\frac{1}{2}}$.

Computing the discrete divergence term as the gradient of the discrete energy, they arrive at the stencil

$$\frac{1}{2h^2}\begin{array}{|c|c|c|}\hline [(\beta-1)b+\alpha(a+c)]_{-+} & \begin{array}{c}[(1-\alpha)c-\alpha a-\beta b]_{++}\\ +[(1-\alpha)c-\alpha a-\beta b]_{-+}\end{array} & [(\beta+1)b+\alpha(a+c)]_{++} \\ \hline \begin{array}{c}[(1-\alpha)a-\alpha c-\beta b]_{-+}\\ +[(1-\alpha)a-\alpha c-\beta b]_{--}\end{array} & \begin{array}{c}[(1-\alpha)(a+c)-(\beta-1)b]_{++}\\ -[(1-\alpha)(a+c)-(\beta-1)b]_{+-}\\ -[(1-\alpha)(a+c)-(\beta-1)b]_{-+}\\ -[(1-\alpha)(a+c)-(\beta-1)b]_{--}\end{array} & \begin{array}{c}[(1-\alpha)a-\alpha c-\beta b]_{++}\\ +[(1-\alpha)a-\alpha c-\beta b]_{+-}\end{array} \\ \hline [(\beta+1)b+\alpha(a+c)]_{--} & \begin{array}{c}[(1-\alpha)c-\alpha a-\beta b]_{+-}\\ +[(1-\alpha)c-\alpha a-\beta b]_{--}\end{array} & [(\beta-1)b+\alpha(a+c)]_{+-} \\ \hline\end{array} \tag{3.51}$$

where $h = h_x = h_y$. Here we have used the abbreviation $++$ to denote the index $i+\frac{1}{2}, j+\frac{1}{2}$ and analogously for the other abbreviations. As this stencil is not symmetric w.r.t. the central entry, it is important to note that the upper row corresponds to entries in row $j+1$, and the right column is concerned with entries in column $i+1$.

To obtain a discrete diffusion process which is stable in the L^2-norm, the parameters have to obey $0 \leq \alpha \leq 0.5$ and $|\beta| \leq 1-2\alpha$. Weickert et al. suggest to replace

$$\beta = \gamma\,(1-2\alpha)\,\operatorname{sgn}(b) \tag{3.52}$$

with a parameter $\gamma \in [-1,1]$. This can help to reduce over- and undershoots of anisotropic processes. Note that for isotropic processes, the choice of β is irrelevant as the diffusion tensor entry b is zero.

Weickert et al. show that for a demosaicking task, varying choices of α and γ can make up a difference of up to 10 dB in PSNR. However, finding good parameters is still dependent on the task at hand,

and sometimes involves trading quality for rotation invariance as we highlight in Chapter 8. As this discretisation is flexible and subsumes various others, it will be our standard choice whenever we deal with diffusion processes in this thesis.

EXPLICIT SCHEMES We are now in the position to present the first numerical algorithm for diffusion problems. To this end, we consider a general diffusion problem of the form

$$\partial_t u = \boldsymbol{\nabla}^\top (\boldsymbol{D} \boldsymbol{\nabla} u) \tag{3.53}$$

with a positive diffusion tensor $\boldsymbol{D}$, initial condition $u(\cdot, 0) = f$ and reflecting boundary conditions.

The *explicit Euler scheme* for this PDE is obtained as follows. One discretises the temporal derivative by a forward difference with a *time step size* τ. Moreover, let $\boldsymbol{A}(\boldsymbol{u}^k)\boldsymbol{u}^k$ implement the divergence term for a discrete signal $\boldsymbol{u}^k$ at a time step k. Then a discrete version of (3.53) is given by

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} = \boldsymbol{A}(\boldsymbol{u}^k)\, \boldsymbol{u}^k. \tag{3.54}$$

Solving this equation for the image $\boldsymbol{u}^{k+1}$ at the new time step yields the explicit scheme

$$\boldsymbol{u}^{k+1} = \Big(\boldsymbol{I} + \tau \boldsymbol{A}(\boldsymbol{u}^k)\Big)\, \boldsymbol{u}^k. \tag{3.55}$$

Starting with $\boldsymbol{u}^0 = \boldsymbol{f}$, this iterative solution strategy computes the temporally evolving signals $\boldsymbol{u}^k$ at discrete points $t = k\tau$ in time. However, these time steps cannot be chosen arbitrarily large. In the following, we present two views on the *stability* of the explicit scheme.

Weickert [369] presents an elaborate theory for the theoretical guarantees of diffusion problems in the continuous, semi-discrete, and the discrete setting. In the latter, he considers a discrete evolution of the form

$$\boldsymbol{u}^{k+1} = \boldsymbol{Q}(\boldsymbol{u}^k)\, \boldsymbol{u}^k \tag{3.56}$$

where $\boldsymbol{u}^0 = \boldsymbol{f}$. This evolution can be identified with the explicit scheme (3.55) by setting $\boldsymbol{Q}(\boldsymbol{u}^k) = \big(\boldsymbol{I} + \tau \boldsymbol{A}(\boldsymbol{u}^k)\big)$.

He proves that if the matrix $\boldsymbol{Q}(\boldsymbol{u}^k)$ is continuous in $\boldsymbol{u}^k$, symmetric, nonnegative with a positive diagonal, irreducible, and has unit column sums, several mathematical guarantees can be established: The diffusion process is well-posed and converges towards the average grey value of the input data $\boldsymbol{f}$. Moreover, the process obeys a maximum-minimum principle, i.e. the evolving signal values never exceed the range of the initial signal.

One can show that reasonable discretisations fulfil the requirements of smoothness, symmetry, unit column sums, and irreducibility by design [369]. However, both nonnegativity and positive diagonal entries require a closer look.

In the case of the discretisation (3.51) the central stencil entry lies in the range $\left[-\frac{4}{h^2}, 0\right]$ for any choice of α, β, and bounded diffusion tensor entries $a, b, c \leq 1$. Thus, the matrix $\boldsymbol{Q}(\boldsymbol{u}^k)$ has diagonal entries in $\left[1 - \frac{4\tau}{h^2}, 1\right]$. Enforcing nonnegativity of the lower bound leads to

$$1 - \frac{4\tau}{h^2} \geq 0 \qquad \Leftrightarrow \qquad \tau \leq \frac{h^2}{4}. \tag{3.57}$$

This is a drawback of the explicit scheme: The time step size is drastically limited. Thus, to reach a diffusion time T, a minimum number of steps $k \geq \left\lceil \frac{4T}{h^2} \right\rceil$ is required. This limits the effectiveness of the explicit scheme in practice.

Moreover, nonnegativity is not fulfilled by standard discretisations of anisotropic models, since the diffusion tensor entry b can be come negative [379]. While there exists the so-called *nonnegativity discretisation* [369], it limits the condition number of the diffusion tensor. As the condition number of a matrix is the ratio between its largest and smallest eigenvalues, this constrains the potential anisotropy of the model.

As a consequence, Weickert et al. [379] replace the stability in the maximum norm by stability in the Euclidean norm. A discrete filter is stable in the Euclidean norm, if

$$\|\boldsymbol{u}^{k+1}\|_2 \leq \|\boldsymbol{u}^k\|_2 \qquad \forall k \geq 0\,. \tag{3.58}$$

Due to how $\boldsymbol{u}^{k+1}$ is obtained from $\boldsymbol{u}^k$, this is guaranteed for

$$\rho\Big(\boldsymbol{Q}(\boldsymbol{u}^k)\Big) = \rho\Big(\boldsymbol{I} + \tau \boldsymbol{A}(\boldsymbol{u}^k)\Big) \leq 1 \tag{3.59}$$

where ρ is the spectral radius as defined in (2.25). This means that the mapping from $\boldsymbol{u}^k$ to $\boldsymbol{u}^{k+1}$ is a *contraction mapping*.

If $\boldsymbol{A}(\boldsymbol{u}^k)$ is negative semi-definite, as is the case for the discretisation (3.51), this leads to the time step size restriction

$$\tau \leq \frac{2}{\rho(\boldsymbol{A}(\boldsymbol{u}^k))} \leq \frac{h^2}{2}\,. \tag{3.60}$$

We see that this limit is half as restrictive as (3.57) which guarantees stability in the maximum norm. This, however, is tied to a less strict notion of stability.

In analogy we transfer these results from numerical algorithms for PDEs to neural networks. We found that a maximum-minimum-principle, while stricter, does not allow much freedom when it actually comes to learning the differential operators of a model. A Euclidean stability constraint is much more suitable in this framework. This will be an important motivation for our considerations in Chapter 7.

ACCELERATION STRATEGIES FOR EXPLICIT SCHEMES The severe time step size restrictions of the explicit scheme have produced various acceleration strategies. In the following, we review two methods designed for accelerating discrete diffusion filters: *fast explicit diffusion (FED)* schemes [374], and *fast semi-iterative (FSI)* schemes [170].

FED schemes were first proposed by Grewenig et al. [160]. They rely on the fact that the solution to a linear diffusion process is given by a Gaussian convolution, which in turn can be approximated by a series of convolutions with a box filter. Thus, a box filter of length $2L+1$ approximates the result of a linear diffusion filter with time $T \in \mathcal{O}(L^2)$. In contrast, a series of L explicit steps yields only a diffusion time $T \in \mathcal{O}(L)$ which is linear in the number of steps. Moreover, this box filter can be factorised by L explicit diffusion stencils with varying time step sizes τ_ℓ for $\ell = 0, \dots, L-1$.

This reasoning can be transferred to arbitrary diffusion filters. One simply replaces the fixed time step size τ by a series of time step sizes

$$\tau_\ell = \frac{\tau}{2\cos^2\left(\pi \frac{2\ell+1}{4L+2}\right)} \qquad (\ell = 0, \dots, L-1) \tag{3.61}$$

and performs a *cycle* of L fractional steps

$$\boldsymbol{u}^{k+\frac{\ell+1}{L}} = \left(\boldsymbol{I} + \tau_\ell \boldsymbol{A}(\boldsymbol{u}^k)\right) \boldsymbol{u}^{k+\frac{\ell}{L}} \qquad (\ell = 0, \dots, L-1)\,. \tag{3.62}$$

The *cycle time* is given by the sum of the individual time step sizes in the cycle:

$$\theta_L = \sum_{\ell=0}^{L-1} \tau_\ell = \tau \frac{L^2 + L}{3}\,. \tag{3.63}$$

As a single cycle approximates a box filter, one should employ multiple cycles to obtain a satisfactory approximation quality.

Interestingly, not all time step sizes obey the stability limit $\tau \leq \frac{h^2}{4}$. Still, it is shown that the resulting scheme is stable in the Euclidean norm [374]. This however requires to keep the nonlinearity $\boldsymbol{A}(\boldsymbol{u}^k)$ constant within a cycle. Moreover, due to accumulation of numerical errors for very large time step sizes, it is advised to use a reordering strategy to alternate between small and large step sizes.

A remedy to these issues is given in the form of fast semi-iterative (FSI) solvers. Introduced by Hafner et al. [170], FSI extrapolates the diffusion result at a fractional time step $k + \frac{\ell}{L}$ with the previous fractional time step $k + \frac{\ell-1}{L}$ and a weight α_ℓ. With a similar motivation as for FED, the weights are computed by means of box filter factorisations as

$$\alpha_\ell := \frac{4\ell + 2}{2\ell + 3} \qquad (\ell = 0, \dots, L-1)\,. \tag{3.64}$$

An FSI acceleration with cycle length L reads

$$\begin{aligned} \boldsymbol{u}^{k+\frac{\ell+1}{L}} = \alpha_\ell \left(\boldsymbol{I} + \tau \boldsymbol{A}\left(\boldsymbol{u}^{k+\frac{\ell}{L}}\right)\right) \boldsymbol{u}^{k+\frac{\ell}{L}} \\ + (1 - \alpha_\ell)\, \boldsymbol{u}^{k+\frac{\ell-1}{L}} \qquad (\ell = 0, \dots, L-1)\,. \end{aligned} \tag{3.65}$$

One formally initialises with $\boldsymbol{u}^{k-\frac{1}{L}} := \boldsymbol{u}^k$.

The crucial difference to FED is that FSI uses a time-varying extrapolation instead of time-varying step sizes. This allows to update the nonlinearity $\boldsymbol{A}\left(\boldsymbol{u}^{k+\frac{\ell}{L}}\right)$ at fractional time steps within a cycle. Consequently, FED and FSI yield equivalent results after each cycle [170]. Moreover, FSI schemes are more resilient to numerical errors.

We find that the extrapolation in FSI is a prime example of a larger class of acceleration strategies which we can efficiently translate into a neural network architecture in Chapter 7.

SEMI-IMPLICIT SCHEMES Another way to circumvent the time step size restrictions of explicit schemes is the *semi-implicit scheme*. To this end, one modifies the right-hand side of the explicit scheme to obtain

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} = \boldsymbol{A}(\boldsymbol{u}^k)\, \boldsymbol{u}^{k+1}. \tag{3.66}$$

The only difference to the explicit scheme is the use of $\boldsymbol{u}^{k+1}$ instead of $\boldsymbol{u}^k$ on the right-hand side. This leads to a linear system of equations:

$$\left(\boldsymbol{I} - \tau \boldsymbol{A}(\boldsymbol{u}^k)\right) \boldsymbol{u}^{k+1} = \boldsymbol{u}^k \tag{3.67}$$

If $\boldsymbol{A}(\boldsymbol{u}^k)$ is negative semidefinite, the matrix $\left(\boldsymbol{I} - \tau \boldsymbol{A}(\boldsymbol{u}^k)\right)$ is invertible as its eigenvalues are not smaller than one. This yields the semi-implicit scheme

$$\boldsymbol{u}^{k+1} = \left(\boldsymbol{I} - \tau \boldsymbol{A}(\boldsymbol{u}^k)\right)^{-1} \boldsymbol{u}^k. \tag{3.68}$$

By identifying $\boldsymbol{Q}(\boldsymbol{u}^k) = \left(\boldsymbol{I} - \tau \boldsymbol{A}(\boldsymbol{u}^k)\right)^{-1}$, this scheme also fits the framework of Weickert [369]. The discussions of all requirements except for positive diagonal entries work analogously. However, the semi-implicit scheme always fulfils the requirement of positive diagonal entries [369].

Thus, the semi-implicit scheme does not suffer from a time step size restriction. On the other hand, it requires to solve a large system of equations. This can still be highly efficient as the matrix $\left(\boldsymbol{I} - \tau \boldsymbol{A}(\boldsymbol{u}^k)\right)$ is often sparse. Iterative methods such as the *Jacobi* or *Gauss–Seidel* solvers [320] are popular in practice. We employ semi-implicit schemes together with a conjugate gradient (CG) solver for solving edge-enhancing diffusion inpainting problems in our experimental comparisons.

For nonlinear problems, multiple semi-implicit steps should be employed. As the nonlinearity of $\boldsymbol{A}(\boldsymbol{u}^k)$ is fixed within one step, regular updates are crucial for the correctness of the result. This naturally leads to the idea of *fully implicit schemes*.

FULLY IMPLICIT SCHEMES To avoid having to update the nonlinearity, a fully implicit scheme computes the nonlinearity $\boldsymbol{A}(\boldsymbol{u}^{k+1})$ at

the new time step. This can be interpreted as a backward difference for the temporal discretisation in the time step $k+1$:

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} = \boldsymbol{A}(\boldsymbol{u}^{k+1})\, \boldsymbol{u}^{k+1}. \tag{3.69}$$

This in turn leads to the nonlinear system of equations

$$\left(\boldsymbol{I} - \tau \boldsymbol{A}(\boldsymbol{u}^{k+1})\right) \boldsymbol{u}^{k+1} = \boldsymbol{u}^k. \tag{3.70}$$

Solving this system is cumbersome, which is why in practice these schemes are rarely encountered. We however find that a fixed point iteration for this problem shows architectural similarities to a specific neural network architecture in Chapter 7.

MULTIGRID SOLVERS *Multigrid solvers* [53, 54, 168] are among the most sophisticated numerical algorithms for PDEs. They are designed to accelerate the convergence speed of standard linear solvers by a large margin.

The core idea is that such solvers attenuate high-frequent error components quickly, while low-frequent ones remain for a long time, creating a bottleneck for convergence speed. Multigrid solvers tackle the problem on multiple, differently fine grid levels. Once a reasonable approximation quality on a fine grid is achieved, the problem is sampled to the next coarser level. As the grid size becomes larger, this transforms low-frequent error components into high-frequent ones, allowing to attenuate them efficiently on the coarse grid. Afterwards, the coarse grid solution is brought to the fine grid again and resulting new errors are quickly attenuated, thus yielding an efficient solution strategy.

This *two-grid cycle* can be extended to arbitrarily many grid resolutions. If the grids are visited in a strict fine-to-coarse-to-fine order, one speaks of a *V-cycle*. Interleaving this cycle with additional steps between grids, one obtains a *W-cycle*. The order in which grids are visited is subject to many optimisation strategies, leading to *full multigrid schemes*. A full multigrid scheme can accelerate standard iterative problem solvers by orders of magnitude.

Similar concepts as for multigrid solvers can be observed in specific neural network architectures. This motivates us to investigate the connections between multigrid solvers and neural networks in Chapter 7. Therein, we also provide a review of multigrid solvers with a mathematical description.

This concludes our introduction to diffusion models and their numerical implementation. In the following, we review wavelet shrinkage and variational methods along with their relations to diffusion. This helps to paint a more complete picture of the connections between PDEs and CNNs.

3.2 VARIATIONAL METHODS

An important concept which is closely related to diffusion is that of *variational models* [27, 137]. Based on the *calculus of variations*, these methods have first been used for regularisation of ill-posed problems independently by Whittaker [391] and Tikhonov [354].

The general idea of variational calculus is to obtain a function as a minimiser of an *energy functional*. The intuition is that just like a function can have a minimum at a position, an energy functional can have a minimum at a function. Similarly, like a minimum of a function is described by its first derivative, the minimiser of a functional is characterised by a set of PDEs which are the so-called *Euler–Lagrange equations*.

Designing an energy functional for a task at hand is systematic: In the energy, several measures concerning certain properties of the solution are accumulated. These measures should be large whenever the solution shows undesirable properties, and small for solutions with desirable characteristics. Often times, they model conflicting assumptions, which leads to a trade-off.

While there are many options to design a functional, we mostly deal with energies of the form

$$E(u) = \int_\Omega \left(D(u,f) + \alpha R(u)\right)\, d\boldsymbol{x} \tag{3.71}$$

which map a function $u : \Omega \to \mathbb{R}$ on a domain $\Omega \subset \mathbb{R}^n$ to a real value $E(u) \in \mathbb{R}$.

The goal is to find a function u such that the energy attains a minimum. A *data term* $D(u,f)$ steers the solution u to be close to an input image f, while a *regularisation term* $R(u)$ enforces smoothness conditions on u. Typically, these conditions are modelled by penalising derivative information of u. The balance between the two terms is controlled by a positive *regularisation parameter* $\alpha \in \mathbb{R}^+$.

As an example, we consider the two-dimensional energy functional

$$E(u) = \int_\Omega \left((u-f)^2 + \alpha\, \Psi\left(|\nabla u|^2\right)\right)\, d\boldsymbol{x}, \tag{3.72}$$

where $\Psi : \mathbb{R} \to \mathbb{R}$ is an increasing *regulariser*. In this case we have a quadratic data term. Thus, if the regulariser is strictly convex, the energy is strictly convex such that it has a unique minimiser. As the energy is of the form

$$E(u) = \int_\Omega F\left(x, y, u, u_x, u_y\right)\, d\boldsymbol{x}, \tag{3.73}$$

the Euler–Lagrange equation is given by

$$E_u = F_u - \partial_x F_{u_x} - \partial_y F_{u_y} = 0, \tag{3.74}$$

Here, E_u is the Gâteaux derivative of the Energy E w.r.t. to the function u (see Section 2.1). This involves several Gâteaux derivatives of the integrand in terms of F_u, F_{u_x}, and F_{u_y}.

Moreover, we obtain natural boundary conditions

$$\boldsymbol{n}^\top \begin{pmatrix} F_{u_x} \\ F_{u_y} \end{pmatrix} = 0 \tag{3.75}$$

on the image boundary $\partial\Omega$.

In particular, for the energy (3.72) we have

$$F_u = 2\,(u - f)\,, \tag{3.76}$$

$$F_{u_x} = 2\alpha\,\Psi'\left(|\nabla u|^2\right) u_x, \tag{3.77}$$

$$F_{u_y} = 2\alpha\,\Psi'\left(|\nabla u|^2\right) u_y. \tag{3.78}$$

This yields an Euler–Lagrange equation

$$u - f - \alpha\,\nabla^\top \left(\,\Psi'\left(|\nabla u|^2\right) \nabla u\right) = 0 \tag{3.79}$$

with boundary conditions

$$\boldsymbol{n}^\top \begin{pmatrix} 2\alpha\,\Psi'\left(|\nabla u|^2\right) u_x \\ 2\alpha\,\Psi'\left(|\nabla u|^2\right) u_y \end{pmatrix} = 0 \quad \Leftrightarrow \quad \boldsymbol{n}^\top \nabla u = 0. \tag{3.80}$$

In the Euler–Lagrange equation, we have summarised the terms $\partial_x F_{u_x}$ and $\partial_y F_{u_y}$ by means of a divergence term and divided the equation by a factor two. The boundary conditions can be reduced to $\boldsymbol{n}^\top \nabla u$ since Ψ is increasing, and thus Ψ' is nonnegative.

This is just one example of an Euler–Lagrange equation. Using more than one variable within the argument of the energy produces multiple Euler–Lagrange equations. Moreover, higher-dimensional functions yield more terms within the equations. For a derivation of Euler–Lagrange equations in the general case, we refer to the monographs [27, 137].

In practice, a minimiser can be found in two different ways: One option is discretising the energy directly and finding a minimum by means of optimisation strategies. Another option is to discretise the Euler–Lagrange equation and solving either the elliptic PDE by solving the arising system of equations, or embedding the equation into a parabolic PDE by the *method of artificial time* and solving it by means of an iterative scheme.

Variational models have been used in countless applications, among which are optical flow [26, 58, 189], denoising [70, 313], inpainting [21, 31], and segmentation [261, 265].

For us, they are particularly interesting due to their connection to diffusion models. Scherzer and Weickert [326] show that Euler–Lagrange

equations of variational regularisation models approximate diffusion processes. For example, a reformulation of the Euler–Lagrange equation (3.79) reads

$$\frac{u - f}{\alpha} = \nabla^\top \left(\Psi' \left(|\nabla u|^2 \right) \nabla u \right), \tag{3.81}$$

which can be interpreted as a fully implicit time discretisation for a nonlinear diffusion process with stopping time $T = \alpha$ and diffusivity $g(s^2) = \Psi'(s^2)$. This plays an important role when considering the connections between activation functions, diffusivities, wavelet shrinkage functions, and variational regularisers in Chapter 6.

Moreover, the core ideas of variational modelling are also transferred to the design of CNN loss functions. More and more, instead of a pure data-driven optimisation, one encounters hybrid models which include prior information in the form of regularisers. We discuss the existing connections between variational methods and CNNs in Section 3.5.

3.3 WAVELET SHRINKAGE

In the following, we review the concept of wavelet shrinkage [108] which is the third and final traditional image processing approach which we will connect to neural networks. Moreover, Chapter 4 of this thesis is devoted to a trainable wavelet shrinkage approach.

ONE-DIMENSIONAL WAVELET SHRINKAGE Wavelet shrinkage represents a noisy signal $\boldsymbol{f} = \boldsymbol{v} + \boldsymbol{n}$ in a different basis, wherein the additive noise $\boldsymbol{n}$ is easier to separate from the true signal $\boldsymbol{v}$ than in the spatial domain. This is achieved by representing $\boldsymbol{f}$ in terms of different frequency components. Thus, the *wavelet transform* is a frequency-based transform. In contrast to the unlocalised Fourier and discrete cosine transforms, the wavelet transform retains local frequency information. Scaled and shifted versions of a lowpass function Φ and a bandpass function Ψ form the *wavelet basis* which represents the signal in terms of localised frequency components.

For simplicity, assume that the input signal $\boldsymbol{f} \in \mathbb{R}^{2^n}$ has length 2^n. The forward wavelet transform for this signal can be expressed as

$$\boldsymbol{f} = \langle \boldsymbol{f}, \boldsymbol{\Phi}_{n,0} \rangle \, \boldsymbol{\Phi}_{n,0} + \sum_{j=1}^{n} \sum_{k=0}^{2^{n-j}-1} \langle \boldsymbol{f}, \boldsymbol{\Psi}_{j,k} \rangle \, \boldsymbol{\Psi}_{j,k}, \tag{3.82}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

The discrete vectors $\boldsymbol{\Phi}_{j,k}$ and $\boldsymbol{\Psi}_{j,k}$ are obtained by sampling the continuous functions

$$\Psi_{j,k}(x) = 2^{-\frac{j}{2}} \, \Psi\left(2^{-j}x - k\right), \tag{3.83}$$

$$\Phi_{j,k}(x) = 2^{-\frac{j}{2}} \, \Phi\left(2^{-j}x - k\right). \tag{3.84}$$

The function $\Psi(x)$ is called the *mother wavelet*, as all other wavelets can be derived from it. The function $\Phi(x)$ is denoted as the *scaling function*. In the transform (3.82), it is only used to encode the average grey value of $\boldsymbol{f}$, however its scaled and shifted versions can be used to efficiently compute the *fast wavelet transform*.

The coefficients of the transformed signal are given by the inner products between the signal and the basis vectors

$$d_{j,k} = \langle \boldsymbol{f}, \boldsymbol{\Psi}_{j,k} \rangle \,, \tag{3.85}$$

$$c_{n,0} = \langle \boldsymbol{f}, \boldsymbol{\Phi}_{n,0} \rangle \,. \tag{3.86}$$

The values $d_{j,k}$ are the *wavelet coefficients*, and $c_{n,0}$ is a *scaling coefficient* which encodes the average grey value of the signal.

In the wavelet representation, the noise affects all wavelet coefficients $d_{j,k}$ while the signal is represented by only a few significant ones [248]. By modifying the coefficients in an appropriate way, a reconstructed signal $\boldsymbol{u}$ can be obtained with the following three-step framework:

1. *Analysis:* The input data $\boldsymbol{f}$ is transformed into wavelet and a scaling coefficient according to (3.85).
2. *Shrinkage:* A shrinkage function S_θ with a threshold parameter θ is applied to the wavelet coefficients $d_{j,k}$. The scaling coefficient $c_{n,0}$ is unaltered as to not change the average grey value of $\boldsymbol{f}$.
3. *Synthesis:* The denoised version $\boldsymbol{u}$ of $\boldsymbol{f}$ is reconstructed from the modified wavelet coefficients by means of the backward wavelet transformation:

$$\boldsymbol{u} = \langle \boldsymbol{f}, \boldsymbol{\Phi}_{n,0} \rangle \, \boldsymbol{\Phi}_{n,0} + \sum_{j=1}^{n} \sum_{k=0}^{2^{n-j}-1} S_\theta\big(\langle \boldsymbol{f}, \boldsymbol{\Psi}_{j,k} \rangle\big) \, \boldsymbol{\Psi}_{j,k} \,. \tag{3.87}$$

The selection of the wavelet basis is an important modelling aspect. While there are many choices available [94, 248], we focus on the Haar wavelet basis [165] in this thesis. It defines the wavelet and scaling function as

$$\Psi(x) = \begin{cases} 1, & 0 \le x \le \frac{1}{2}, \\ -1, & \frac{1}{2} < x \le 1, \\ 0, & \text{else}, \end{cases} \tag{3.88}$$

$$\Phi(x) = \begin{cases} 1, & 0 \le x \le 1, \\ 0, & \text{else}. \end{cases} \tag{3.89}$$

Figure 3.3 visualises these functions. It shows that a discretised version of the mother wavelet Ψ represents a finite difference operation corresponding to a first order derivative. The scaling function Φ implements a simple averaging.

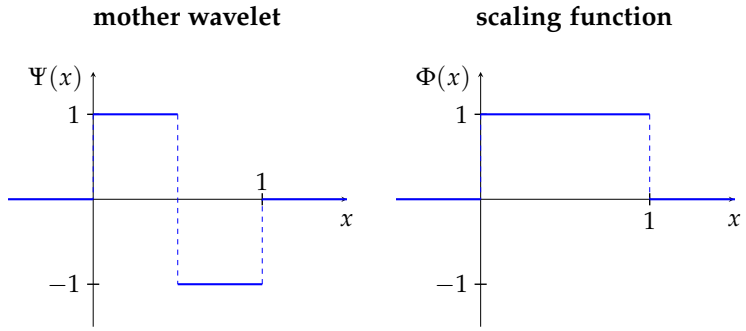


Figure 3.3: Mother wavelet and scaling function. The mother wavelet implements a derivative operation, while the scaling function constitutes an averaging.

Moreover, the choice of the shrinkage function is important for the behaviour of the process. For example, the soft shrinkage function [107] eliminates all coefficients which are smaller than a threshold θ, and shrinks all others by θ:

$$S_\theta(x) = \begin{cases} 0, & |x| \leq \theta, \\ x - \theta \operatorname{sgn}(x), & |x| > \theta. \end{cases} \tag{3.90}$$

While typical shrinkage functions are easy to apply in a practical setting, they suffer from the fact that they use the same threshold parameter for all scales and the binary decision structure. There is no clear boundary between noise and signal coefficients such that eliminating too many coefficients always destroys signal details and eliminating too few coefficients leaves too much noise in the reconstruction. Furthermore, the first scale might require a significantly different threshold for this decision as it represents a finer scale where noise is much more noticeable. These downsides serve as a motivation for our trainable wavelet shrinkage model in Chapter 4.

TRANSLATION INVARIANT WAVELET SHRINKAGE. The wavelet transformation (3.82) is not translation invariant: Shifting the input signal $\boldsymbol{f}$ produces a different set of wavelet coefficients. To overcome this problem, Coifman and Donoho propose *cycle spinning* [85]. The input signal is shifted, wavelet shrinkage is applied, and the results are averaged for all possible shifts.

This strategy yields a shift-invariant transformation and additionally removes typical block artefacts arising due to the discrete shifts of the wavelets. A downside which is often overlooked is that this kind of transformation destroys the assumption of independence between wavelet coefficients, which is the assumption under which the classical shrinkage functions have been designed [175].

As the number of possible shifts is the same as the signal size, computing the translation-invariant result in a naive way is compu-

tationally expensive. However, an efficient implementation can be achieved using the *algorithme à trous* of Holschneider et al. [187]. A more detailed discussion of this algorithm can be found in Chapter 4.

We require the translation invariant wavelet transform to benefit from the following connections between wavelet shrinkage and nonlinear diffusion.

RELATION TO NONLINEAR DIFFUSION An interesting connection between wavelet shrinkage and nonlinear diffusion has been found by Mrázek et al. [264].

They show that the application of Haar wavelet shrinkage on the finest level of the shift-invariant wavelet decomposition is equivalent to one step of nonlinear diffusion with time step size τ and diffusivity g, if shrinkage function and diffusivity obey

$$S(x) = x \left(1 - 4\tau\, g\left(|\sqrt{2}x|\right)\right). \tag{3.91}$$

The connection allows us to use any established diffusivity function from the diffusion literature and translate it into a suitable shrinkage function. As we know the behaviour of the diffusivity, we can infer the behaviour of the shrinkage function at least on the finest scale of the wavelet transformation. On all coarser levels, the shrinkage function essentially performs nonlinear diffusion on the respective level of the Gaussian pyramid [6] of the input image. This is an important foundation of our trainable wavelet model in Chapter 4. Moreover, it helps us to connect wavelet shrinkage to residual networks in Chapter 6.

3.4 DEEP LEARNING AND NEURAL NETWORKS

Since more than a decade, *deep learning* [152, 220, 223, 225, 330] has fundamentally changed signal and image processing. This has been the result of the increasing availability of data, advances in parallel computing hardware, and progress in optimisation methods. The *data-driven* nature of neural networks makes them highly flexible such that they can be adapted to a plethora of tasks.

Before the advent of deep learning, approaches were largely *model-driven*. While machine learning techniques were already successful, they relied on hand-crafted representations of the data to which a machine learning algorithm could be applied. The core success of neural networks is their ability to learn these *features* from the data at hand together with the appropriate mapping to the desired outputs [152].

TERMINOLOGY Before we dive in to the field of deep learning and neural networks, we want to specify a more rigorous terminology. To this end, we follow the hierarchy of Goodfellow et al. [152]: The overarching term is that of *artificial intelligence (AI)*. The goal of AI is to

create machines which can solve tasks based on an understanding of the problem at hand. However, an AI system can rely on hard-coded rule sets and decision strategies, while not necessarily being able to obtain knowledge from data.

Giving machines the ability to extract this knowledge is the goal of *machine learning*. This requires information which is shaped in such a way that simple algorithms can base their decisions on it. The shape of the data is denoted as a *representation*. The representation of the data at hand is crucial for the success of machine learning models. When the semantic distance between the data and a suitable representation for decisions is large, learning simple representations alone is not appropriate. For example, finding a direct mapping between the set of pixels of an image and a semantic understanding of what is displayed is infeasible.

This is where *deep learning* models that rely on *neural networks* excel. The notion of 'deep' is not rigorously defined: One option is to call networks 'deep' whenever they are able to learn multiple representations of the data with growing complexity. For example, Zeiler and Fergus [406] have shown that an image classifier represents the image data first by simple descriptors such as e.g. edges. With growing depth, descriptors are combined into more complex ones, before the resulting high-level representations can be used e.g. for classification.

The topics of this thesis range from machine learning to deep learning. In particular, Part I makes use of machine learning frameworks to improve wavelet- and diffusion-based models. Moreover, we consider fundamental concepts of diffusion models and their numerical solution strategies and connect them to neural network architectures in Part II. Lastly, Part III presents one instance of full deep learning models which are used in conjunction with diffusion PDEs.

3.4.1 *From Single Neurons to Convolutional Neural Networks*

In the following, we introduce *convolutional neural networks* (CNNs) inductively by starting with their simplest component: a *neuron*.

A neuron takes a series of input components $\boldsymbol{x} \in \mathbb{R}^n$ and computes a single output as a function $f : \mathbb{R}^n \to \mathbb{R}$. To this end, it uses weights $\boldsymbol{w} \in \mathbb{R}^n$ for every input, a *bias* $b \in \mathbb{R}$, and an *activation function* $\varphi : \mathbb{R} \to \mathbb{R}$ as follows:

$$f(\boldsymbol{x}) = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right). \tag{3.92}$$

The argument of the activation function is a linear combination of the input components and their corresponding weights. In addition, this combination can be shifted by a bias. The activation function then decides whether the neuron is *excited* or not. While traditional neural

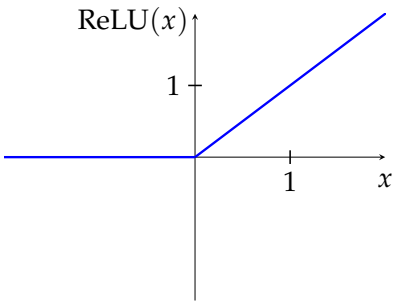


Figure 3.4: Visualisation of the rectified linear unit (ReLU) activation. Only positive arguments are activated.

modelling [253, 307] considered only binary decisions, i.e. either the neuron outputs one if it is excited and zero otherwise, modern neural networks consider a wide range of activation functions.

In particular, the *rectified linear unit (ReLU)* [149, 266] is the most popular activation function to date. It is a linear function which is truncated at zero (see Figure 3.4):

$$\mathrm{ReLU}(x) = \begin{cases} x, & x \geq 0, \\ 0, & x < 0. \end{cases} \tag{3.93}$$

While the weights decide the linear combination of the various input components for the output of a neuron, the activation function introduces nonlinearity. This is a crucial aspect for the success of deep neural networks.

FULLY CONNECTED NEURAL NETWORKS The usefulness of a single neuron is obviously limited. Their power only unfolds when combining multiple neurons within networks. To this end, one arranges neurons in two directions: parallel and sequential. A parallel ensemble of neurons is called a *layer*, and multiple layers in sequence form a *neural network*. The first layer of a network is called the *input layer*, and the last one is denoted as the *output layer*. Layers in between are called *hidden layers*.

Connecting the outputs of neurons to their successors is key to combining the relatively simple outputs of single neurons into powerful compositions of functions. When every neuron in a layer is connected to all neurons of the subsequent layer, the network is *fully connected*. An example of a fully connected network with three layers is presented in Figure 3.5. Each of the four neurons within the hidden layer has its own set of weights. Similarly, the two neurons of the output layer combine the results of all hidden neurons in another weighted combination with a final activation. This composition is already complex in this simple example, which shows why networks are often described by means of graphical representations instead of rigorous mathematical expressions.

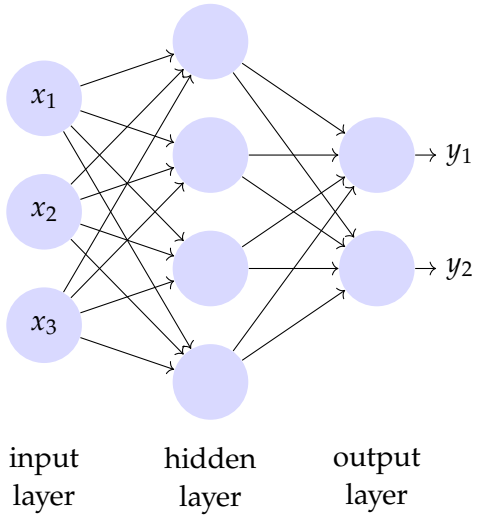


Figure 3.5: A fully connected network with three inputs, two outputs, and a single hidden layer with four neurons. All outputs of a layer are connected to all neurons of the subsequent one. This network contains 26 weight parameters.

The exemplary network contains 26 weights. Each hidden neuron requires a weight for each of the three inputs, and a bias. In addition, the output neurons combine four inputs with a bias for each output. This shows that for fully connected networks, the number of weights grows dramatically. This has motivated several strategies for reducing the number of weights.

CONVOLUTIONAL NEURAL NETWORKS The most popular solution is the use of convolutions [132, 225]. In a *convolutional neural network (CNN)*, each neuron within a layer performs the same operation on a sparse set of neighbouring inputs by means of a discrete convolution (see Section 2.2). The weights of a neuron form a kernel $\boldsymbol{w} \in \mathbb{R}^n$ such that a neuron at position j within a layer computes its output as

$$f_j(\boldsymbol{x}) = \varphi\Big((\boldsymbol{w} * \boldsymbol{x})_j + b\Big)\,. \tag{3.94}$$

Typically, the kernel $\boldsymbol{w}$ is sparse: It contains only a small set of non-zero entries. Moreover, a kernel is shared between all neurons within a layer. This strategy assumes a spatial correlation of the input components. While switching two inputs in a fully connected network can be compensated by exchanging the corresponding weights, this is not possible in a convolutional network. However, meaningful features in images can already be computed on very small neighbourhood sizes.

Note that the convolution operation in CNNs is predominantly implemented as a cross-correlation. The only difference between a cross-correlation operation and the discrete convolution defined in (2.8) is that the kernel is not mirrored before the product of the two functions is computed. In practical settings with learned parameters, this tech-

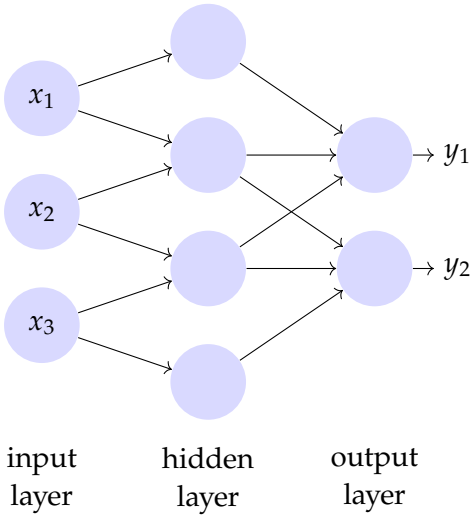


Figure 3.6: A convolutional network with three inputs, two outputs, and a single hidden layer with four neurons. Each layer takes only a sparse set of inputs to compute its output. This network contains only 7 weight parameters.

nical nuance is compensated by the adaptivity of the model. Throughout the thesis, we use the convolution notion as we have introduced it in Section 2.2.

The introduction of a convolution is beneficial in several ways. First, the operations can be computed more efficiently as the number of operations shrinks drastically: A fully connected layer computes the multiplication $\boldsymbol{W}\boldsymbol{x}$ with a dense matrix $\boldsymbol{W} \in \mathbb{R}^{n \times m}$ for n neurons with m inputs. A convolutional layer computes $\boldsymbol{w} * \boldsymbol{x}$ once, which can be realised efficiently by a sparse matrix multiplication. As this reduces the number of effective inputs to a neuron, convolutional layers are much more efficient.

Moreover, using the same weights in each neuron of a layer severely reduces the amount of parameters. Figure 3.6 presents a convolutional counterpart of the fully connected network from Figure 3.5. Therein, each neuron in the hidden layer considers only the two closest inputs, with the same weights for each hidden neuron. The output neurons only take the three closest hidden neuron results as inputs with the same weights for each output. With the same number of neurons, this network only requires 7 parameters. The hidden layer contains two weights and a bias, and the output layer requires three weights and a bias.

Another benefit is that CNNs are invariant to translations of the input by design. This *shift invariance* implies that a translation of the input signal implies the same translation in the output signal, and constitutes one fundamental invariance which is desirable for image processing applications. This is not the case for rotation invariance as opposed to many PDE models, an issue which motivates our contribu-

tions of Chapter 8 where we translate rotation invariance guarantees from PDE models to specific neural networks.

POOLING OPERATIONS Another important concept for CNNs is *pooling* [51, 388]. A pooling operation creates a summary of statistics on a small local neighbourhood. For example, a *max-pooling* on a 2×2 neighbourhood of an image produces a lower resolution image by replacing pixels in the neighbourhood by their maximum. An *average pooling* replace the pixels by their average value.

Pooling serves multiple purposes. The dimensionality reduction plays an important role when moving from local statistics such as edges towards feature representations of higher complexity such as eyes. This is intuitive, as the location of these features cannot be pinpointed to a pixel neighbourhood of the original image, but rather a coarse localisation. Thus, shifting the image by one pixel does not change the result of a 2×2 pooling operation. Consequently, the classifier part of an image classification network always obtains the same feature representation, regardless of the dimension of the input image.

The use of pooling also improves the efficiency of networks. Computations within a network which never reduces the original image size are cumbersome both for the forward pass, as well as the backward pass required for training (see Section 3.4.3). The dimensional reduction alleviates both memory and computational burden.

CHANNELS AND TENSORS The simple CNNs which we described so far take generic input data in the form of vectors. This is not the usual case in practice. To this end, we introduce the notion of *network channels* and the convenient concept of *tensors*.

When a convolutional layer is applied within a network for image data, it typically does not map a single image to another representation of it. Instead it takes multiple versions of the image and maps it to multiple modifications thereof. The different input and output versions are interpreted as a multi-channel image. For each output channel, the layer has a different set of weights to combine all input channels. For example, if a layer maps from four to eight channels with a kernel size of 3×3, it uses $4 \cdot 8 \cdot (3 \cdot 3) = 288$ parameters. To this end, one denotes each layer by its kernel size and its output dimension. In the above example, the layer is a $3 \times 3 \times 8$ convolutional layer.

During training, networks perform their inference on batches of images (see Section 3.4.3). The additional batch dimension and the notion of channels quickly inflate the representations within the network. Thus, for convenience one switches to the notion of tensors. A practical way to think of tensors is as arrays with multiple axes. For example, a batch of 16 images of size 256×256 with eight channels is a tensor of dimension $16 \times 256 \times 256 \times 8$. This convention makes it

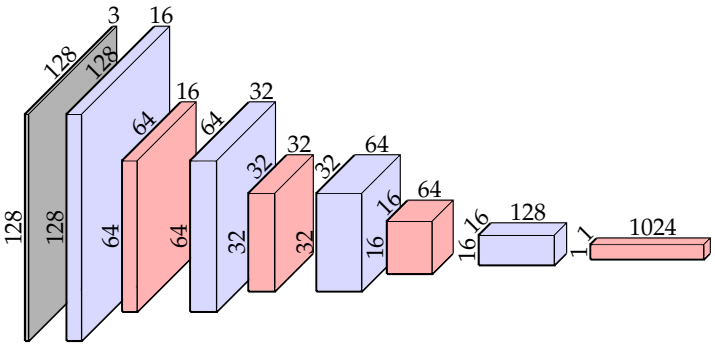


Figure 3.7: Visualisation of a two-dimensional convolutional neural network reducing an input colour image of dimensions $128 \times 128 \times 3$ to a vector of 1024 features. Width and height of the feature representations are denoted by top left and left aligned numbers, and the channel dimension is written above the cuboids. Convolutional layers are coloured in blue, pooling layers are coloured in red, and the input layer is coloured in grey.

easy to quickly grasp the structure of networks. In theory we could apply the same vectorisation approach as for images (see Section 2.5), however the tensor notation is often more convenient.

Due to the sheer complexity of modern networks, representing them in mathematical terms is cumbersome. In contrast, graphical representations are more intuitive. A typical approach for image-based CNNs is to represent layers in terms of their output dimensionality by means of a cuboid. Their width and height denote the image resolution, and their depth shows the number of channels. Often, one explicitly adds the tensor dimensions except for the batch size to the layer. Activation functions, on the other hand, are often implicit. Figure 3.7 presents an example of a network which takes colour images with input dimension $128 \times 128 \times 3$ and transforms them through eight layers into a feature vector of size $1 \times 1 \times 1024$.

3.4.2 *Network Types and Popular Architectures*

In the following, we briefly review the fundamental design choices within neural networks. Moreover, we introduce the two most central network architectures for this thesis.

FEED-FORWARD VERSUS RECURRENT NEURAL NETWORKS One design aspect of a network is whether it contains loops within its information flow. If layers are only connected to subsequent ones, the network is a *feed-forward network*. For example, the network from the graphical example in Figure 3.7 is such a network.

If layers are connected to themselves or to earlier layers, one obtains a *recurrent neural network (RNN)* [142, 182, 188, 316]. RNNs are suitable

for processing sequential data in tasks such as speech recognition [322] and machine translation [396].

The two types of architectures are not completely unrelated. By means of *unrolling* [75, 217, 260, 345], one can transform the loops into a sequence of feed-forward layers with sequential inputs and potentially shared parameters.

In this thesis, we encounter both network structures. In particular, Part II connects both variants to numerical algorithms for PDEs. Moreover, the model proposed in Chapter 5 can be interpreted as an unrolled network, and in Chapter 10 we make use of a feed-forward network.

CLASSIFICATION VERSUS IMAGE-TO-IMAGE NETWORKS Another important aspect is the output dimension of the network. This is dependent on the application. A *classification network* (see e.g. [220, 348, 406]) reduces the input images to a vector of features, just like the network example in Figure 3.7. As the features themselves are no classification on their own, a fully connected layer takes the feature vector as an input and produces a vector of scores for classification. Each score denotes the confidence of the network for an image belonging to a certain class, where the classes are defined a priori.

Classification is one example of a task where the input data is transformed to a different output dimension. In contrast, a range of tasks require to obtain an output with a similar dimensionality as the original image. These include segmentation [78], medical image analysis [106, 306], and optical flow [198]. Corresponding networks are fundamentally different in their architecture: While they also create high-dimensional intermediate representations of the input data, their output is transformed back to the original input dimensionality or e.g. a downsampled version thereof.

RESIDUAL NETWORKS The most important network architecture for this thesis, and arguably the most popular one to date is the *residual network (ResNet)* [173]. Its main contribution is the introduction of so-called *skip connections* which facilitate training of very deep networks.

A residual network consists of residual blocks. A single block computes an output signal $\boldsymbol{u}$ from an input $\boldsymbol{f}$ as

$$\boldsymbol{u} = \varphi_2(\boldsymbol{f} + \boldsymbol{W}_2\, \varphi_1(\boldsymbol{W}_1 \boldsymbol{f} + \boldsymbol{b}_1) + \boldsymbol{b}_2)\,. \tag{3.95}$$

One first applies an inner convolution parametrised by weights $\boldsymbol{W}_1$ and a bias vector $\boldsymbol{b}_1$ to the input signal and passes the result into an inner activation function φ_1.

Afterwards an outer convolution $\boldsymbol{W}_2$ with a bias vector $\boldsymbol{b}_2$ is applied to the output of the activation. One adds its result to the original signal $\boldsymbol{f}$, and finally applies an outer activation function φ_2 to obtain the output signal $\boldsymbol{u}$.

Adding the original signal before the final activation forms a skip connection, which is the crucial novelty of ResNets over standard feed-forward networks. Intuitively, the original signal information 'skips' the inner activation and both convolutions. It is the key to training deep networks with large amounts of layers efficiently and without suffering from the vanishing gradient problem. This phenomenon appears when backpropagation gradients approach zero for very deep networks, bringing the training process to a halt [42, 181, 347]; see also Section 3.4.3.

From the perspective of differential equations, ResNets are particularly attractive due to their skip connection. This allows to connect them to explicit schemes [10, 11, 15, 167, 319, 412] which in turn yields provable stability guarantees. This plays the central role in Part II of this thesis.

Moreover, a continuous-time extension of ResNets called *neural ordinary equations (NODEs)* [73] sparked great interest. Instead of providing the layers explicitly, the output is given by a black-box *ordinary differential equation (ODE)* solver of a parametrised dynamic. Instead of backpropagating through the solver, the parameter updates are computed by an adjoint sensitivity analysis. Even though this idea has had great success, recent research suggests that NODEs suffer from a strong interdependence between model and solver [164, 274], and arguments from weight scaling imply that they might in fact not be the continuous-time extension of ResNets [82]. Our philosophy is different from that of NODEs. We argue that a neural architecture realises capabilities of a numerical solver, and that training this architecture corresponds to finding a suitable model.

U-NETS The second popular network architecture which we deal with is the U-net which was introduced by Ronneberger et al. [306]. As the name suggests, information passes through the network in a U-shape. In the downward pass, the image dimensions are reduced on each level of the network and the channel number is increased. After reaching a coarsest level, the acquired information is subsequently upsampled and combined with information on the next finer level. This allows to efficiently create multiscale representations of the data, which has proven useful throughout various tasks such as segmentation [106, 306], pose estimation [268], inpainting [9, 91], and many more.

Figure 3.8 visualises a U-net with three levels and two convolutional layers on each level. After bringing the input 128×128 colour image to a feature representation with 16 channels, a pooling layer reduces the image size by a factor of two in each direction. Popular choices are max- and average pooling, however pooling operators can also be learned [202, 209].

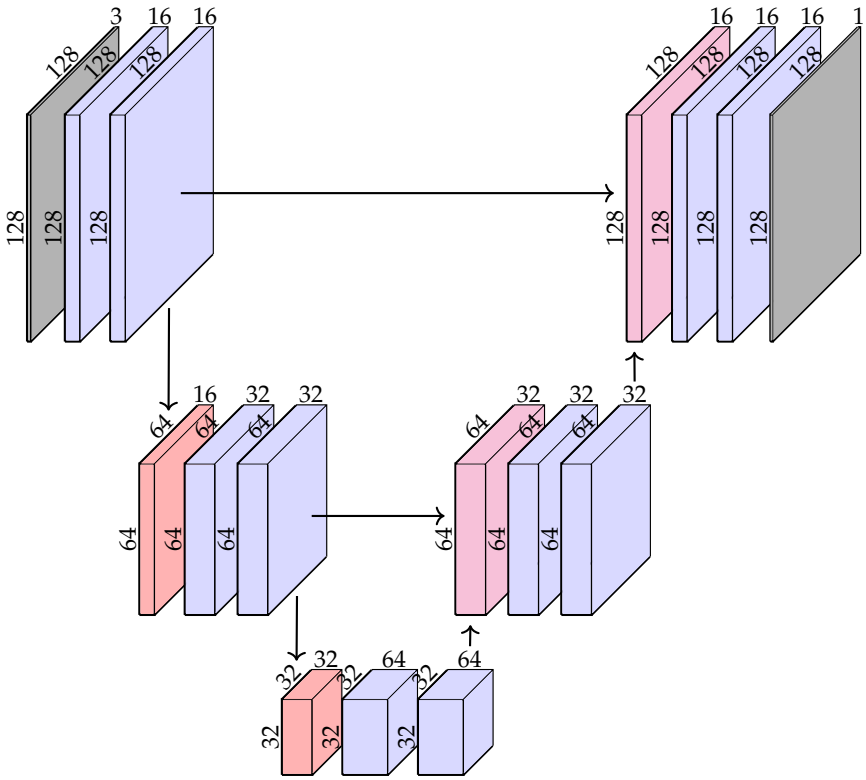


Figure 3.8: Visualisation of a U-net with three levels and two convolutional layers per level. On each level, the image dimensions are halved and the number of channels is doubled. The input and output layers are coloured in grey, convolutional layers in blue. Red layers are pooling layers, and magenta layers combine an upsampled version of the lower level output with the information on the same level.

This strategy is continued until the finest level. After the convolutional layers on that level, information is upsampled by means of interpolation. A simple nearest neighbour interpolation is often sufficient, however also for the upsampling operator, trainable choices are available. We implicitly denote the upsampling within the convolutional layer on the higher level. Moreover, the upsampling reduces the channel dimension again.

The layer additionally receives information which was acquired on the downsampling pass from the same level. While the original U-net [306] combines this information and the upsampled one by concatenating the channels, other works such as [268] use addition. For our theoretical considerations in Part II, we focus on the model which uses addition. In the practical setting of Chapter 10, we use the original variant.

Information is successively upsampled, combined, and convolved until reaching the finest layer again. Then the last convolutional layer reduces the information to the desired output dimension. In this case, we chose a 128×128 grey value image. For example, this result could describe a foreground-background segmentation map.

In contrast to the example of classification networks, features become more complex and abstract on lower levels, while on higher levels information is organised in a similar way as the input. This allows to produce outputs which correlate with the original image, such as segmentations thereof.

We benefit both theoretically and practically from the U-net. In Chapter 7 we show that U-nets realise a sophisticated multigrid strategy, and in Chapter 10 we use it to generate sparse masks for diffusion-based inpainting.

3.4.3 *Training a Neural Network*

The most complex network architectures are of no use without training them. In the following, we explain the basic concept behind a training process, and review popular optimisation methods for realising the training.

LEARNING BY BACKPROPAGATION The central concept for training a neural network is *backpropagation*, which has been popularised by Rumelhart et al. [316]. It is a strategy to compute the way in which parameters have to be updated to better fit the optimisation objective. Starting with the objective, one sequentially passes backwards through the layers, accumulating partial derivatives along the way.

Assume that the neural network is a parametrised function $\mathcal{N}_\theta$ which maps an input $\boldsymbol{f}$ to an output $\boldsymbol{u}^K$ after K layers, based on some parameter set $\boldsymbol{\theta} = \left(\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^{K-1}\right)$. Moreover, let an intermediate

output $\boldsymbol{u}^{k+1}(\boldsymbol{u}^k, \boldsymbol{\theta}^k)$ only depend on the parameters $\boldsymbol{\theta}^k$ of the layer k and the previous output $\boldsymbol{u}^k$, with $\boldsymbol{u}^0 = \boldsymbol{f}$.

One seeks to optimise an objective, a so-called *loss function* $\mathcal{L}(\boldsymbol{u}^K, \boldsymbol{v})$, which maps pairs of prediction $\boldsymbol{u}^K$ and ground truth $\boldsymbol{v}$ to a loss value. The loss is supposed to minimal when $\boldsymbol{u}^K = \boldsymbol{v}$, and grows with larger distance between the two. To adapt the parameters in such a way that the loss is steered towards its minimum, we need to move them opposite to the gradient of the loss w.r.t. the parameter vector $\boldsymbol{\theta}$. Thus, the parameters $\boldsymbol{\theta}^k$ of the k-th layer should be updated by means of the gradient

$$\frac{\partial \mathcal{L}(\boldsymbol{u}^K, \boldsymbol{v})}{\partial \boldsymbol{\theta}^k} = \frac{\partial \boldsymbol{u}^K}{\partial \boldsymbol{\theta}^k} \frac{\partial \mathcal{L}(\boldsymbol{u}^K, \boldsymbol{v})}{\partial \boldsymbol{u}^K} \,. \tag{3.96}$$

Here, we used the chain rule to split the gradient into two derivatives. Note that the derivative $\partial \mathcal{L}(\boldsymbol{u}^K, \boldsymbol{v}) / \partial \boldsymbol{u}^K$ of the scalar loss w.r.t. the prediction vector is a column vector, as we use *denominator-layout notation*. This helps to better visualise the steps of the backpropagation. The derivative $\partial \boldsymbol{u}^K / \partial \boldsymbol{\theta}^k$ of the prediction vector w.r.t. the parameter vector of layer k is a matrix, where position i, j contains $\partial u_j^K / \partial \boldsymbol{\theta}_i^k$.

Further expanding the first term with the chain rule, we obtain

$$\frac{\partial \mathcal{L}(\boldsymbol{u}^K, \boldsymbol{v})}{\partial \boldsymbol{\theta}^k} = \frac{\partial \boldsymbol{u}^{k+1}}{\partial \boldsymbol{\theta}^k} \frac{\partial \boldsymbol{u}^{k+2}}{\partial \boldsymbol{u}^{k+1}} \cdots \frac{\partial \boldsymbol{u}^K}{\partial \boldsymbol{u}^{K-1}} \frac{\partial \mathcal{L}(\boldsymbol{u}^K, \boldsymbol{v})}{\partial \boldsymbol{u}^K} \,. \tag{3.97}$$

The denominator notation elegantly shows the backpropagation effect: Starting with the loss derivative vector, one subsequently multiplies it with layer derivatives. After each step, one obtains the update for the respective parameter set.

The specific derivative shapes depend on the layer structure. For convolutional layers

$$\boldsymbol{u}^{k+1} = \varphi^k\Big(\boldsymbol{W}^k \boldsymbol{u}^k + \boldsymbol{b}^k\Big) \tag{3.98}$$

with individually parametrised multi-channel convolutions $\boldsymbol{W}^k$, biases $\boldsymbol{b}^k$, and activations φ^k, we have

$$\frac{\partial \boldsymbol{u}^{k+1}}{\partial \boldsymbol{u}^k} = \Big(\boldsymbol{W}^k\Big)^\top \operatorname{diag}\bigg(\Big(\varphi^k\Big)'\Big(\boldsymbol{W}^k \boldsymbol{u}^k + \boldsymbol{b}^k\Big)\bigg) \,, \tag{3.99}$$

$$\frac{\partial \boldsymbol{u}^{k+1}}{\partial w_{ij}^k} = u_j^k \, \boldsymbol{e}_i \operatorname{diag}\bigg(\Big(\varphi^k\Big)'\Big(\boldsymbol{W}^k \boldsymbol{u}^k + \boldsymbol{b}^k\Big)\bigg) \,, \tag{3.100}$$

$$\frac{\partial \boldsymbol{u}^{k+1}}{\partial \boldsymbol{b}^k} = \operatorname{diag}\bigg(\Big(\varphi^k\Big)'\Big(\boldsymbol{W}^k \boldsymbol{u}^k + \boldsymbol{b}^k\Big)\bigg) \,. \tag{3.101}$$

The derivative w.r.t. the weight matrix $\boldsymbol{W}^k$ is given in terms of the individual components $w_{i,j}$ for notational convenience. The vector $\boldsymbol{e}_i$ is the i-th basis vector.

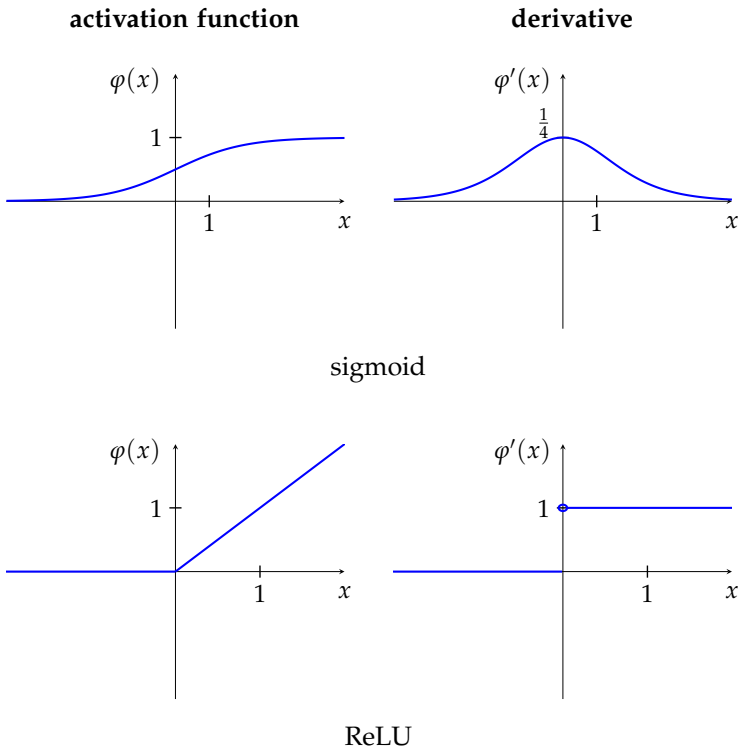


Figure 3.9: ReLU and sigmoid activations with their derivatives. The sigmoid function suffers from vanishing gradients for large absolute values of the input. The ReLU function does not, but is nondifferentiable at zero.

VANISHING GRADIENT PHENOMENON We observe that in the above derivations, the derivative of the activation function $\left(\varphi^k\right)'$ is involved. This is a major cause of the *vanishing gradient phenomenon* [42, 181, 347], where derivatives become too small during backpropagation such that the training process comes to a halt. For example, the sigmoid activation

$$\varphi(x) = \frac{1}{1+e^{-x}} \tag{3.102}$$

approaches zero and one in the limit for $x \to -\infty$ and $x \to \infty$, respectively. Thus, its derivative

$$\varphi'(x) = \frac{e^{-x}}{(1+e^{-x})^2} \tag{3.103}$$

becomes very small for large inputs. Multiplying a large number of these derivatives in the backpropagation (3.97) leads to vanishing gradients as $\partial \mathcal{L}\left(\boldsymbol{u}^K, \boldsymbol{v}\right) / \partial \boldsymbol{\theta}^k$ approaches zero and the training process stops. This made it, for a long time, infeasible to train deep neural networks.

One remedy is given in the form of the ReLU activation, which has only two possible derivative values: zero for negative inputs, and

one for nonnegative inputs. Still, it can happen that the other components of the derivatives become small, and the vanishing gradient phenomenon can persist. Moreover, the ReLU suffers from a nondifferentiability at zero. While this is often ignored in practice as it does rarely affect optimisation methods, some researchers argue that an artificial derivative at zero can even act as a hyperparameter [47].

Another option is to use *batch normalisation* [199] layers. These layers normalise their input to have a mean of zero and a standard deviation of one. If an input $\boldsymbol{f}$ has mean μ and standard deviation σ, then the batch normalisation computes

$$\boldsymbol{u} = \frac{\boldsymbol{f} - \boldsymbol{\mu}}{\sqrt{\sigma^2 + \varepsilon}}\,. \tag{3.104}$$

Here, $\boldsymbol{\mu}$ is a vector of the same size as $\boldsymbol{f}$ with all entries equal to μ, and ε is a numerical constant avoiding division by zero. Batch normalisation helps, among other things, to normalise the data to a range where the activation function provides useful derivatives.

Lastly, residual networks solve the vanishing gradient problem by design. The skip connection allows the gradient information to bypass the derivative of the convolutions and the inner activation function, thus avoiding small derivatives completely.

OPTIMISATION METHODS Thankfully, in practice there is no need to compute network derivatives by hand. Deep learning frameworks such as Tensorflow [1] make use of automatic differentiation [35] to provide the gradients, which greatly speeds up implementation.

To apply the derivative information to the parameters, most popular algorithms rely on *gradient descent*. For an overview of the plethora of optimisation methods, we refer to the monograph of Nocedal and Wright [270]. Another overview with a focus on modern machine learning is given by Kochenderfer and Wheeler [218]. Finally, the condensed review of Ruder [312] provides a quick reference from a practical viewpoint. In the following, we present a selection of the most popular optimisation algorithms which are important for this thesis.

PURE GRADIENT DESCENT The original gradient descent as known from convex optimisation is also the baseline of many algorithms used to train neural networks. Intuitively, the gradient of the loss function w.r.t. the parameters describes the steepest ascent direction of the loss. Thus, subtracting the gradient from the current parameter set with a reasonably small step size should yield a parameter combination with a smaller loss. Repeatedly applying this process steers the parameters into a local minimum of the loss.

For a single prediction $\boldsymbol{u}$ with ground truth $\boldsymbol{v}$ at a descent step n, one updates

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \alpha \frac{\partial \mathcal{L}(\boldsymbol{u}^K, \boldsymbol{v})}{\partial \boldsymbol{\theta}_n} \tag{3.105}$$

As the gradient denotes the steepest ascent direction in the loss function, one performs a step in the opposite direction. The step size, which is called the *learning rate* in the context of deep learning, is denoted by $\alpha > 0$.

Usually one trains a neural network on large sets of data. In this case, a naive strategy performs parameter updates based on single data samples. As the gradient computation in this case depends on the samples, this strategy is called *stochastic gradient descent* [301]. Due to the randomness of gradients, this process is able to escape bad local minima and saddle points of the energy surface. However, as one uses single samples from the large data set, the gradient information is noisy. On the other hand, computing all gradients on the full data set is computationally burdensome.

To combine the best of both worlds, researchers have proposed to use *mini-batch gradient descent*. Therein, each optimisation step averages gradient information on a small subset of the data, a so-called *batch*. Optimising the batch size is crucial for finding a good balance between randomness in the gradient and reliable gradient information.

MOMENTUM METHODS Methods that rely on pure gradient descent exhibit bad convergence around minima where gradients become small. Moreover, in settings where the energy has elongated level lines, gradient descent tends to oscillate orthogonally to the optimal descent direction.

To this end, *Polyak's heavy ball method* [287] introduces *momentum* to the descent trajectory. Intuitively, this strategy models a ball which moves on the energy surface. Instead of only letting the ball move down slopes due to gravity, one additionally considers the velocity of the ball that is accumulated while moving downhill.

One introduces a velocity vector $\boldsymbol{\vartheta}$ and changes the update rule to

$$\boldsymbol{\vartheta}_n = \gamma\, \boldsymbol{\vartheta}_{n-1} - \alpha \frac{\partial \mathcal{L}(\boldsymbol{u}, \boldsymbol{v})}{\partial \boldsymbol{\theta}_n}, \tag{3.106}$$

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \boldsymbol{\vartheta}_n. \tag{3.107}$$

Here, the parameter $\gamma \in [0, 1]$ is the velocity parameter. It controls the balance between accumulated previous gradients in the form of $\boldsymbol{\vartheta}$ and new updates. A velocity of $\gamma = 0$ yields the original gradient descent formulation, and $\gamma = 1$ allows arbitrarily large accumulation of gradients, i.e. the ball does not reach a terminal velocity.

A more efficient option is the use of *Nesterov accelerated gradient descent* [267], which considers the new gradients already at the updated

position, and thus performs a correction if the momentum leads to an overshoot.

Both momentum methods help to escape saddle points and bad local minima and to improve convergence around minima. For a rigorous analysis in terms of convergence rates, we refer e.g. to [270].

ADAPTIVE LEARNING METHODS While momentum methods were designed for classical optimisation problems, their use in neural networks has triggered many adaptations. A core challenge is that the multitude of parameters describes various features which do not live on the same scale. Moreover, some features provide useful gradients sparsely throughout several batches. This requires to adaptively tune the learning rate for each feature. As this is infeasible to do by hand, a multitude of *adaptive learning methods* has emerged.

The most popular one is *adaptive moment estimation (Adam)* [212]. It combines the advantages of two previous adaptive learning methods, namely RMSprop [177] and AdaGrad [111]. Adam tracks decaying averages of the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\sigma}$ of past gradients, each with their own momentum parameter β_1, β_2:

$$\boldsymbol{\mu}_n = \beta_1 \boldsymbol{\mu}_{n-1} + (1-\beta_1) \frac{\partial \mathcal{L}(\boldsymbol{u}, \boldsymbol{v})}{\partial \boldsymbol{\theta}_n}, \tag{3.108}$$

$$\boldsymbol{\sigma}_n = \beta_2 \boldsymbol{\sigma}_{n-1} + (1-\beta_2) \left(\frac{\partial \mathcal{L}(\boldsymbol{u}, \boldsymbol{v})}{\partial \boldsymbol{\theta}_n} \right)^2. \tag{3.109}$$

To counteract the initialisation bias, one computes bias-corrected estimates

$$\hat{\boldsymbol{\mu}}_n = \frac{\boldsymbol{\mu}_n}{1-\beta_1^n}, \tag{3.110}$$

$$\hat{\boldsymbol{\sigma}}_n = \frac{\boldsymbol{\sigma}_n}{1-\beta_2^n}. \tag{3.111}$$

Then the update rule is given by

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \frac{\alpha}{\sqrt{\hat{\boldsymbol{\sigma}}_n} + \varepsilon} \hat{\boldsymbol{\mu}}_n. \tag{3.112}$$

We see that the learning rate α is normalised by the standard deviation estimate $\sqrt{\hat{\boldsymbol{\sigma}}_n}$, where a small numerical constant ε avoids division by zero. This brings all gradients to the same scale. In addition, both estimates incorporate momentum.

There exist several other adaptive learning methods [218, 312], however, Adam is arguably the most popular one. In this thesis, we predominantly use the Adam optimiser with the suggested standard parameters $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$ [212] if not specified otherwise. Chapter 4 poses an exception to this rule. Therein, we use the quasi-second order method of L-BFGS [236] as the proposed model is shallow enough for this purpose.

3.4.4 *A Short History of Deep Learning*

The present success of deep learning builds on a long history of research. To understand the current state of deep learning, and to appreciate the advances in its mathematical foundations, it is important to keep this history in mind.

While deep learning today is omnipresent, it started as a subfield within the broad research area of artificial intelligence. This section is a humble attempt at summarising the main milestones in deep learning. It closely follows the excellent and extremely detailed overview of Schmidhuber [330], with additional references from the book of Goodfellow et al. [152]. Figure 3.10 accompanies this section with a visual timeline.

SURGES IN ARTIFICIAL INTELLIGENCE Much like any other research field, artificial intelligence experienced fruitful phases, and so-called *AI winters*, where research almost came to a halt due to various circumstances. In their book, Goodfellow et al. [152] identify three surges in research: *cybernetics*, *connectionism*, and *deep learning*.

The phase of cybernetics ranged from the 1940s to the 1960s and was mainly inspired by biological learning processes in the brain. The phase of connectionism dated from the 1980s to the 1990s, with its central concept being the idea that many neurons performing simple computations can solve complex tasks when arranged in a network. Lastly, deep learning emerged under its current name in 2006, with no AI winter in sight.

CYBERNETICS Already in 1943, McCulloch and Pitts [253] presented a linear model for the behaviour of neurons in the brain. However, this neuron was not able to learn, an idea which was brought forth by Hebb in 1949 [174].

The first trainable model was presented in 1958 in the form of the *Perceptron* by Rosenblatt [307]. It is considered the foundation of modern neural networks. A Perceptron is a single linear neuron with binary inputs and a single binary output. Based on weights and a bias, the neuron can learn its output. However, Minsky and Papert [256] showed that the Perceptron is not able to solve the XOR-problem: It cannot learn the logical exclusive or (XOR) function which is one when only one of two inputs is one, and zero otherwise. This was the beginning of the first AI winter.

Nevertheless, at the end of the cybernetics phase, Ivakhnenko provided one more foundation of deep learning with the *group method of data handling* [200]. These models arrange Perceptron-like neurons in a network, with polynomial activation functions. However, the number of neurons per layer and the number of layers are learned dynamically.

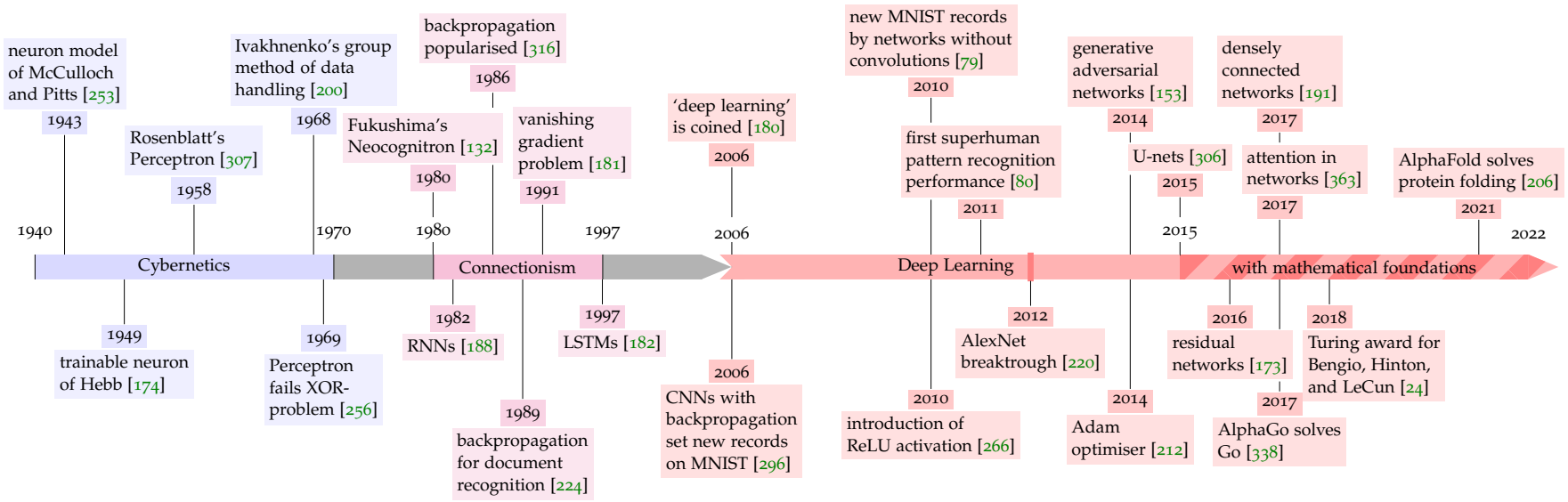


Figure 3.10: Historic timeline of the surges in artificial intelligence and selected milestones.

CONNECTIONISM In 1980, Fukushima [132, 133] presented the *Neocognitron*. Based on biological inspiration from experiments on the visual cortex of cats [192], the Neocognitron introduced convolutions as well as average-pooling. Thus, the Neocognitron was effectively the first deep convolutional network, even though not trained by means of backpropagation.

The phase of connectionism was also the birth of *recurrent neural networks (RNNs)*, first proposed by Hopfield in 1982 [188]. The distinction between RNN-like networks and feed-forward networks exists still to this day.

The development of backpropagation happened in parallel, and was only later joined with neural networks. Backpropagation was first explicitly described by Linnainmaa in 1970 [234, 235], however without a reference to neural networks. For neural networks, it was first proposed by Werbos in 1982 [389, 390], with follow-up works of LeCun [222]. Finally, the 1986 paper of Rumelhart et al. [316] popularised the concept of backpropagation significantly.

In 1989, backpropagation was successfully applied to a Neocognitron model by LeCun et al. [224, 225], thus learning to classify handwritten digits. This constituted a milestone: The combination of convolutional layers, pooling, and GPU implementations is still present in contemporary deep learning architectures.

In 1991, Hochreiter [181] was the first to identify the phenomenon of vanishing and exploding gradients as the central reason why deep networks were hard to train with backpropagation. Several solutions to this problem were developed, the most popular invention being *long short term memories (LSTMs)* [142, 182], a specific form of RNNs. A long time before residual networks solved the problem with skip connections, similar ideas in the design of LSTMs introduced constant derivatives such that backpropagated errors could not vanish.

In the middle of the 1990s, deep learning research entered a second AI winter. Goodfellow et al. [152] attribute this to both neural networks failing to fulfil expectations, as well as advances in other areas of machine learning.

DEEP LEARNING The term *deep learning* as we know it today was coined by Hinton in 2006 [179, 180]. Its march to success was mainly catalysed by two factors: the growing availability of data, and the advances in parallel computing hardware such as *graphics processing units (GPUs)*.

Shortly after, CNNs trained with backpropagation [296] set new records for classification on the MNIST handwritten digit dataset, and GPU implementations [71, 273] became increasingly popular.

Another MNIST record was set in 2010 through standard neural networks without convolutions by Ciresan et al. [79]. These networks were trained by means of GPU-based backpropagation, and, for the first

time, without any pre-training strategies. In 2011, Ciresan et al. [80] transferred this strategy to CNNs with max-pooling and ensembles thereof [81], constituting the first instance of superhuman performance for pattern recognition. Around this time, the introduction of ReLUs also solved the problem of vanishing gradients to a large extent [149, 266].

This finally led to the breakthrough year of 2012. Krizhevsky et al. [220] presented *AlexNet*, a GPU-based max-pooling CNN which improved the error on the ImageNet classification benchmark [318] by over 75% from 26.1% to 15.3% [152]. This tremendous advance popularised CNNs throughout various research communities and led to an explosion of performance. New benchmarks were quickly set, primarily for classification [340, 348, 406], but also for various other tasks [330].

Since then, deep learning has taken a hold of all research fields involving any form of signal processing. The introduction of new concepts and architectures [22, 153, 173, 191, 213, 306, 363] along with better optimisation methods [212], further progress in parallel computing hardware, the availability of large datasets [318], and the introduction of integrated development environments such as Tensorflow [1] led to an hitherto unseen explosion of interest in deep learning.

According to the Google Scholar statistics at the end of 2021, deep learning is omnipresent [155]. In Nature, the most influential journal in the last five years according to the h5-index, three out of the ten top publications make use of deep learning. Out of the top ten conferences and journals, three are purely devoted to learning. Note that these statistics cover all areas of research tracked by Google Scholar.

When focusing on computer vision and pattern recognition, there is almost no alternative to deep learning: At the *IEEE Conference of Computer Vision and Pattern Recognition*, the top computer science conference, the first non-deep learning paper out of the most cited ones over the last five years appears at position 43. The second non-deep learning paper is found at rank 96, not considering evaluations and papers which introduce datasets.

It is not surprising that in the year 2018, the *ACM A. M. Turing Award*, which is also referred to as the Nobel prize of computer science, was awarded to Yoshua Bengio, Geoffrey Hinton, and Yann LeCun for their contributions to neural networks and deep learning [24].

The advances in deep learning research have surprisingly rapid practical implications. In 2017, an artificial intelligence system by DeepMind relying on deep learning was able to beat one of the best Go players in the world [338]. Go is a competitive board game of Asian origin with a particular relevance for artificial intelligence due to its vast search space. The same company presented AlphaFold, which

is the first program to produce practically sufficient accuracy for the fundamental problem of protein folding in biology [206].

While this is certainly not the end of the road, a growing community of researchers believes that the progress in deep learning is slowed down by a fundamental lack of understanding of the inner mechanisms of neural networks.

3.4.5 *From Deep Learning to Deep Understanding*

The evolution of deep learning from 2012 on was largely data-driven: Deep neural networks were used as black-box models that could solve tasks when given large amounts of data. The excessive speed at which these models evolved left the understanding of them trailing behind. This led to several undesirable side effects.

One example is the concept of *adversarial examples* [154, 404]. In their seminal paper, Goodfellow et al. [154] show that a classification network can easily be fooled: Adding a small perturbation to an image which is imperceptible to a human leads the classifier to deviate from a very confident and correct classification to a wildly different one. This discovery has initiated a new field of research, where attackers try to fool networks, and defenders improve their architectures and optimisation algorithms to be robust to such adversarial attacks. At their core, this phenomenon relies on a nonsmoothness in the energy landscape of CNNs: While clean and corrupted images are close to each other in the input space, their corresponding outputs are far away from each other. From the viewpoint of PDEs, this indicates that CNNs suffer from a lack of well-posedness: A small change of the input data can lead to large deviations in the output.

Another example of a side effect is the *double descent phenomenon* [40]. A guiding principle for classical machine learning models is the *bias-variance trade-off*. Complex models have a small bias, but a large variance and vice versa. This implies that a model should be designed in such a way to hit the sweet spot between over- and under-fitting. This, however, does not seem to hold for modern neural architectures as shown by Belkin et al. [40]: If models are over-parameterised, they exhibit a double descent phenomenon during training where the test error decreases, increases, and decreases again. While the explanation is more intricate, this example stresses that CNNs seem to fundamentally change established paradigms, warranting the need for mathematical foundations.

The goal of this thesis is to improve the transparency of models. On one hand, we equip already transparent image processing models with trainable components, without obfuscating the behaviour of the model. On the other hand, we connect the elementary building blocks of specific neural network architectures to concepts for PDEs, thus shedding light on their inner workings. We are not alone in this

endeavour: In the meantime, many researchers have contributed to the *mathematical foundations of deep learning*. The following section presents a selection of related work from these fields.

3.5 MATHEMATICAL FOUNDATIONS OF DEEP LEARNING

Establishing mathematical foundations of deep learning has become a prominent research avenue. Researchers from various communities across PDEs, variational methods, wavelets, inverse problems, and many more have started to use their expertise to connect neural networks to their fields; see e.g. [365] for an early overview. This section is intended as a broad introduction into these connections. More detailed discussions can be found in the individual parts and chapters of the thesis.

DIFFERENTIAL EQUATIONS AND NEURAL NETWORKS For us, the most important connections are those between differential equations and neural networks. They form a basis for our considerations in Part II of the thesis.

One of our core ideas is to translate numerical algorithms for PDEs into neural building blocks. This strategy is not shared by many others. For example, one can use inspiration from acceleration strategies for explicit schemes [240], implicit schemes [229], higher order numerics based on Runge–Kutta methods [241], semi-implicit schemes [166], and even discretisations of integrodifferential equations [49] to come up with improved neural architectures. However, such examples are rare.

Often times, the connections are used to derive stability and robustness guarantees for neural networks. For example, Ruthotto and Haber [319] interpret residual networks as explicit Euler schemes for PDEs and derive conditions for Euclidean stability and robustness of the network evolution. In a similar spirit, Haber and Ruthotto [167] come up with similar conditions from an ODE viewpoint. In both cases, however, they assume monotone activation functions. We on the other hand derive stability conditions while also allowing nonmonotone activation functions.

Mathematical guarantees are also important considerations for NODEs (see Section 3.4.2) as the continuous-time extension of residual networks [73]. Recent works study their approximation capabilities [315, 409], stability properties [359], and robustness [399]. While the concept of NODEs attracts lots of positive attention, several recent works [82, 164, 274] argue that some of the original claims of NODEs do not hold true.

Researchers have also started to use the flexibility of deep learning to learn PDE dynamics from data [72, 75, 238, 314, 324] or to solve them [86, 114, 145, 291, 293]. These ideas have also been extended

to PDEs on manifolds [231] and special forms of networks such as graph [118] and quantised neural networks [41].

VARIATIONAL METHODS AND NEURAL NETWORKS Energy-based models have a long tradition in learning [109, 226, 350]. Recently, deep learning has been used to design variational models in a data-driven way; see e.g. [121, 285, 290].

Kobler et al. [217] present *variational networks*, where the gradient descent steps of a variational energy are parametrised and learned. This is a form of unrolling [260, 345], which constitutes one fundamental way to train such models [75, 216].

Another option is to use contrastive learning [178], where the energy is maximised on training pairs that do not fulfil the model assumptions, and minimised on pairs which do so. The adversarial training process of Wasserstein GANs [22] is an instance of this training strategy.

Similar as for PDEs, learning can not only be used to model variational energies, but also to solve them. This is the idea of *deep energies* [150], where a neural network is tasked to produce solutions which minimise a given energy functional. We extend this idea to PDEs by prescribing the squared residual of a PDE as an optimisation objective. This is especially useful when no energy for a PDE is available, or when it is hard to implement numerically.

As variational methods are a common tool for solving inverse problems, these connections manifest in several learned iterative schemes for inverse problems and trainable regularisers. Extensive overviews are given in [23, 252].

An increasingly popular interpretation is also that neural network architectures possess a regularising effect [104, 360]. This is known under the notion of *deep priors* [360]. Other authors suggest to use networks that were trained for denoising as regularisers for other tasks [305]. Our view on neural network architectures is different: We argue that the network design expresses capabilities of a numerical solver, while its training corresponds to finding a suitable model.

WAVELETS AND CNNS The seminal work of Bruna and Mallat [60] employs wavelet operations to come up with scattering networks that perform well on classification problems. This has been the starting point for a number of wavelet-inspired CNNs; see e.g. [131, 302, 392, 393] and the references therein. Usually they exploit the spectral information or the multiscale nature of wavelets. In particular, Ramzi et al. [295] introduce trainable wavelets as *learnlets* and connect their model to U-nets. For our goals, wavelets often serve as a practical test bed due to their efficiency, and as a suitable framework to connect them to CNNs based on their inherent connection to diffusion PDEs [262–264, 344, 377].

ROBUSTNESS, APPROXIMATION, AND INVARIANCES Apart from their connections to PDEs, robustness and stability of CNNs can be achieved for example by controlling the Lipschitz constant of the mapping that individual layers or full networks represent, either by design [28, 227, 351] or on the fly during training [62, 157]. Genzel et al. [140] empirically observe that CNNs for inverse problems naturally exhibit certain robustness properties.

The approximation capabilities of CNNs are important to measure the expressiveness of different models. Already in 1989, Cybenko [89] as well as Hornik et al. [190] showed that feedforward networks with sigmoid activation functions can approximate functions with arbitrary precision given enough hidden units within a single hidden layer. In modern times, the question of the influence of depth on the expressiveness of networks has become increasingly popular; see e.g. [95, 116, 233, 276, 286].

Lastly, bringing invariances beyond shift-invariance to neural networks is an important part of their mathematical foundations. In particular, we are interested in rotation invariance. While learning-inspired approaches such as [74, 123, 221, 339] only approximate rotation invariance, other works guarantee it by design [84, 102, 112, 139, 249, 337, 382, 395]. However, these methods are often complex in their design, inflate the network architecture heavily, or suffer from a too coarse discretisation of sampled orientations. In Chapter 8, we propose a strategy inspired by diffusion models which solves some of the aforementioned problems.

This concludes our overview on the mathematical foundations of deep learning. In the following, we conclude this chapter by reviewing the practical problems that we tackle in this thesis.

3.6 APPLICATIONS

In the following, we briefly discuss the central experimental applications which we encounter in this thesis: *denoising* and *inpainting*. Denoising serves as a good test bed for our models, while the motivation of our inpainting experiments is to improve inpainting-based compression models.

3.6.1 *Denoising*

Denoising is a fundamental problem in image processing. As one of the oldest problems, it often constitutes the first step in longer processing pipelines, and subsequent methods build upon good denoising results.

We can describe *additive noise* by

$$\boldsymbol{f} = \boldsymbol{v} + \boldsymbol{n}, \tag{3.113}$$

where the corrupted image $\boldsymbol{f}$ arises from a clean version $\boldsymbol{v}$ by adding noise $\boldsymbol{n}$. In this thesis, we only consider *additive white Gaussian noise*, i.e. $\boldsymbol{n}$ follows a Gaussian distribution with mean zero and standard deviation σ. The goal of denoising is to recover $\boldsymbol{v}$ from $\boldsymbol{f}$ without knowledge of the noise realisation $\boldsymbol{n}$ as best as possible.

There exists a plethora of successful denoising strategies. A non-exhaustive lists includes diffusion models [16, 279, 369], total variation regularisation [313], bilateral filters [355], nonlocal means [61], patch-based methods [90], wavelet shrinkage [85, 107, 248], dictionary learning [8, 75, 308, 331], and of course deep learning [411]. For an overview of traditional and deep learning models for denoising, we refer to the overviews [161] and [353], respectively.

For us it is not about achieving state-of-the-art denoising performance. Nowadays, this is not possible without highly sophisticated models. In Part I, we use denoising as a simple test bed to improve classical methods by means of learning. In Part II, we choose denoising since diffusion-based denoising is the prototypical setting of a well-posed process. This helps us to establish our mathematical guarantees.

Figure 3.11 visualises how the diffusion models from Table 3.1 perform in a denoising setting. To this end, we corrupt the *peppers* image with Gaussian noise of standard deviation $\sigma_n = 40$. We have optimised all involved parameters such as diffusion time, contrast and presmoothing parameters to obtain optimal results w.r.t. MSE. As this only reflects the denoising quality on a single image and with the same diffusivity function for all nonlinear models, this comparison is not supposed to give a strict ranking of models.

Nevertheless, we can make out the systematic strengths and weaknesses of the models. The linear isotropic homogeneous and biharmonic diffusion models blur the image to a certain extent to remove the high-frequent noise components. The Perona–Malik model suffers from noise outliers which are preserved as they produce large gradients. Similar effects can be found in the higher-order nonlinear models and that of Niessen as the smoothing parameter is small. This justifies the motivation of the regularised model of Catté et al.: By presmoothing the noisy image, more reliable gradient information is obtained and the denoising quality improves. As a resulting downside, it cannot remove noise around prominent edges. As the nonlinear model of Fritsch considers the gradient information of the noisy image, it cannot extract useful structural information and essentially collapses to homogeneous diffusion. Overall, the EED model performs best as it additionally employs directional smoothing around edges. With small gaps in performance each, the Catté and Perona–Malik models follow, with the remaining models trailing behind.

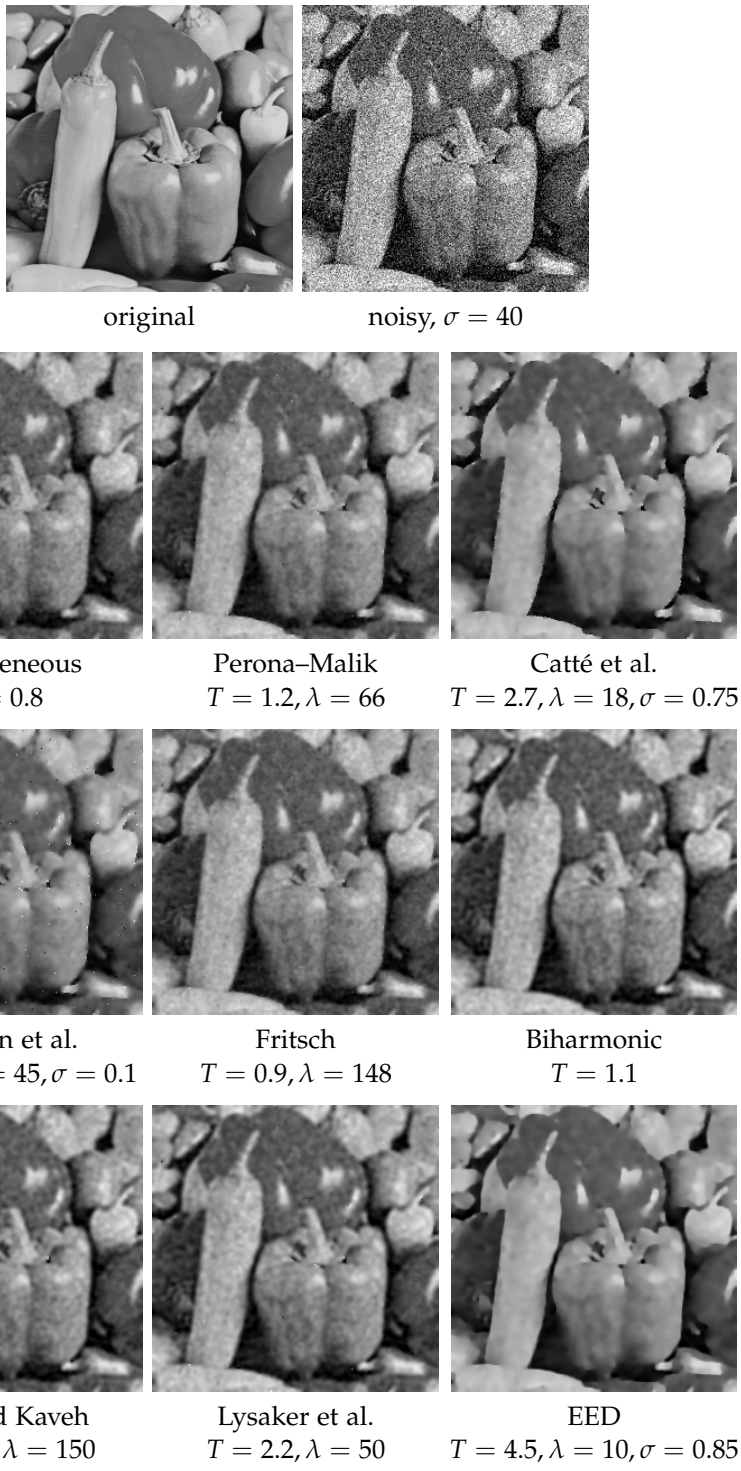


Figure 3.11: Denoising effect of the diffusion models from Table 3.1 for the *peppers* image. All models use optimised diffusion times and parameters and the exponential Perona–Malik diffusivity in the nonlinear case.

3.6.2 *Inpainting*

Inpainting is the task of reconstructing missing data in images from a subset of the original image data. Originally, this technique was used for manual art restoration, where damaged paintings were in-painted by hand. For digital images, the technique was popularised by Masnou and Morel [251] as well as Bertalmío et al. [46].

As for denoising, many strategies have been developed for solving the inpainting problem founded on diffusion [52, 135, 334, 378], Euler's elastica [56, 251, 335], Shepard interpolation [4, 258, 280, 336], exemplar-based strategies for textures [87, 122, 208], deep learning [176, 197, 237, 277, 298, 321, 400, 403], and many others; see e.g. [45, 162] for overviews.

Applications include upsampling [34, 38, 247], denoising [5], colourisation [282], object removal [87], image compression [76, 134, 280, 327], and video compression [17, 18, 219, 328].

The success of PDE-based image compression is our main motivation for considering inpainting problems within this thesis. Already in 1988, Carlsson [64] showed that images can be compressed using a sparse set of pixels and recovered by means of inpainting. In 2005, Galić et al. [134, 135] introduced *diffusion-based image compression*. Therein, one stores only a very sparse subset of pixels in the form of a triangular tree (cf. also [103]), and the image is reconstructed from these data by diffusion processes. A series of improvements followed: Schmaltz et al. [329] managed to beat the transform-based codec of JPEG2000 [349] with their R-EED codec relying on anisotropic diffusion inpainting of rectangular tree-based data points, and the R-EED-LP codec of Schmaltz et al. [327] extended this strategy to colour images. This constitutes the current state-of-the-art codec for general purpose image compression, albeit several improvements of individual pipeline components [77, 207] and adaptations to specific image data [186, 203, 204, 245] have been proposed.

DIFFUSION-BASED INPAINTING We are mainly concerned with the inpainting problem itself, and less with the compression aspect. For a continuous image $f : \Omega \to \mathbb{R}$ which is only available on the *inpainting mask* as a subset $K \subset \Omega$ of the image domain, we want to obtain a reconstruction u as the solution of the inpainting equation

$$(1 - c)\, Lu - c\,(u - f) = 0 \tag{3.114}$$

with reflecting boundary conditions and a general differential operator L. Here, $c : \Omega \to \mathbb{R}$ is a confidence function which determines whether a data point is known or not. In case of a binary confidence

which is one for known data points and zero for unknown points, a reformulation of the inpainting equation is more intuitive:

$$Lu = 0 \qquad \text{on } \Omega \backslash K, \tag{3.115}$$

$$u = f \qquad \text{on } K. \tag{3.116}$$

Thus, the known data are kept, and the missing parts are interpolated by the process specified by the differential operator L.

Diffusion processes have proven particularly useful for this task due to their implicit smoothness assumptions and long-range interactions. For example, choosing $L = \Delta$ leads to homogeneous diffusion inpainting. The choice of $L = -\Delta^2$ gives biharmonic inpainting, and $L = \nabla^\top (\boldsymbol{D} \nabla u)$ allows anisotropic diffusion inpainting.

A fundamental problem in inpainting-based compression is to optimise a discrete inpainting mask such that the data is suitable for reconstructing the image, and also easy to store. A multitude of data optimisation strategies has been proposed, relying on analytic results [39], continuous nonsmooth optimisation [50, 76, 183, 184, 271], sparsification [97, 246, 250] and densification [77, 93, 208] strategies, and on feature-based information such as gradient data [55], corners [19], and point features [3, 64, 92, 383]. A detailed review is given in Chapter 10.

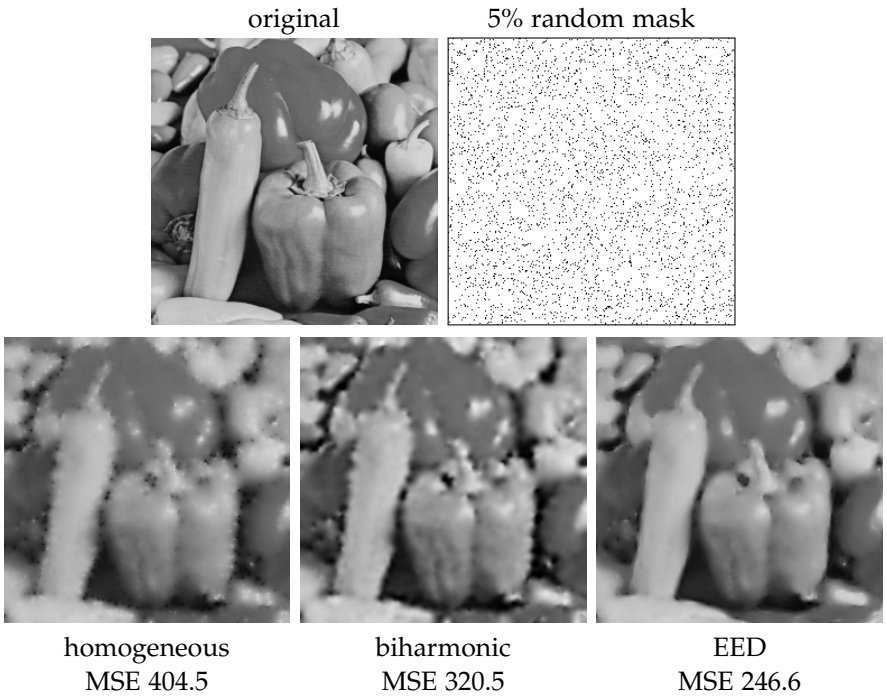
For our work, the spatial optimisation strategies of Mainberger et al. [246] are essential. They consist of *probabilistic sparsification (PS)* and *nonlocal pixel exchange (NLPE)*.

Probabilistic sparsification starts with a full mask and successively removes a fraction p of candidate pixels, computes the inpainting, and reintroduces a fraction q of the candidates with the largest local inpainting error. One repeats this step until reaching a desired mask density d which describes the percentage of remaining pixels.

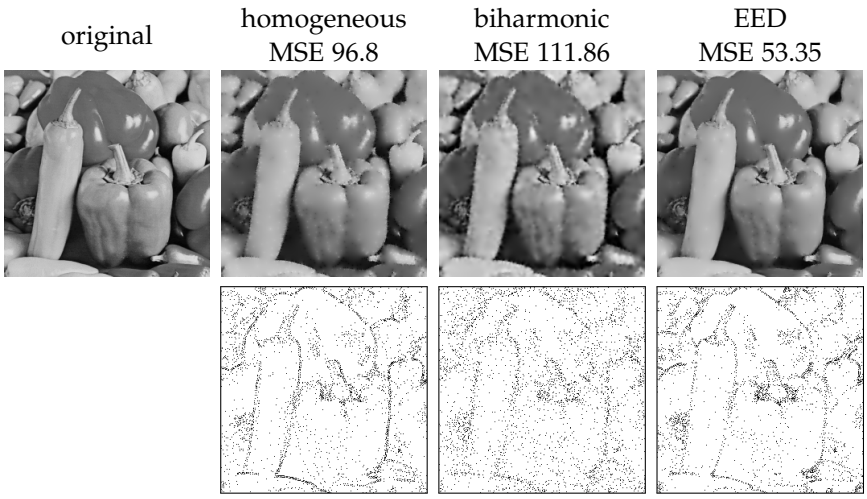
Since sparsification is a greedy local approach, it can get trapped in bad local minima. NLPE provides a remedy for this: Therein, pixel candidates in a sparsified mask are exchanged for an equally large set of non-mask pixels. If the new inpainting result improves, the exchange is kept, otherwise it is discarded. In theory, NLPE can only improve the mask, but in practice convergence is slow.

Figure 3.12 depicts the inpainting performance of homogeneous, biharmonic, and edge-enhancing diffusion in the case of random masks and stochastically optimised ones. We choose the *peppers* image and prescribe a density of 5%. For the optimised masks, we take the best out of five sparsification runs and optimise the result further with ten cycles of NLPE. In each cycle, a mask point is visited once on average. The sparsification uses the standard candidate fractions $p = 0.1$ and $q = 0.05$.

Without mask optimisation, the ranking is as expected: The more complex the inpainting operator, the better the inpainting result gets. A characteristic of the homogeneous diffusion inpainting is that mask points form singularities. While the biharmonic model does not suffer



(a) random mask on *peppers* with 5% density



(b) probabilistic sparsification with additional nonlocal pixel exchange on *peppers* with 5% density

Figure 3.12: Visual comparison of inpainting operators on *peppers* without and with spatial mask optimisation. Mask points are shown in black, and mask images are framed for better visibility. Optimal masks for different inpainting operators can differ strongly.

from these artefacts, it still lacks anisotropy and thus creates ragged edges. Given that mask points are chosen randomly, the inpainting quality of EED is already impressive.

However, the spatial optimisation strategies boost all methods to completely different quality levels. Interestingly, the homogeneous diffusion model supersedes the biharmonic one in this example. Its mask points cluster around edges, while those for biharmonic inpainting are more evenly distributed. Yet both are not competitive with EED. As already few mask points along edges are sufficient to reconstruct them, more points can be placed in other image areas.

This concludes our review of related work. The following chapter commences Part I where we equip classical mathematical image processing models with trainable components, starting with an adaptive wavelet shrinkage model.

Part I

IMPROVING MATHEMATICAL MODELS THROUGH LEARNING

4

TRAINABLE ADAPTIVE WAVELET SHRINKAGE

This chapter contains our first novel contributions and is an instance of the first vision outlined therein: improving mathematical models by means of learning. While yielding highly tailored solutions, deep learning models are often too complex for a rigorous analysis. In contrast, classical model-driven approaches provide a sound theoretical foundation but often cannot compete with their learning-based counterparts. To this end, we present an approach for equipping wavelet shrinkage with trainable components to learn adaptive dynamics of the process. As a crucial difference to other model-based learning ideas, we reduce the learned parameters to obtain a condensed model with only a handful of control parameters.

We exploit the flexibility of a learning-based approach to train a smooth shrinkage function that adapts both to the wavelet scales and to the noise level. The proposed shrinkage function can even amplify wavelet coefficients and thus enhance important image structures, a property that most established shrinkage functions do not share. A low number of trainable parameters allows us to manually inspect the learned results and infer smooth connections between them. From these, we design a generic compact shrinkage function that incorporates the learned adaptivity, while only using two parameters. Experiments show that our shrinkage function outperforms classical ones by a significant margin. To the best of our knowledge, this is the first work to present a learning-based shrinkage function with this level of compactness.

The choice of wavelet-based denoising as a framework for our ideas is well-motivated: First, wavelet shrinkage exhibits a long history of research as well as established connections to diffusion, ideas which we can use within our model design. Moreover, wavelet shrinkage poses only two modelling challenges: finding a suitable wavelet basis, and designing an appropriate shrinkage function. This allows us to focus on finding good trainable model components. Lastly, wavelet shrinkage can be very efficient. Results are usually obtained with just one shrinkage step consisting of a forward and a backward transformation with a nonlinear function application in between. Thus, we can perform highly efficient backpropagation. In fact, this chapter is the only instance where the parameter gradients are computed by hand and used within a quasi-second order optimisation method.

RELATED WORK We have already covered the basic idea of wavelet shrinkage in Section 3.3. Classical wavelet shrinkage [108] functions

use a single threshold parameter for all scales of the wavelet transformation inducing a binary decision: If the wavelet coefficient is too small, it likely is a result of noise and is eliminated. If it is large, it probably represents important signal structure and is thus kept and possibly modified. Prominent representatives of such functions are hard [248], soft [107] and garrote [136] shrinkage.

However, individual scales might require distinct thresholds as they are differently affected by noise. To this end, adaptive shrinkage methods have been proposed. Early works of Zhang et al. [414, 415] already study a smooth adaptive shrinkage function with trainable thresholds. Other authors train arbitrarily shaped shrinkage functions [7, 175, 346], some also train the wavelets [158] or even general adaptive filters for shrinkage operations [331]. Non-trainable adaptive statistical models include [69, 230, 288]. Most learning-based methods produce large amounts of trained parameters while relations between them are rarely investigated. In contrast to this, we directly employ a tightly parametrised model from which it is easy to infer smooth underlying parameter relations.

An important connection between two-dimensional wavelet shrinkage and nonlinear diffusion filtering has been established by Mrázek and Weickert [263]. It allows us to directly translate a diffusivity into a trainable shrinkage function. We use a so-called *forward-and-backward* (FAB) diffusivity, resulting in a shrinkage function that can amplify coefficients. Only few works employ this property directly [343], but also results for learned shrinkage functions suggest its usefulness [175]. The corresponding concept of backward diffusion has also shown to be successful [44, 75, 147, 386].

PUBLICATION INFORMATION This chapter is based on the conference paper of Alt and Weickert [13] presented at the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing. It is extended with detailed descriptions of the shift-invariant wavelet shrinkage algorithm, a detailed derivation of the parameter gradients, and additional figures and graphs. Moreover, Appendix A links rotation-invariant wavelet shrinkage to diffusion discretisations.

ORGANISATION OF THE CHAPTER The remainder of the chapter is structured as follows: In Section 4.1 we review shift-invariant wavelet shrinkage in the two-dimensional setting and its relation to nonlinear diffusion. We propose our model in Section 4.2, which is experimentally evaluated in Section 4.3. Finally, Section 4.4 summarises our conclusions.

4.1 REVIEW: SHIFT-INVARIANT WAVELET SHRINKAGE

In Section 3, we have already introduced one-dimensional wavelet shrinkage. In the following, we review the two-dimensional shift-invariant extension which provides the basis of this chapter.

Let us consider a noisy image $\boldsymbol{f}$ which originates as $\boldsymbol{f} = \boldsymbol{v} + \boldsymbol{n}$ from a clean image $\boldsymbol{v}$ corrupted by additive white Gaussian noise $\boldsymbol{n}$. The two-dimensional shift-invariant wavelet shrinkage follows the same three-step structure as its one-dimensional counterpart.

ANALYSIS In the analysis step, the input image $\boldsymbol{f}$ is transformed into wavelet and scaling coefficients. In contrast to the one-dimensional setting, one now obtains three sets of wavelet coefficients: two axial sets, and one diagonal set. We assume that the image is of size $2^L \times 2^L$ such that we obtain L wavelet scales. We denote the scaling coefficients on scale $\ell = 1, \dots, L$ by $\boldsymbol{w}^{(\ell)}$, and the corresponding wavelet coefficients by $\boldsymbol{w}_x^{(\ell)}, \boldsymbol{w}_y^{(\ell)}$, and $\boldsymbol{w}_{xy}^{(\ell)}$. They are computed from the noisy image $\boldsymbol{f}$ as

$$\boldsymbol{w}^{(\ell)} = \boldsymbol{Q}^{(\ell)} \boldsymbol{f}, \tag{4.1}$$

$$\boldsymbol{w}_x^{(\ell)} = \boldsymbol{Q}_x^{(\ell)} \boldsymbol{f}, \tag{4.2}$$

$$\boldsymbol{w}_y^{(\ell)} = \boldsymbol{Q}_y^{(\ell)} \boldsymbol{f}, \tag{4.3}$$

$$\boldsymbol{w}_{xy}^{(\ell)} = \boldsymbol{Q}_{xy}^{(\ell)} \boldsymbol{f}. \tag{4.4}$$

In the case of the Haar wavelet basis, the matrices $\boldsymbol{Q}^{(\ell)}, \boldsymbol{Q}_x^{(\ell)}, \boldsymbol{Q}_y^{(\ell)}, \boldsymbol{Q}_{xy}^{(\ell)}$ implement convolutions with the following stencils:

$$\boldsymbol{Q}^{(\ell)} \equiv \frac{1}{2^\ell h} \underbrace{\boxed{\begin{array}{c} \\ \mathbf{1} \\ \\ \end{array}}}_{2^\ell} \Big\} 2^\ell \quad , \qquad \boldsymbol{Q}_x^{(\ell)} \equiv \frac{1}{2^\ell h} \underbrace{\boxed{\begin{array}{c|c} \\ \mathbf{1} & -\mathbf{1} \\ \\ \end{array}}}_{2^{\ell-1} \quad 2^{\ell-1}} \Big\} 2^\ell \quad ,$$

$$\boldsymbol{Q}_y^{(\ell)} \equiv \frac{1}{2^\ell h} \underbrace{\boxed{\begin{array}{c} -\mathbf{1} \\ \hline \mathbf{1} \end{array}}}_{2^\ell} \begin{array}{l} \Big\} 2^{\ell-1} \\ \Big\} 2^{\ell-1} \end{array} \quad , \qquad \boldsymbol{Q}_{xy}^{(\ell)} \equiv \frac{1}{2^\ell h} \underbrace{\boxed{\begin{array}{c|c} -\mathbf{1} & \mathbf{1} \\ \hline \mathbf{1} & -\mathbf{1} \end{array}}}_{2^{\ell-1} \quad 2^{\ell-1}} \begin{array}{l} \Big\} 2^{\ell-1} \\ \Big\} 2^{\ell-1} \end{array} \quad .$$

Here we observe two things. First, we notice that the two-dimensional wavelet basis can be constructed as the tensor product of its one-dimensional counterpart. For example, the wavelet coefficients in x-direction can be computed by first employing a lowpass filter in y-direction, followed by the highpass filter in x-direction, or vice versa.

Secondly, we note that the output coefficients are of the same size as the input image, as the filter matrices are square. This is the effect

of the shift-invariant wavelet transform [85]. In the original wavelet transformation, one would need to equip the convolutions with strides of the filter size, i.e. a stencil at scale ℓ is only applied to every 2^ℓ-th pixel.

As a consequence, the naive implementation of the shift-invariant transformation is inefficient as stencils on coarse scales become as large as the original image. The *algorithme à trous* of Holschneider et al. [187] presents a remedy to this problem. Instead of computing the coefficients independently of each other, one computes coefficients on coarser scales from the results on finer scales by

$$\boldsymbol{w}^{(\ell+1)} = \boldsymbol{W}^{(\ell)} \boldsymbol{w}^{(\ell)}, \tag{4.5}$$

$$\boldsymbol{w}_x^{(\ell+1)} = \boldsymbol{W}_x^{(\ell)} \boldsymbol{w}^{(\ell)}, \tag{4.6}$$

$$\boldsymbol{w}_y^{(\ell+1)} = \boldsymbol{W}_y^{(\ell)} \boldsymbol{w}^{(\ell)}, \tag{4.7}$$

$$\boldsymbol{w}_{xy}^{(\ell+1)} = \boldsymbol{W}_{xy}^{(\ell)} \boldsymbol{w}^{(\ell)} \tag{4.8}$$

for all $\ell = 0, \dots, L-1$. One formally initialises with $\boldsymbol{w}^{(0)} = \boldsymbol{f}$. The modified filter matrices $\boldsymbol{W}^{(\ell)}, \boldsymbol{W}_x^{(\ell)}, \boldsymbol{W}_y^{(\ell)}, \boldsymbol{W}_{xy}^{(\ell)}$ implement the convolutions with the stencils

$$\boldsymbol{W}^{(\ell)} \equiv \frac{1}{2^\ell h} \begin{array}{|c|c|c|} \hline 1 & & 1 \\ \hline & \mathbf{0} & \\ \hline 1 & & 1 \\ \hline \end{array} \begin{array}{l} \Big\} 2^{\ell-1}-1 \\ \\ \\ \underbrace{}_{2^{\ell-1}-1} \end{array}, \quad \boldsymbol{W}_x^{(\ell)} \equiv \frac{1}{2^\ell h} \begin{array}{|c|c|c|} \hline 1 & & -1 \\ \hline & \mathbf{0} & \\ \hline 1 & & -1 \\ \hline \end{array} \begin{array}{l} \Big\} 2^{\ell-1}-1 \\ \\ \\ \underbrace{}_{2^{\ell-1}-1} \end{array},$$

$$\boldsymbol{W}_y^{(\ell)} \equiv \frac{1}{2^\ell h} \begin{array}{|c|c|c|} \hline -1 & & -1 \\ \hline & \mathbf{0} & \\ \hline 1 & & 1 \\ \hline \end{array} \begin{array}{l} \Big\} 2^{\ell-1}-1 \\ \\ \\ \underbrace{}_{2^{\ell-1}-1} \end{array}, \quad \boldsymbol{W}_{xy}^{(\ell)} \equiv \frac{1}{2^\ell h} \begin{array}{|c|c|c|} \hline -1 & & 1 \\ \hline & \mathbf{0} & \\ \hline 1 & & -1 \\ \hline \end{array} \begin{array}{l} \Big\} 2^{\ell-1}-1 \\ \\ \\ \underbrace{}_{2^{\ell-1}-1} \end{array}.$$

We observe that these stencils represent dilated versions of the filters at the finest scale. The introduction of holes — *trous* in French — in the form of zeros into the stencils gives the algorithm its name.

As a consequence, the pixels involved in the computation of one set of coefficients at scale ℓ drops from $2^{2\ell}$ down to four, making the algorithm highly efficient.

The final analysis step is thus given by stacking all matrices into a single one $\boldsymbol{W}$ such that

$$\boldsymbol{W}\boldsymbol{f} = \left(\boldsymbol{w}_x^{(1)}, \boldsymbol{w}_y^{(1)}, \boldsymbol{w}_{xy}^{(1)}, \boldsymbol{w}_x^{(2)}, \dots, \boldsymbol{w}_x^{(L)}, \boldsymbol{w}_y^{(L)}, \boldsymbol{w}_{xy}^{(L)}, \boldsymbol{w}^{(L)}\right)^\top. \tag{4.9}$$

The intermediate scaling coefficients $\boldsymbol{w}^{(\ell)}$ for $\ell < L$ do not need to be stored as this information is redundant.

SHRINKAGE A shrinkage function S_θ with a threshold parameter θ is applied individually to the wavelet coefficients. The scaling coefficients remain unchanged.

Many shrinkage functions have been proposed. We will consider the most prominent ones of hard [248], soft [107], and garrote [136] shrinkage for comparison. Figure 4.1 displays them along with their mathematical formulation. Hard wavelet shrinkage treats coefficients as either completely stemming from noise, or from clean image structures. Thus, it eliminates all coefficients which lie in $[-\theta, \theta]$, and leaves the rest of them unchanged. Soft shrinkage considers coefficients outside of this interval to be partly corrupted, and additionally shrinks them by θ. Garotte shrinkage combines both ideas by shrinking larger coefficients gradually less.

While these functions are easy to use in a practical setting, they suffer from applying the same threshold parameter to all scales and their binary decision structure. Finer scales might require a different threshold than coarser scales as they are more affected by noise. Furthermore, there is no clear separation between noise and signal coefficients such that eliminating too many coefficients always destroys signal details and eliminating too few leaves noise in the reconstruction.

The shrinkage function is typically applied pointwise to the individual wavelet coefficients, such that

$$\begin{aligned} S_\theta(\boldsymbol{W}\boldsymbol{f}) = \Big(& S_\theta\left(\boldsymbol{w}_x^{(1)}\right), S_\theta\left(\boldsymbol{w}_y^{(1)}\right), S_\theta\left(\boldsymbol{w}_{xy}^{(1)}\right), S_\theta\left(\boldsymbol{w}_x^{(2)}\right), \dots, \\ & S_\theta\left(\boldsymbol{w}_x^{(L)}\right), S_\theta\left(\boldsymbol{w}_y^{(L)}\right), S_\theta\left(\boldsymbol{w}_{xy}^{(L)}\right), \boldsymbol{w}^{(L)} \Big)^\top . \end{aligned} \tag{4.10}$$

However, in the following we will see that a more sophisticated shrinkage operation connects wavelet shrinkage to nonlinear diffusion and yields a more rotationally invariant result.

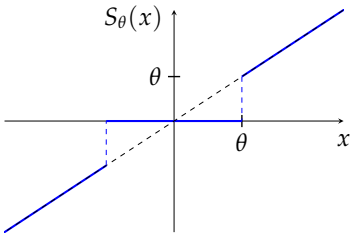
SYNTHESIS Finally, the backtransformation of the shrunken coefficients is computed iteratively starting from scale L as

$$\begin{aligned} \boldsymbol{w}^{\ell-1} = \frac{1}{4} \Big(& \left(\boldsymbol{W}^{(\ell)}\right)^\top \boldsymbol{w}^{(\ell)} + \left(\boldsymbol{W}_x^{(\ell)}\right)^\top S_\theta\left(\boldsymbol{w}_x^{(\ell)}\right) \\ & + \left(\boldsymbol{W}_y^{(\ell)}\right)^\top S_\theta\left(\boldsymbol{w}_y^{(\ell)}\right) + \left(\boldsymbol{W}_{xy}^{(\ell)}\right)^\top S_\theta\left(\boldsymbol{w}_{xy}^{(\ell)}\right) \Big) \end{aligned} \tag{4.11}$$

for all $\ell = L, \dots, 1$, producing the reconstruction $\boldsymbol{u} = \boldsymbol{w}^{(0)}$. The additional factor stems from the shift-invariant transformation; the original wavelet transform is orthonormal. If we stack the iterative application of the scaled and transposed matrices into a single backtransformation $\widetilde{\boldsymbol{W}}$, the reconstruction is computed as

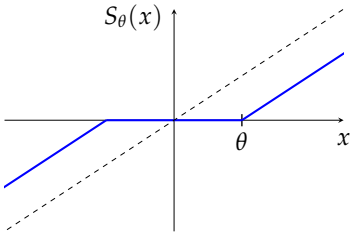
$$\boldsymbol{u} = \widetilde{\boldsymbol{W}} S_\theta(\boldsymbol{W}\boldsymbol{f}) . \tag{4.12}$$

function plot **mathematical expression**



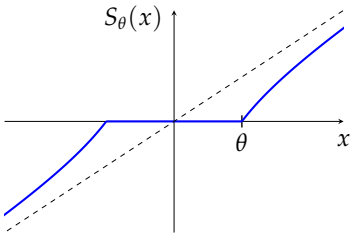
$$S_\theta(x) = \begin{cases} 0, & |x| \leq \theta \\ x, & |x| > \theta \end{cases}$$

hard shrinkage



$$S_\theta(x) = \begin{cases} 0, & |x| \leq \theta \\ x - \theta\,\mathrm{sgn}(x), & |x| > \theta \end{cases}$$

soft shrinkage



$$S_\theta(x) = \begin{cases} 0, & |x| \leq \theta \\ x - \frac{\theta^2}{x}, & |x| > \theta \end{cases}$$

garrote shrinkage

Figure 4.1: Hard, soft, and garrote shrinkage functions. The thresholding parameter θ determines the deadzone where wavelet coefficients are completely eliminated. The dashed grey line denotes the identity function.

RELATION TO NONLINEAR DIFFUSION For the two-dimensional wavelet transform, wavelet coefficient channels for x-, y-, and diagonal direction are obtained. To design rotationally invariant shrinkage rules, special care is required. Mrázek and Weickert [263] achieve this by a channel coupling that is inspired by nonlinear diffusion filtering. Their Haar wavelet shrinkage rule for three individual coefficients w_x, w_y, and w_{xy} at the same position is given by

$$\boldsymbol{S}_\theta \left(\begin{pmatrix} w_x \\ w_y \\ w_{xy} \end{pmatrix} \right) = \left(1 - g\left(w_x^2 + w_y^2 + 2\,w_{xy}^2\right)\right) \begin{pmatrix} w_x \\ w_y \\ w_{xy} \end{pmatrix}. \tag{4.13}$$

The argument $w_x^2 + w_y^2 + 2\,w_{xy}^2$ is a consistent approximation to the rotationally invariant gradient magnitude $|\nabla u|^2$. The function g is a diffusivity function from a nonlinear diffusion filter; see Section 3.1.1.

Note that in their original paper, Mrázek and Weickert [263] introduce two parameters for rotational invariance. A parameter q steers the contribution of diagonal and axial discretisations outside the diffusivity, and a weight c controls the contribution of w_{xy}^2 to the diffusivity argument. Our choices here correspond to $q = 1$ and $c = 2$. However, we found that one can link these two parameters by means of the diffusion discretisation of Weickert et al. [379]. For more information this alternative parameterisation, we refer to Appendix A.

Mrázek and Weickert [263] have shown that one explicit time step of nonlinear diffusion with diffusivity g is equivalent to coupled Haar wavelet shrinkage (4.13). Thus, we can directly translate existing diffusivities into shrinkage functions.

4.2 ADAPTIVE WAVELET SHRINKAGE

In our model, we equip the two-dimensional coupled Haar wavelet shrinkage approach from [263] with a trainable adaptive shrinkage function.

To this end, we make use of the connection between nonlinear diffusion and Haar wavelet shrinkage to translate a diffusivity into a shrinkage function. The resulting function differs from the traditional choices in several aspects. Instead of enforcing a binary decision, the shrinkage function is smooth, which allows to work within a smooth optimisation framework. Moreover, the function uses a separate set of parameters for each scale of the wavelet transformation, providing an individual adaptation. Finally, it is able to amplify coefficients, thus enhancing important image structures such as edges. Nevertheless, the shrinkage function is tightly parametrised. This lets us manually inspect the learned parameters and infer a smooth connection between them.

Our model builds on the shift-invariant two-dimensional Haar wavelet shrinkage as described in Section 4.1. The crucial differ-

ence is that we employ the shrinkage function $S_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(L)})^\top$ for each scale to the corresponding wavelet coefficients. Coefficients on scale ℓ are modified component-wise by $\boldsymbol{S}_{\boldsymbol{\theta}^{(\ell)}}$ according to the coupled shrinkage rule (4.13). This yields modified wavelet coefficients which are coupled:

$$\begin{aligned} \boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{W}\boldsymbol{f}) = \Big(\boldsymbol{S}_{\boldsymbol{\theta}^{(1)}}\left(\boldsymbol{w}_x^{(1)}, \boldsymbol{w}_y^{(1)}, \boldsymbol{w}_{xy}^{(1)}\right), \boldsymbol{S}_{\boldsymbol{\theta}^{(2)}}\left(\boldsymbol{w}_x^{(2)}, \boldsymbol{w}_y^{(2)}, \boldsymbol{w}_{xy}^{(2)}\right), \\ \dots, \boldsymbol{S}_{\boldsymbol{\theta}^{(L)}}\left(\boldsymbol{w}_x^{(L)}, \boldsymbol{w}_y^{(L)}, \boldsymbol{w}_{xy}^{(L)}\right), \boldsymbol{w}^{(L)} \Big)^\top . \end{aligned} \tag{4.14}$$

Moreover, instead of scaling only the backward transformation by $\frac{1}{4^\ell}$ depending on the scale, we evenly distribute it by scaling the forward and the backward transformation by $\frac{1}{2^\ell}$. This comes down to rescaling the wavelet basis, which has a beneficial effect: It guarantees that the wavelet coefficients live in the same range, regardless of the scale ℓ that they represent. With this modification, we can easily compare the learned shrinkage functions over the levels.

With these redefined forward and backward transformations $\boldsymbol{W}$ and $\widetilde{\boldsymbol{W}}$ we obtain the reconstruction $\boldsymbol{u}$ by applying the backward transformation $\widetilde{\boldsymbol{W}}$ to the modified set of wavelet coefficients and the unaltered scaling coefficients:

$$\boldsymbol{u} = \widetilde{\boldsymbol{W}} \boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{W}\boldsymbol{f}) . \tag{4.15}$$

CHOICE OF SHRINKAGE FUNCTION We found the forward-and-backward (FAB) diffusivity of Smolka [341] to be a good candidate for modelling our shrinkage function. It uses two contrast parameters λ_1 and λ_2 that control the amount of forward and backward diffusion. The diffusivity is given by

$$g(s^2) = 2 \exp\left(\frac{-s^2}{\lambda_1^2}\right) - \exp\left(\frac{-s^2}{\lambda_2^2}\right), \quad \lambda_2 \geq \lambda_1. \tag{4.16}$$

It is translated into the shrinkage function according to the coupled shrinkage rule (4.13).

For the extreme case of $\lambda_1 = \lambda_2$, the diffusivity simplifies to the exponential Perona-Malik diffusivity [279], corresponding to pure shrinkage. For larger differences between λ_2 and λ_1, the backward diffusion becomes more pronounced and the shrinkage function damps small coefficients and amplifies larger ones. Figure 4.2 visualises this behaviour.

LEARNING FRAMEWORK To train the shrinkage functions, we add additive white Gaussian noise of standard deviation σ to a database of ground truth images $(\boldsymbol{v}_i)_{i=1}^n$. This yields noisy images $\boldsymbol{f}_i$ from which we compute denoised results $\boldsymbol{u}_i$. The trainable parameters for the proposed shrinkage function on scale ℓ are given by $\boldsymbol{\theta}^{(\ell)} = \left(\lambda_1^{(\ell)}, \lambda_2^{(\ell)}\right)$.

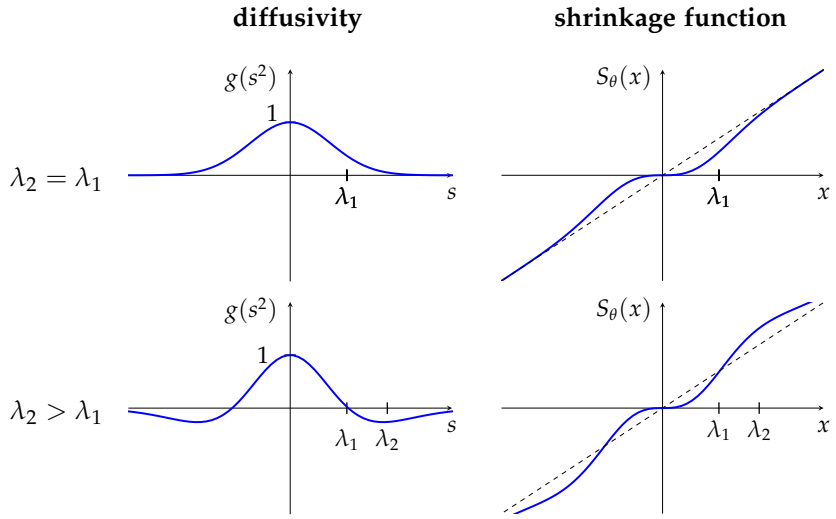


Figure 4.2: Effect of parameter choices on the FAB diffusivity and its corresponding shrinkage function. The amount of backward diffusion is determined by the difference between λ_1 and λ_2. The larger the difference, the stronger the image enhancement effect. The corresponding shrinkage function amplifies wavelet coefficients.

As an objective function, we choose the mean square error between the reconstruction $\boldsymbol{u}_i$ and the corresponding ground truth image $\boldsymbol{v}_i$, averaged over all image pairs and normalised by the number of pixels:

$$\mathcal{L}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\|\boldsymbol{u}_i - \boldsymbol{v}_i\|_2^2}{2^{2L}} \,. \tag{4.17}$$

The parameters are optimised with the gradient-based L-BFGS algorithm [236]. To that end, we compute the gradients of the objective function w.r.t. all trainable parameters.

In a first step, we obtain as a gradient of the loss function

$$\frac{\partial \mathcal{L}(\boldsymbol{u}, \boldsymbol{v})}{\partial \boldsymbol{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{\theta}} \frac{2\,(\boldsymbol{u}_i - \boldsymbol{v}_i)}{2^{2L}} \,. \tag{4.18}$$

The partial derivative $\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{\theta}}$ describes the gradient of the shrinkage equation (4.15) w.r.t. the parameters and can be computed as

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\theta}} = \frac{\partial \widetilde{\boldsymbol{W}} \boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{W} \boldsymbol{f})}{\partial \boldsymbol{\theta}} = \frac{\partial \boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{W} \boldsymbol{f})}{\partial \boldsymbol{\theta}} \widetilde{\boldsymbol{W}}^\top \,. \tag{4.19}$$

For the sake of readability, we leave out the image index i. Note that we use denominator notation, yielding the transposed backward transformation on the right hand side due to the chain rule.

Both $\boldsymbol{S}_{\boldsymbol{\theta}}(\boldsymbol{W}\boldsymbol{f})$ as well as $\boldsymbol{\theta}$ are vectors, giving a matrix for the first part of above expression. Taking the derivative for a specific $\lambda_j^{(\ell)}$ yields one row of this matrix as

$$\begin{aligned}\frac{\partial S_{\boldsymbol{\theta}}\left(\boldsymbol{W}\boldsymbol{f}\right)}{\partial \lambda_j^{(\ell)}} &= \frac{\partial}{\partial \lambda_j^{(\ell)}} \Big(\boldsymbol{S}_{\boldsymbol{\theta}^{(1)}}(\cdot), \dots, \\ &\qquad \boldsymbol{S}_{\boldsymbol{\theta}^{(\ell-1)}}(\cdot), \boldsymbol{S}_{\boldsymbol{\theta}^{(\ell)}}(\cdot), \boldsymbol{S}_{\boldsymbol{\theta}^{(\ell+1)}}(\cdot), \\ &\qquad \dots, \boldsymbol{S}_{\boldsymbol{\theta}^{(L)}}(\cdot) \Big)^\top \\ &= \left(\mathbf{0}, \dots, \mathbf{0}, \frac{\partial \boldsymbol{S}_{\boldsymbol{\theta}^{(\ell)}}(\cdot)}{\partial \lambda_j^{(\ell)}}, \mathbf{0}, \dots, \mathbf{0} \right)^\top .\end{aligned} \tag{4.20}$$

Here, we abbreviated the arguments of the shrinkage functions for the sake of readability. The specific ordering of the rows depends on the ordering of parameters in $\boldsymbol{\theta}$, as each parameter defines a row. Thus, what is left is to compute the derivative of the shrinkage function w.r.t. its parameters. These derivatives for a vector $\boldsymbol{w}^{(\ell)} = \left(w_x^{(\ell)}, w_y^{(\ell)}, w_{xy}^{(\ell)}\right)^\top$ of wavelet coefficients at a specific position are given by

$$\frac{\partial \boldsymbol{S}_{\theta^{(\ell)}}\left(\boldsymbol{w}^{(\ell)}\right)}{\partial \lambda_1^{(\ell)}} = -\frac{4\,s^2}{\left(\lambda_1^{(\ell)}\right)^3} \exp\left(-\frac{s^2}{\left(\lambda_1^{(\ell)}\right)^2}\right) \boldsymbol{w}^{(\ell)}, \tag{4.21}$$

$$\frac{\partial \boldsymbol{S}_{\theta^{(\ell)}}\left(\boldsymbol{w}^{(\ell)}\right)}{\partial \lambda_2^{(\ell)}} = \frac{2\,s^2}{\left(\lambda_2^{(\ell)}\right)^3} \exp\left(-\frac{s^2}{\left(\lambda_2^{(\ell)}\right)^2}\right) \boldsymbol{w}^{(\ell)}, \tag{4.22}$$

where $s^2 = w_x^2 + w_y^2 + 2\,w_{xy}^2$ by the coupled shrinkage rule (4.13).

The resulting gradient information is fed into the L-BFGS algorithm and determines the parameter update step. Already in this shallow model, the computation of gradients is very cumbersome. This exemplary derivation shows why one should highly appreciate automatic differentiation frameworks.

4.3 EXPERIMENTS

In our experimental setup, we use 400 images from the BSDS500 database [20] as a training set and the 68 images introduced in [308] as a test set. Their grey values are in $[0, 255]$ and the grid size is set to $h = 1$. From each image we select random regions of size 256×256, i.e. the number of scales is $L = 8$. All images are corrupted by additive white Gaussian noise of standard deviation σ. We do not clamp the resulting pixel grey values to the original grey value range to preserve the Gaussian statistics of the noise. We have found the learned parameters to be robust w.r.t. any reasonable initialisation, so no pretraining is performed.

EVALUATION OF THE LEARNED SHRINKAGE FUNCTIONS In a first experiment, we train the adaptive shrinkage function for $\sigma = 25$. The top right quadrants of the learned shrinkage functions for different scales are presented in Figure 4.3.

On the first and finest wavelet scale, all coefficients are shrunken. On the second scale, we observe coefficient amplification. We presume that this compensates the loss of image details caused by shrinking coefficients on the first scale. All further scales do not perform significant shrinkage as the learned function approaches the identity.

When we increase the noise level to $\sigma = 50$, we observe that more scales are involved in the shrinkage process. The trained shrinkage functions are displayed in Figure 4.4. Both configurations are in line with our conjecture that the diffusivity should change smoothly over the scales and the noise levels. Shrinkage and amplification decrease for coarser scales, i.e. $\lambda_1^{(\ell)}$ and $\lambda_2^{(\ell)}$ tend to zero.

With increasing noise, shrinkage and amplification become stronger and affect more scales. This is highlighted in Figure 4.5 where we display the average PSNR of reconstructions on the test set in dependence of how many scales are involved in the shrinkage process. Starting with the finest scale, gradually performing shrinkage on coarser scales improves the reconstruction quality. For higher noise levels, coarser scales become more important. This comparison also stresses again that an individual shrinkage on each scales is important as shrinkage on coarse scales is undesirable.

ABLATION STUDY To investigate the effectiveness of different aspects of our model, we perform the following ablation study. We start with classical hard wavelet shrinkage and equip it with the non-decimating wavelet transformation and the coupled shrinkage rule to enable a fair comparison. For $\sigma = 20$, we obtain an average PSNR on the test set of 27.88 dB.

In a second step, we use the proposed shrinkage function restricted to $\lambda_2 = \lambda_1$, so no amplification can take place. Still, the shrinkage function does not adapt to the individual scales. This yields a comparable PSNR of 27.83 dB, showing that smoothness of the shrinkage function alone does not matter for reconstruction quality.

When we remove the restriction on the shrinkage function, the PSNR increases to 28.06 dB which indicates that the amplification of wavelet coefficients is helpful for a good reconstruction.

Finally, introducing adaptivity to the scales boosts the PSNR to 28.55 dB, demonstrating that the scale dynamic is the crucial component for a good denoising result.

FINDING A GENERALISED SHRINKAGE FUNCTION So far, the contrast parameters are trained for each pair of scale ℓ and noise level σ from which we will now infer a generic relation. Figure 4.6 shows the

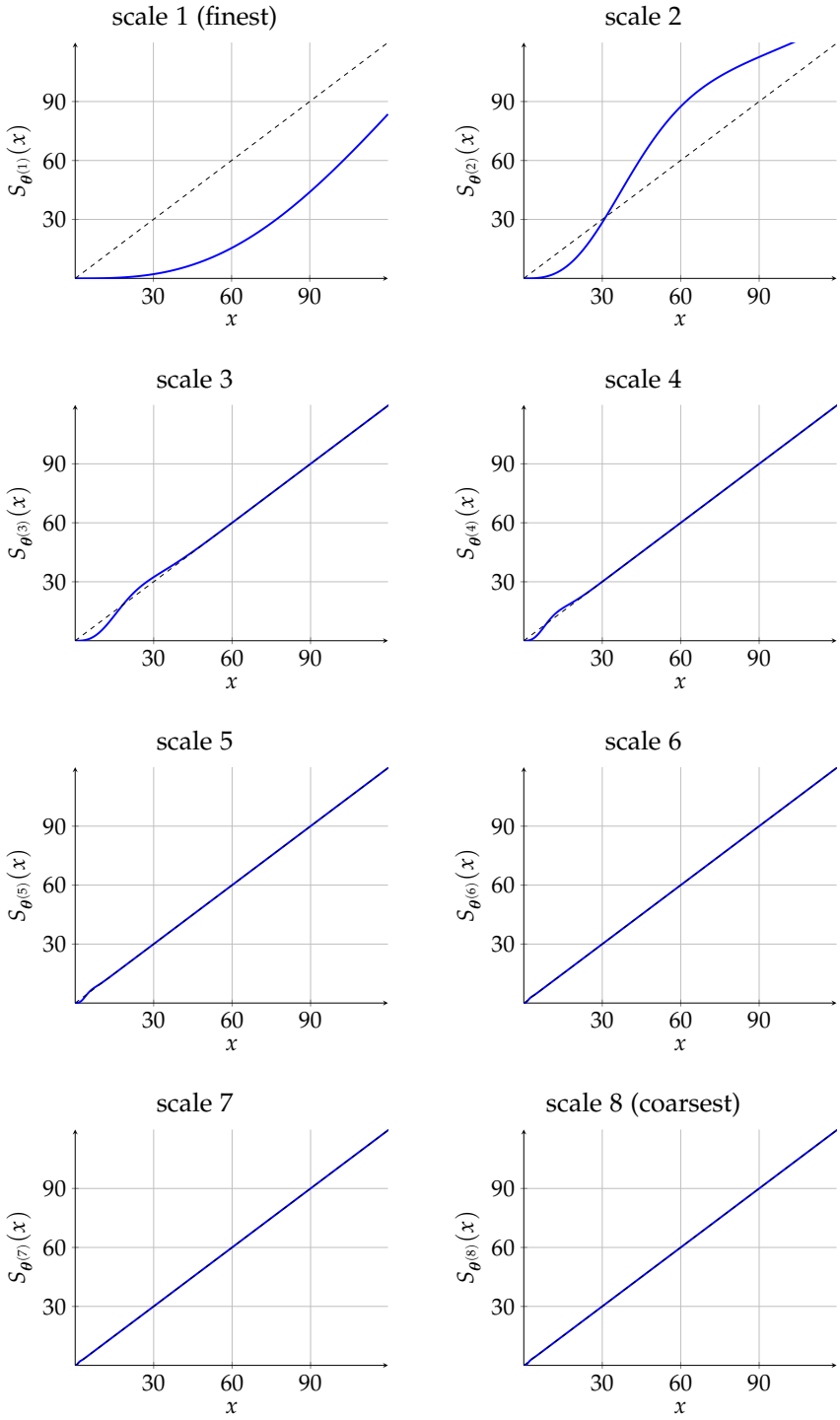


Figure 4.3: Trained shrinkage functions for $\sigma = 25$. On fine scales, both shrinkage and amplification are performed. The coarser the scale, the less wavelet coefficients are modified.

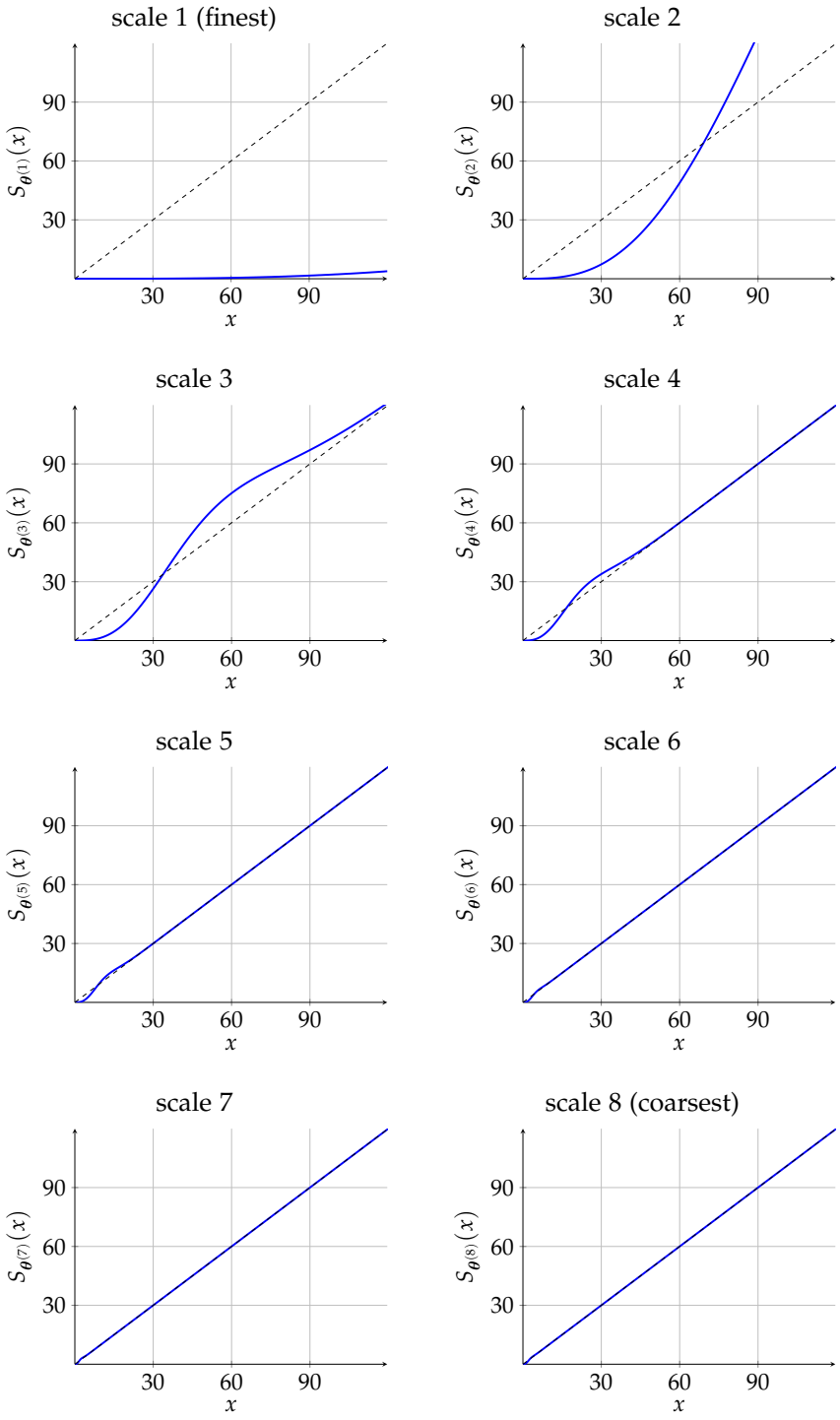


Figure 4.4: Trained shrinkage functions for $\sigma = 50$. With increasing noise, shrinkage becomes more drastic and more scales are involved.

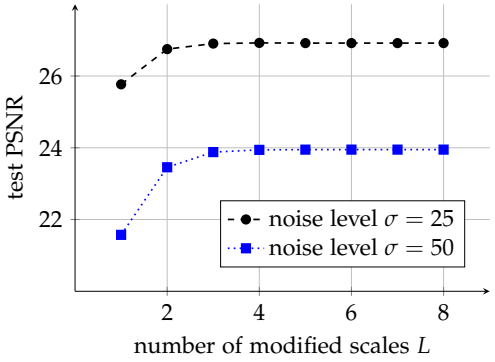


Figure 4.5: Contribution of shrunken scales to the reconstruction quality. Shrinkage should be performed on fine scales, and coarse scales should not be modified.

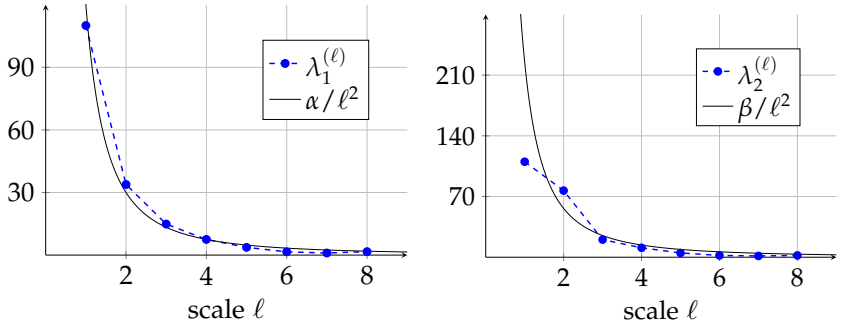


Figure 4.6: Relations between trained parameters $\lambda_1^{(\ell)}$ (left), $\lambda_2^{(\ell)}$ (right) and the shrinkage scale ℓ for $\sigma = 25$. Both parameters decrease with coarser scales with $\frac{1}{\ell^2}$.

evolution of both contrast parameters over the scales. A function of type $\frac{\alpha}{\ell^2}$ with an appropriate scalar α can provide a good description of the scale dependence of $\lambda_1^{(\ell)}$. For $\lambda_2^{(\ell)}$, the values on fine scales do not follow this relation. However, in these cases all relevant coefficients are already covered by shrinkage through a large λ_1, making the choice of λ_2 irrelevant.

Regarding the relationship between the shrinkage functions and the noise standard deviation it was already noted in [175] that a simple rescaling of shrinkage functions is sufficient for adapting to a new noise level. For our parametrisation, rescaling the complete shrinkage function is equivalent to rescaling both λ_1 and λ_2. In Figure 4.7 we can see that indeed such a rescaling is learned.

These two insights suggest that a suitable generalisation of the shrinkage function parameters which is smooth over the scales ℓ and the noise standard deviation σ is given by $\lambda_1(\ell, \sigma) = \frac{\alpha\sigma}{\ell^2}$ and $\lambda_2(\ell, \sigma) = \frac{\beta\sigma}{\ell^2}$ where α and β are scalars that have yet to be determined.

To empirically show that this parametrisation indeed captures the underlying relations in a reasonable way, we compare two models: One model trains the proposed shrinkage function for each pair (ℓ, σ)

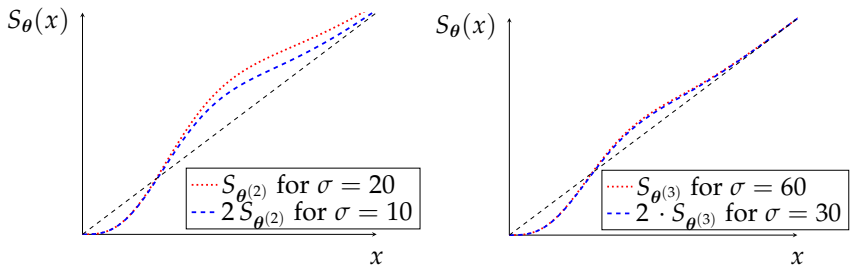


Figure 4.7: Relations between trained parameters $\lambda_1^{(\ell)}, \lambda_2^{(\ell)}$ and noise level σ. Approximately, both parameters increase linearly with higher noise levels.

individually, while the other one only optimises the factors α and β of the generalised parameters. To ensure a fair comparison, both models are trained on a new training and test set combining images with noise levels between $\sigma = 10$ and $\sigma = 60$ in steps of 2.5.

Indeed, the generic model performs only 0.3 dB worse than the model with individual parameters in terms of PSNR, while training only 2 instead of 336 parameters. With this result we conclude that the generic shrinkage function captures the adaptivity to scales and noise levels well. The scalars are learned as $\alpha = 5.4$ and $\beta = 8.9$, yielding a combined generic coupled shrinkage function (4.13) with diffusivity

$$g(s^2, \ell, \sigma) = 2 \exp\left(\frac{-s^2}{\left(\frac{5.4\,\sigma}{\ell^2}\right)^2}\right) - \exp\left(\frac{-s^2}{\left(\frac{8.9\,\sigma}{\ell^2}\right)^2}\right). \tag{4.23}$$

COMPARISON TO CLASSICAL SHRINKAGE Lastly, we compare our generic shrinkage function to soft, hard, and garrote shrinkage over a range of noise levels. We optimise the threshold parameter of the classical functions individually for each noise level, while the generic function is used as is from (4.23). The results are displayed in Figure 4.8. Although the classical approaches are optimised for each noise level, they are inferior to the generic shrinkage function. Over the range of noise levels used for training, improvements of up to 0.65 dB with an average of 0.34 dB are obtained compared to the best classical approaches.

For two exemplary noise levels of $\sigma = 25$ and $\sigma = 50$, Figures 4.9 and 4.10 show reconstructions along with the noisy input and the ground truth image. In both cases, soft shrinkage blurs images too strongly since all wavelet coefficients are decreased by the same margin. Hard shrinkage suffers from remaining noise as it does not shrink large noisy wavelet coefficients. While less pronounced, this is also the case for garrote shrinkage. Both garrote and hard shrinkage also blur important image structures. Our generic shrinkage function outperforms all classical approaches. By strongly shrinking coefficients on fine scales, noise is efficiently removed. To compensate for lost image

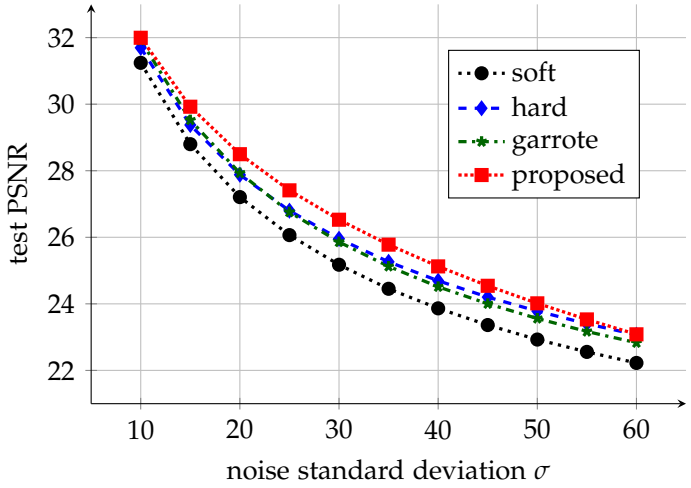


Figure 4.8: Comparison of PSNR values on the test set (higher is better) for individually optimised classical approaches and our generic shrinkage function. The proposed function outperforms all classical shrinkage functions.

details, amplification of wavelet coefficients on coarser scales enhances important structures.

4.4 CONCLUSIONS

Our approach of learning a compact shrinkage function for wavelet denoising combines the advantages of model-driven and data-driven approaches: In contrast to other parameter learning strategies, we can cope with as little as two parameters without substantially sacrificing performance. This results in an interpretable shrinkage function and a transparent, but adaptive model.

This is the first example of how learning can be used to improve classical models. In the following chapter, we employ a similar strategy to come up with a better denoising model by learning scale-adaptive diffusivities.

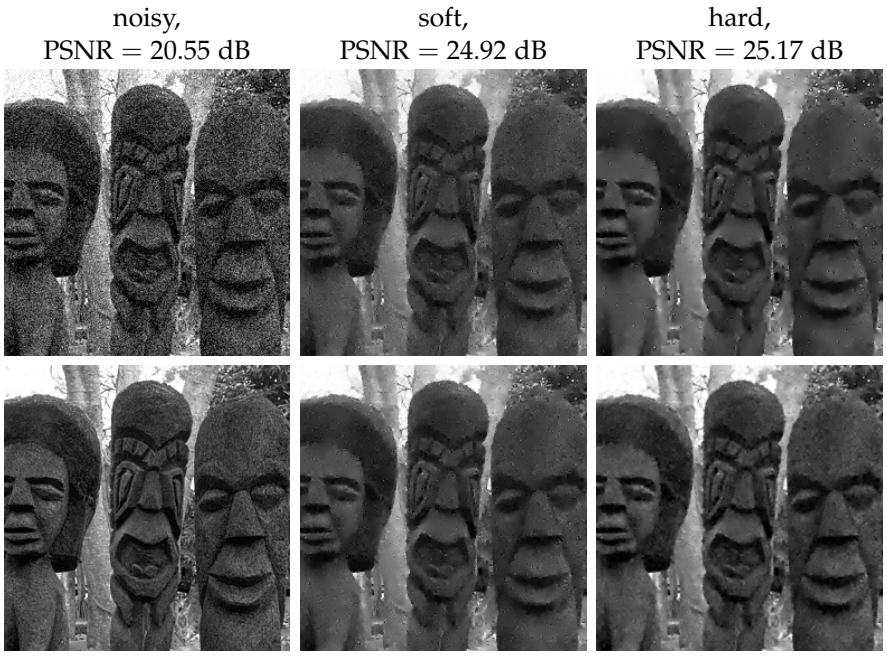


Figure 4.9: Visual comparison of reconstructions for classical and proposed shrinkage functions for $\sigma = 25$. The proposed function achieves a better balance between blur of important image structures and noise removal.

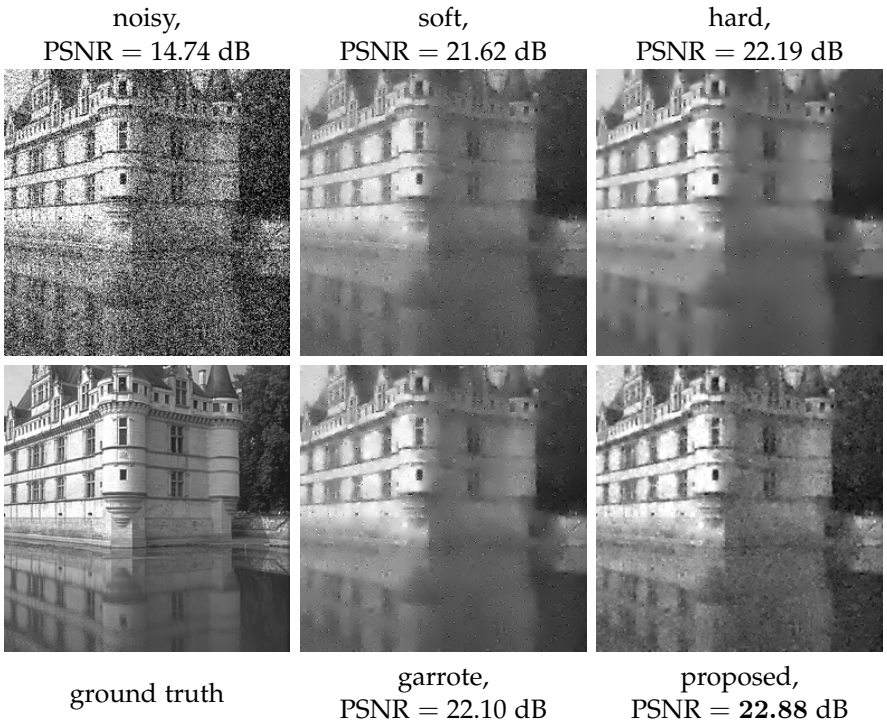


Figure 4.10: Additional visual comparison for $\sigma = 50$. For higher noise levels, the advantages of the proposed function become more pronounced as the adaptive shrinkage affects more scales.

5 TRAINABLE INTEGRODIFFERENTIAL DIFFUSION

Building on the experiences from the previous chapter, we can now transfer the model-based learning approach to diffusion models. This endeavour is more complex in comparison to the wavelet setting in several aspects: First and foremost, the notion of scales is not defined as clearly as for wavelets. While for wavelet shrinkage scales are defined as the powers of two, a continuous diffusion model can work on any positive real-valued scale. As traditional diffusion models work on a single — often the finest — scale, we extend these models by means of integrodifferential equations.

Moreover, instead of the three-step procedure from wavelet shrinkage, temporal discretisations of diffusion models require several iterations to capture nonlinear behaviour. This also requires to come up with stability guarantees for the discrete iterations of the model.

As in the previous chapter, we are aiming at the best of two worlds: We benefit from the compactness and mathematical foundations of PDE-inspired modelling, while we improve its performance by learning a small set of parameters. In contrast to most other approaches, however, we compress the model further by understanding the functional dependence of these parameters. In other words: *We drive learning-based modelling to the extreme by aiming at the most compact and insightful model.* In this chapter, we focus on image denoising with edge-enhancing anisotropic diffusion (EED) [368] as an exemplary application.

We propose a multiscale extension of EED based on a transparent integrodifferential formulation. Our integrodifferential anisotropic diffusion (IAD) model adapts to image structures by combining edge information from multiple scales and steering the diffusion process accordingly.

Given a set of images corrupted by Gaussian noise, we learn scale-adaptive parameters of the diffusion process by optimising the denoising performance. Afterwards, we explore the trained parameters and reduce them by explicitly modelling the parameter dynamics over the scales and the noise level. We show that this is possible without decreasing the denoising performance significantly.

The resulting IAD model outperforms both its counterpart of integrodifferential isotropic diffusion (IID), as well as EED, while only requiring one additional parameter. In ablation studies, we show that both multiscale information and anisotropy are crucial for its success.

Lastly, we investigate the potential of IAD for sparse image inpainting and provide an outlook on possible improvements of the model.

RELATED WORK While PDEs are omnipresent in image analysis, models that explicitly involve the more general integrodifferential equations are surprisingly rare. They are considered occasionally for image decompositions [25], nonlocal generalisations of PDE evolutions [63, 146], and in connection with fractional calculus for linear image processing [88]. These works are only very mildly related to the work in this chapter.

Since our IAD model sums up contributions from multiple scales, it has some conceptual similarities to wavelet shrinkage. Didas and Weickert [100] relate wavelet shrinkage to nonlinear diffusion, but only for the isotropic case. As each scale is treated independently, anisotropy cannot be achieved. Welk et al. [387] present an anisotropic diffusion method based on wavelet shrinkage, but only on the finest scale. We combine the best of these two ideas into a novel model: Our IAD model combines structural image information over multiple scales to create an anisotropic diffusion process.

Integrodifferential models for nonlinear diffusion predominantly involve models with Gaussian-smoothed derivatives. Most of these models [65, 269, 326, 368] have been proposed as regularisations of the Perona–Malik filter [279]. For enhancing coherent structures, one also considers a smoothed structure tensor [127] that captures directional information to steer the diffusion process accordingly [369]. However, all of these models only consider Gaussian smoothing on a fixed scale and do not incorporate an integration over multiple scales.

Large-scale trainable models involving PDEs have recently become very successful. Chen and Pock [75] train flux functions and derivative filters of a diffusion-inspired model to obtain exceptional denoising results. Other authors only train nonlinearities of models [13, 32, 331] or learn PDEs directly with symbolic approaches [238, 291, 324]. Instead of focusing on denoising performance, we want to obtain insights into the scale behaviour of our model. In contrast to [75], we have stability guarantees in the Euclidean norm and require very few parameters that allow insights into their functional dependency. This helps us to obtain a model which is transparent and outperforms its diffusion-based predecessors.

PUBLICATION INFORMATION This chapter is based on the conference paper of Alt and Weickert [14] presented at the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing. It is extended with more information on the discrete model and its regularisation, along with a refined discretisation and a stability analysis. Furthermore, we provide a deeper insight into the process of

reducing the learned parameters, our ablation studies, and we overhauled the function fittings. Moreover, we exchanged Perona–Malik diffusion [279] for the model of Catté et al. [65] as a competitor, as it is the natural competitor to IID. We enriched the experimental section with a finer sampling of scales, as well as analysis of the eigenvalues of the learned multiscale structure tensor. Lastly, we extend the IAD model to the image inpainting setting and compare its performance against that of EED.

ORGANISATION OF THE CHAPTER In Section 5.1, we review a reformulation of EED. This serves as a basis for our novel IAD model that is introduced in Section 5.2. In several experiments in Section 5.3, we reduce its parameters and compare it to other diffusion filters. We extend our considerations to the task of sparse image inpainting in Section 5.4. Finally, we present our conclusions and give an outlook on future work in Section 5.5.

5.1 USEFUL REFORMULATION OF EED

In Section 3.1.1, we have introduced the EED model of Weickert [368]. As a reminder, EED embeds a greyscale image $f(x,y) : \Omega \to \mathbb{R}$ into a family of simplified versions $\{u(x,y,t) \mid t \geq 0\}$ by solving the diffusion equation

$$\partial_t u = \nabla^\top (\boldsymbol{D}_\lambda(\nabla_\sigma u)\, \nabla u)\,. \tag{5.1}$$

The process is initialised with $u(x,y,0) = f(x,y)$, and on the domain boundary $\partial\Omega$ one uses reflecting boundary conditions.

The diffusion tensor is defined by its eigenvalues $g_\lambda\left(|\nabla_\sigma u|^2\right), 1$ and normalised eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ as

$$\boldsymbol{D}_\lambda(\nabla_\sigma u) = g_\lambda\left(|\nabla_\sigma u|^2\right)\, \boldsymbol{v}_1 \boldsymbol{v}_1^\top + 1\, \boldsymbol{v}_2 \boldsymbol{v}_2^\top, \tag{5.2}$$

allowing diffusion along edges, but inhibiting smoothing across them. Here, we explicitly denote the contrast parameter λ in the diffusivity g_λ and the diffusion tensor $\boldsymbol{D}_\lambda$.

To prepare ourselves for our novel integrodifferential model, we reformulate the EED model with the structure tensor [127]

$$\boldsymbol{J} = \nabla_\sigma u \nabla_\sigma^\top u\,. \tag{5.3}$$

By plugging in, we see that the matrix $\boldsymbol{J}$ has eigenvectors $\boldsymbol{v}_1 \parallel \nabla_\sigma u$ and $\boldsymbol{v}_2 \perp \nabla_\sigma u$ with eigenvalues $\nu_1 = |\nabla_\sigma u|^2$ and $\nu_2 = 0$.

Expressing the diffusivity function g_λ in terms of a power series, we can generalise it to matrix-valued arguments such as $\boldsymbol{J}$. Then $g_\lambda(\boldsymbol{J})$ is a matrix as well. It has the same eigenvectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ as $\boldsymbol{J}$, and its eigenvalues satisfy

$$\lambda_1 = g_\lambda(\nu_1) = g_\lambda\left(|\nabla_\sigma u|^2\right)\,, \tag{5.4}$$

$$\lambda_2 = g_\lambda(\nu_2) = g_\lambda(0) = 1\,. \tag{5.5}$$

Thus, we have $\boldsymbol{D}_\lambda(\boldsymbol{\nabla}_\sigma u) = g_\lambda(\boldsymbol{J})$ and can write the EED evolution as

$$\partial_t u = \boldsymbol{\nabla}^\top (g_\lambda(\boldsymbol{J})\, \boldsymbol{\nabla} u) \,. \tag{5.6}$$

It closely resembles the regularised nonlinear isotropic diffusion model of Catté et al. [65] (see also Section 3.1.1):

$$\partial_t u = \boldsymbol{\nabla}^\top \left(g_\lambda\left(|\boldsymbol{\nabla}_\sigma u|^2 \right) \boldsymbol{\nabla} u \right). \tag{5.7}$$

Indeed, by replacing the matrix-valued tensor product $\boldsymbol{J} = \boldsymbol{\nabla}_\sigma u \boldsymbol{\nabla}_\sigma^\top u$ by the scalar-valued inner product $\boldsymbol{\nabla}_\sigma^\top u \boldsymbol{\nabla}_\sigma u = |\boldsymbol{\nabla}_\sigma u|^2$, we end up with the Catté model. Thus, there is an elegant connection between isotropic and anisotropic models.

The choice of the scale σ is crucial for the denoising performance of EED. A single scale may be too coarse for fine-scale details, while at the same time not capturing large-scale structures. We will see that by accumulating information over multiple scales within the structure tensor, we obtain a more useful representation of image information which again leads to improved denoising results.

5.2 INTEGRODIFFERENTIAL DIFFUSION

CONTINUOUS MODEL To account for multiscale information within the structure tensor, we introduce the following *integrodifferential anisotropic diffusion (IAD)* model:

$$\partial_t u = \int_0^\infty \boldsymbol{\nabla}_{\sigma,\gamma}^\top \left(g_\lambda(\boldsymbol{J}_\gamma)\ \boldsymbol{\nabla}_{\sigma,\gamma} u \right) d\sigma \,, \tag{5.8}$$

where the *multiscale structure tensor*

$$\boldsymbol{J}_\gamma = \int_0^\infty \boldsymbol{\nabla}_{\sigma,\gamma} u\, \boldsymbol{\nabla}_{\sigma,\gamma}^\top u \; d\sigma \tag{5.9}$$

accumulates structural information over all smoothing scales σ. This integration generates anisotropy on each scale, since the eigenvectors of $\boldsymbol{J}_\gamma$ usually are not parallel to $\boldsymbol{\nabla}_{\sigma,\gamma} u$.

To adapt the contribution of each scale, we introduce an additional weight parameter $\gamma(\sigma)$ in the smoothed gradient, i.e.

$$\boldsymbol{\nabla}_{\sigma,\gamma} u = \boldsymbol{\nabla} \left(\gamma(\sigma) K_\sigma * u \right) . \tag{5.10}$$

In contrast to other models employing an outer smoothing with a fixed weighting [310, 373], the IAD model allows to attenuate scales if they do not offer useful information. This weighting, along with the multiscale structure tensor, is the key to the success of the IAD model.

The diffusion tensor $g_\lambda(\boldsymbol{J}_\gamma)$ inherits its eigenvectors from $\boldsymbol{J}_\gamma$, and its eigenvalues are given by $g_\lambda(\nu_1)$ and $g_\lambda(\nu_2)$ where ν_1, ν_2 are the eigenvalues of $\boldsymbol{J}_\gamma$. In contrast to EED, both eigenvalues may be different from one. This makes a difference for corner regions where

$\nu_1 \geq \nu_2 \gg 0$. Such penalisations have been considered by Weickert and Brox [373] and Peter et al. [284].

Moreover, we make the contrast parameter $\lambda(\sigma)$ of the diffusivity scale-adaptive. This individually steers the balance between smoothing and edge enhancement on each scale.

Despite smoothing the gradient operators outside of the diffusion tensor, the IAD model is still anisotropic as it uses a multiscale structure tensor. Due to the integration, its eigenvectors do not coincide with the smoothed gradient operators. While related ideas have been used in the context of multiple image channels [98, 373, 376], the idea of creating anisotropy through multiscale information is novel.

An isotropic counterpart of the IAD model, which we call *integrodifferential isotropic diffusion (IID)*, arises directly by switching the order of transposition within the structure tensor. In this case,

$$\int_0^\infty \nabla_{\sigma,\gamma}^\top u \nabla_{\sigma,\gamma} u \, d\sigma = \int_0^\infty |\nabla_{\sigma,\gamma} u|^2 \, d\sigma \tag{5.11}$$

is no structure tensor any more, but a multiscale gradient magnitude. In our experiments, we use the IID model for comparisons.

TRAINABLE DISCRETE MODEL In a practical setting, we need a discrete version of the continuous IAD model. To this end, we employ an explicit finite difference scheme: We discretise the temporal derivative by a forward difference with time step size τ, and the individual integrands by the nonnegativity discretisation from [369]. Additionally, we select a set of L discrete scales $\sigma_1, \dots, \sigma_L$ according to an exponential sampling. This yields discrete weights $\gamma_\ell = \gamma(\sigma_\ell)$ and contrast parameters $\lambda_\ell = \lambda(\sigma_\ell)$, which we obtain by training.

We end up with the explicit scheme

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \tau \sum_{\ell=1}^{L} \omega_\ell \boldsymbol{K}_\ell^\top \left(g_{\lambda_\ell}\left(\boldsymbol{J}_\gamma^k \right) \boldsymbol{K}_\ell \boldsymbol{u}^k \right), \tag{5.12}$$

where $\boldsymbol{K}_\ell$ is a discretisation of the scaled smoothed gradient $\nabla_{\sigma,\gamma}$. The discrete multiscale structure tensor $\boldsymbol{J}_\gamma^k$ is computed as

$$\boldsymbol{J}_\gamma^k = \sum_{m=1}^{L} \omega_m \left(\boldsymbol{K}_m \boldsymbol{u}^k \right)^\top \left(\boldsymbol{K}_m \boldsymbol{u}^k \right). \tag{5.13}$$

Here, we renamed the scale index to m to avoid confusion with the outer summation.

In both of the above equations, we encounter a scale-dependent constant ω_ℓ. It is a result of turning the infinitesimal integration into a discrete summation over scales. As the scales are not sampled uniformly, their individual distance must be accounted for in the summation. We decide on the simple option of $\omega_\ell = \sigma_{\ell+1} - \sigma_\ell$. Thus, we see that for $L \to \infty$ and correspondingly $\omega_\ell \to 0$, the summation consistently approximates the continuous formulation.

In our publication [14] we left out the constant ω_ℓ. However we argue that both models have the same capacity, as an adaption of the parameters γ_ℓ can subsume ω_ℓ. Still, in the interest of cleanly separating the discretisation of the model and its nonlinear dynamics, the explicit introduction of ω_ℓ should be preferred.

As we discuss in Chapter 8 and prove in Appendix C, this scheme is stable in the L^2 norm if the time step size obeys

$$\tau \leq \frac{2}{P \sum_{\ell=1}^{L} \omega_\ell \|\boldsymbol{K}_\ell\|_2^2}, \tag{5.14}$$

where P is the Lipschitz constant of the flux function, and $\|\cdot\|_2$ is the spectral radius. In our case, we have $P = 1$. The spectral radius of $\boldsymbol{K}_\ell$ is estimated on the fly during training with Gershgorin's circle theorem [143] (see Section 2.4).

Finally, we iterate the method for a small number of K explicit steps. This yields an approximation of the continuous model for a diffusion time of $T = K\tau$. Iterating the IAD model allows to capture a nonlinear evolution.

By design, the discrete IAD model is stable in the Euclidean norm. Moreover, since the algorithm consists of a concatenation of continuous function evaluations, it constitutes a well-posed discrete evolution. In particular, this implies that the output depends continuously on the input data. This shows that we have created a rigid mathematical framework within which we can train the desired parameters without giving up any of the mathematical guarantees.

LEARNING FRAMEWORK To train the $2L$ parameters γ_ℓ and λ_ℓ of the discrete model, we consider a training set of 200 grey value images of size 256×256 with grey value range $[0, 255]$.

As in the previous chapter, we crop them from the BSDS500 dataset [20] and corrupt them with additive Gaussian noise of standard deviation n. The resulting grey values are not cut off if they exceed the original grey value range to preserve the Gaussian statistics of the noise. A disjoint test set of 100 images is generated accordingly.

We train the model with a sufficient number of explicit steps by minimizing the average mean square error over the training set. Moreover, we always choose τ in such a way that it is stable and allows for a suitable diffusion time. We found that $K = 10$ steps are already sufficient for noise levels up to $n = 60$. Note that we rename the noise standard deviation to n in this chapter, as σ already denotes scale.

5.3 FINDING SCALE-ADAPTIVE PARAMETER FUNCTIONS

To gain insights into the behaviour of the IAD model, we develop smooth relations for the discrete trained parameters in terms of the scale σ and the noise standard deviation n.

ELIMINATING SCALE DEPENDENCY We start by investigating the dynamics of the parameters over the scales. To this end, we train the full model for $K = 10$ explicit steps and $L = 16$ discrete scales for varying noise standard deviations $n \in \{10, 15, 20, \dots, 60\}$. To ensure that the parameters $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_L)$ and $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_L)$ vary smoothly over the scales, we add a small regularisation to the optimisation loss. It penalises variations of the respective parameters over the scales in the squared Euclidean norm as

$$\mathcal{R}_1(\boldsymbol{\gamma}) = \rho_1 \sum_{\ell=1}^{L-1} \left(\frac{\gamma_{\ell+1} - \gamma_\ell}{\sigma_{\ell+1} - \sigma_\ell} \right)^2, \tag{5.15}$$

$$\mathcal{R}_2(\boldsymbol{\lambda}) = \rho_2 \sum_{\ell=1}^{L-1} \left(\frac{\lambda_{\ell+1} - \lambda_\ell}{\sigma_{\ell+1} - \sigma_\ell} \right)^2. \tag{5.16}$$

These expressions are discretisations of the continuous quadratic regularisers $(\partial_\sigma \gamma(\sigma))^2$ and $(\partial_\sigma \lambda(\sigma))^2$ by means of forward differences. Thus, they enforce smooth parameter evolutions over the scale. The regularisation parameters ρ_1, ρ_2 steer the individual amount of smoothness as the parameters cover different ranges. Empirically, we found that $\rho_1 = 6.5 \cdot 10^{-4}$ and $\rho_2 = 1.3 \cdot 10^{-3}$ are good choices for providing a degree of smoothness that allows to fit continuous dynamics.

We present the resulting parameter dynamics over the scales in Figure 5.1 for a representative noise level of $n = 40$. For the weight parameters γ_ℓ, we find that the importance of the structural information decreases with the scale. We normalise the parameters γ_ℓ such that $\gamma_1 = 1$, as a rescaling of γ_ℓ can be compensated by adapting the time step size τ and the contrast parameters λ_ℓ accordingly.

The contribution of smoothing scales with $\sigma > 3.0$ is essentially non-existent. This is in accordance with what we have observed for scale-adaptive wavelet shrinkage in Chapter 4, where useful shrinkage is only performed on the finer scales.

We see that an intermediate Gaussian fit of the form

$$\gamma(\sigma, n) = \exp\left(-\frac{\sigma^2}{2(\theta_1(n))^2} \right) \tag{5.17}$$

appropriately captures the dynamics of the weight parameters. An intermediate parameter $\theta_1(n)$ serves as the standard deviation, which depends on the noise level n.

Also for the contrast parameters a similar Gaussian fit of the form

$$\lambda(\sigma, n) = \theta_2(n) \exp\left(-\frac{\sigma^2}{2(\theta_3(n))^2} \right) \tag{5.18}$$

is effective. Here, an intermediate parameter $\theta_2(n)$ determines the amplitude of the Gaussian, and $\theta_3(n)$ models the standard deviation. Both depend on the noise level n.

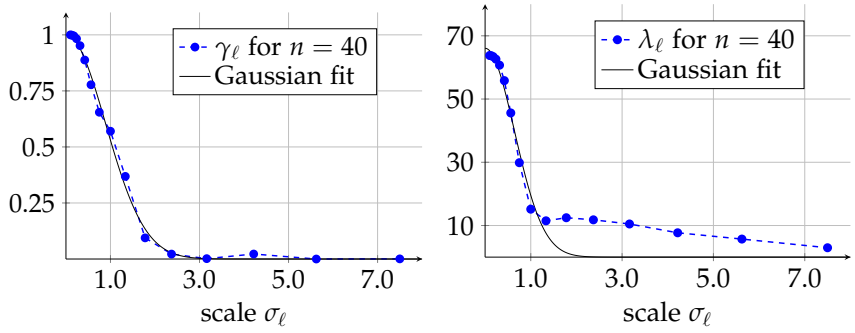


Figure 5.1: Learned weight and contrast parameters $\gamma_\ell, \lambda_\ell$ for a noise standard deviation $n = 40$.

Outliers for large scales can be ignored, as $\gamma \to 0$ dampens the influence of λ. Thus, the deviations of the contrast parameter from the fit for larger scales do not affect the performance, as we show in our ablation study.

ELIMINATING NOISE DEPENDENCY Now that we have found a function for the contrast and weight parameters over the scales, we eliminate the noise dependency of the intermediate parameters. To this end, we retrain the IAD model with the intermediate parametrisations for λ_ℓ and γ_ℓ and learn the parameters $\theta_1(n), \theta_2(n)$, and $\theta_3(n)$ over a discrete set of noise levels $n \in \{10, 15, 20, \dots, 60\}$.

Figure 5.2 presents the learned Gaussian evolutions for both parameters, along with selected function fits for the intermediate parameters.

We find that the standard deviation $\theta_1(n)$ of the Gaussian function for the weight parameters $\gamma(\sigma, n)$ can be modelled as

$$\theta_1(n) = \alpha \sqrt[4]{n}\,, \tag{5.19}$$

where α is a constant. Thus, the variance of the Gaussian grows with $\sqrt{n}$.

Larger noise levels require to put more weight on coarser scales. As the images are corrupted more strongly, larger scales are needed to extract useful structural information.

The Gaussian function for the contrast parameters $\lambda(\sigma, n)$ both requires to fit an amplitude $\theta_2(n)$ as well as a standard deviation $\theta_3(n)$.

When increasing the noise level, the contrast parameters scale linearly, a relation which we have already observed in the previous chapter. The amplitude can thus be modelled as

$$\theta_2(n) = \lambda_0\, n\,, \tag{5.20}$$

where the constant λ_0 determines the contrast parameter at $\sigma = 0$.

Interestingly, the standard deviation does not show any dependency over the noise and can be modelled by a constant

$$\theta_3(n) = \beta\,. \tag{5.21}$$

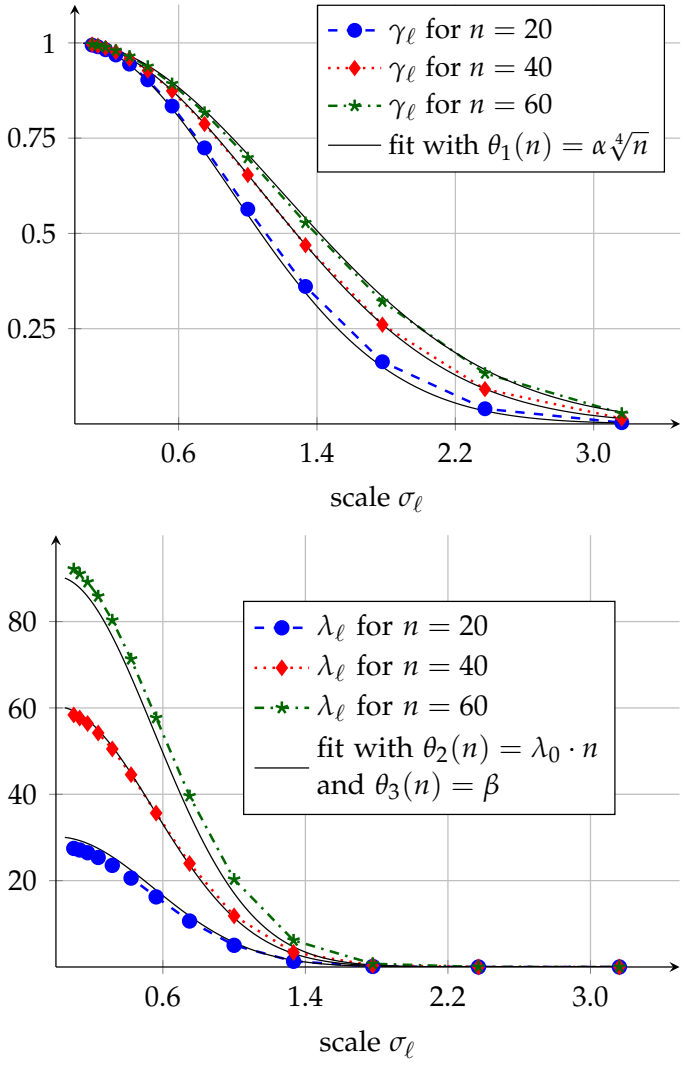


Figure 5.2: Learned intermediate parameters $\theta_1, \theta_2, \theta_3$ with fits over noise levels n. The regularised learned parameter distributions can be modelled well by a generalised continuous function.

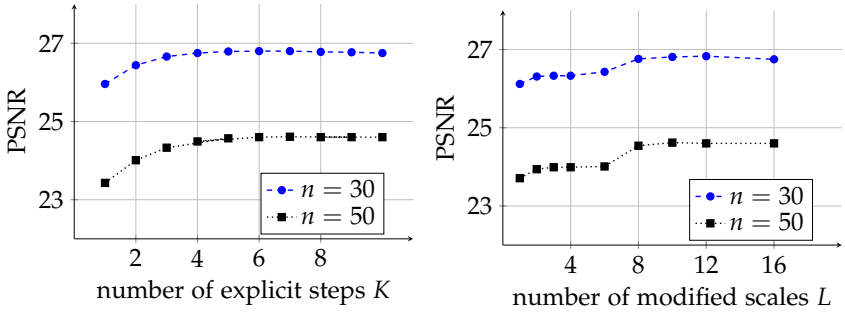


Figure 5.3: Ablation study on how many explicit steps are needed, and which scales are important for good denoising quality.

Thus, the nonlinear response of the diffusivity g_λ is the same for all noise levels, except for a linear scaling. This can be explained by the way in which weight and contrast parameters work together: The weight parameters determine the argument of the diffusivity, while the contrast parameters only determine the diffusivity response. As the weight parameters already account for the influence of noise on the data, the contrast parameters do not need to adjust the balance of scales.

This yields final parameter dynamics for the weight parameters $\gamma(\sigma, n)$ as

$$\gamma(\sigma, n) = \exp\left(-\frac{\sigma^2}{2\alpha^2\sqrt{n}}\right), \tag{5.22}$$

and for the contrast parameters $\lambda(\sigma, n)$ as

$$\lambda(\sigma, n) = \lambda_0 \, n \, \exp\left(-\frac{\sigma^2}{2\beta^2}\right). \tag{5.23}$$

Through these insights, we have effectively reduced the parameter set to only three trainable parameters α, β, λ_0, one more than for EED.

ABLATION STUDIES In a first ablation study, we show that we hardly sacrifice any performance for the sake of parameter reduction. In a first step, we train the full model for $K = 10$ explicit steps, $L = 16$ scales, and noise levels $n \in \{10, 15, 20, \ldots, 60\}$ without any smoothness regularisation. With two parameters per scale and noise level, this amounts to 352 parameters. Afterwards, we train the parameters α, β, and λ_0 of the reduced model for the same configuration jointly for all noise levels. We obtain $\alpha = 0.35$, $\beta = 0.54$ and $\lambda_0 = 1.70$.

On average, the fully trained model with 352 parameters performs only 0.03 dB better than the reduced model with three parameters. This marginal performance difference shows that the parameter relations which we have modelled represent the true parameter dynamics well.

In a second ablation study, we evaluate how many explicit steps and scales are required for practical efficiency of the model. In Fig-

ure 5.3, we display the performance of the fully trained model without regularisation for a selected number of both explicit steps and scales.

We observe that for the displayed noise levels of $n = 30$ and $n = 50$, performance improvements for more than $K = 5$ explicit steps are only marginal. This is to be expected, as for denoising already a small stopping time is often sufficient. Due to the multiscale information in IAD, the required diffusion times seem to be even smaller than those of single scale models.

The scale ablation shows an interesting behaviour. We perform diffusion only on a subset of the original 16 sampled scales, and gradually add larger scales to the set. However, the performance improvement is not linear. We can make out two jumps within both performance curves. One is at two scales, which is not surprising as this constitutes the switch between an isotropic and an anisotropic model.

However, an even larger jump can be found between 6 and 8 scales. This suggests that information on scales is not of equal importance. The scales in question that are added in this case are 0.56 and 0.75, indicating that information on these scales is valuable for good denoising. Interestingly, the EED model for $n = 50$ selects an optimal presmoothing scale of $\sigma = 0.66$. We conjecture that the IAD model in fact realises EED at its core, but additionally equips the process with additional structural information around the optimal smoothing scale. This would be very much in line with our motivation.

COMPARISON TO OTHER DIFFUSION FILTERS We compare the reduced model to the model of Catté et al. [65] as well as EED [368]. Additionally, we consider IID as the isotropic variant of the IAD model. This four-way comparison is designed in an ablative way to show that both multiscale modelling and anisotropy are crucial for the success of the IAD model: Multiscale information is not considered for EED and Catté, while anisotropy is not used for Catté and IID. We explicitly compare only to other diffusion-based methods, as it is our intention to transparently improve those compact and insightful models rather than to produce state-of-the-art results.

To ensure a fair comparison, all models are trained with the exponential Perona–Malik diffusivity. For Catté and EED, we optimise the parameters for each noise level individually. However, for IID and IAD, we respectively use the reduced parameters over the full noise range.

Figures 5.4 and 5.5 show representative denoising results on the *peppers* image for $n = 20$, and on the *cameraman* image for $n = 50$. The Catté model removes noise effectively in homogeneous areas, but leaves noisy edges. EED yields sharp edges in the foreground. However, due to the single smoothing scale, both are not able to denoise the background efficiently, where a much larger smoothing

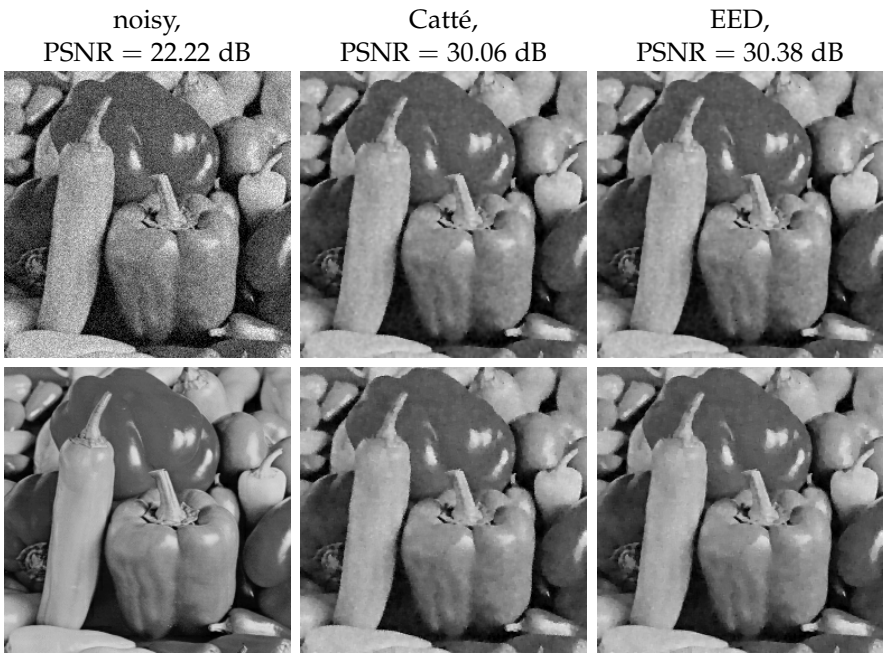


Figure 5.4: Qualitative comparison of denoising performance on the *peppers* image for $n = 20$. The multiscale information of IID and IAD helps to adapt smoothing to different scales, and the anisotropy of EED and IAD allows for directional smoothing along edges.

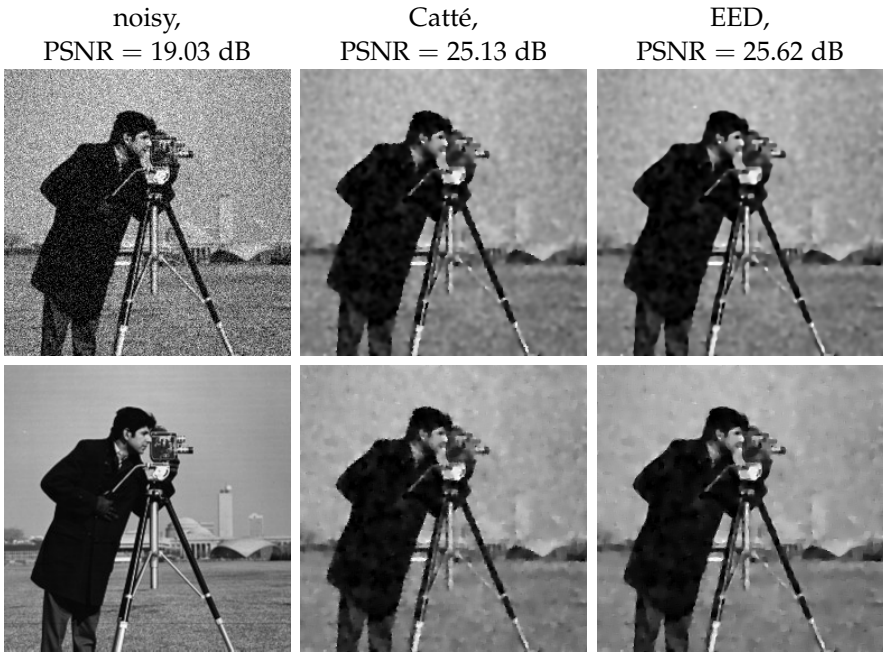
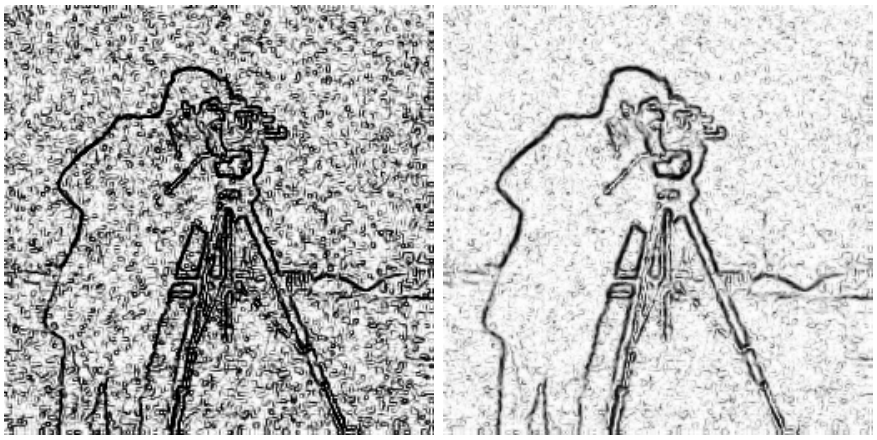


Figure 5.5: Qualitative comparison of denoising performance on the *cameraman* image for $n = 50$. As pixels are more strongly corrupted, the benefits of the multiscale information in the IAD model increase.



EED IAD

Figure 5.6: Smaller eigenvalue of the diffusion tensors for EED and the finest scale of IAD at an intermediate diffusion time of $T = 2.2$. Large grey values indicate high values, thus encouraging smoothing. The diffusion tensor of IAD captures the structural information of the ground truth image better than that of EED.

Table 5.1: Average PSNR on the test set of the four models considered. Higher values indicate better quality.

n	Catté	EED	IID	IAD
10	32.07	32.12	32.05	**32.29**
20	28.19	28.31	28.25	**28.53**
30	26.28	26.38	26.43	**26.68**
40	24.94	25.06	25.16	**25.39**
50	24.10	24.22	24.33	**24.53**
60	23.40	23.53	23.67	**23.84**

scale would be appropriate. The IID and IAD models do not suffer from this effect as the multiscale information can identify structures on all scales. However, IID suffers from remaining noise around edges. This is an intrinsic drawback of the isotropic model which detects only the location of an edge, but not its orientation. Finally, IAD combines edge enhancement together with efficient denoising in homogeneous regions.

An investigation of the diffusion tensor supports these considerations. In Figure 5.6, we display the smaller eigenvalue of the EED diffusion tensor along with the smaller eigenvalue of the IAD diffusion tensor on the finest scale. We obtain both diffusion tensors at an intermediate diffusion time of $T = 2.2$. Larger grey values indicate larger eigenvalues, leading to more smoothing. We see that IAD does not inhibit the diffusion in the background as strongly and inhomo-

geneously as EED. Edges in the IAD model are more pronounced due to the multiscale structure tensor information.

Lastly, we compare the performance of the four models on different noise levels. Table 5.1 shows the average PSNR on the test set. Note that these numbers deviate from those published in [14] due to the different fitting and a different selection of smoothing scales. However this does not fundamentally change our insights.

Surprisingly, already the IID model is able to beat EED for Gaussian noise with standard deviation $n \geq 30$. With its additional anisotropy, the IAD model consistently outperforms its competitors. With higher amounts of noise, IID and IAD increase the gap to their PDE-based predecessors, indicating that multiscale information becomes increasingly important with noise.

5.4 EXTENSION TO INPAINTING

Our denoising experiments have shown that IAD is a fruitful extension of EED for this setting. The multiscale nature helps to identify structures on multiple scales and steers the denoising process accordingly.

A natural question is whether IAD can advance EED also in other applications. For denoising EED is not the state of the art, yet it is very hard to beat in the setting of sparse image inpainting. In this section, we investigate the potential of IAD for this task.

We want to obtain a reconstruction u from an image f which is only partially available on an inpainting mask $K \subset \Omega$. To this end, we solve the inpainting problem

$$Lu = 0 \qquad \text{on } \Omega \setminus K, \tag{5.24}$$

$$u = f \qquad \text{on } K, \tag{5.25}$$

with reflecting boundary conditions, where L is the inpainting operator of choice.

We consider three models for this purpose. The IAD model uses

$$Lu = \int_0^\infty \nabla_{\sigma,\gamma}^\top \Big(g_\lambda(\boldsymbol{J}_\gamma)\ \nabla_{\sigma,\gamma} u \Big)\, d\sigma. \tag{5.26}$$

As a crucial modification, we replace the exponential Perona–Malik diffusivity [279] by the Charbonnier diffusivity [70] as is typical for nonlinear diffusion operators for inpainting. The quickly decaying Perona–Malik diffusivity inhibits propagation of information from mask points too strongly, which can be remedied by the Charbonnier diffusivity.

As a second model, we consider EED inpainting which is defined by

$$Lu = \nabla^\top \big(\boldsymbol{D}(\nabla_\sigma u)\, \nabla u \big). \tag{5.27}$$

EED inpainting also uses the Charbonnier diffusivity.

As we will see, the smoothing of the outer differential operators in the IAD model is problematic for inpainting. Thus, we consider an intermediate model between IAD and EED. To this end, we equip EED with a multiscale structure tensor, giving us the *multiscale EED (MS-EED)* model as

$$Lu = \boldsymbol{\nabla}^\top \left(g(\boldsymbol{J}_\gamma)\, \boldsymbol{\nabla} u \right). \tag{5.28}$$

This model substitutes the diffusion tensor $\boldsymbol{D}(\boldsymbol{\nabla}_\sigma u)$ which relies on the single smoothing scale σ by a diffusivity of a multiscale structure tensor as defined in (5.9). It corresponds to the IAD model without the outer integration and weighting of differential operators.

We reduce the complexity of the multiscale models by choosing only $L = 8$ discrete scales, exponentially sampled between 0.1 and 3.0. Moreover, the role of the noise level n in the learned parameter functions (5.17) and (5.18) is now taken over by the mean free path length of the inpainting mask. It denotes the average distance between known data points, which is inversely proportional to the mask density. However, as we do not aim at general parameters for all mask densities in this section, we directly learn the modified parameter functions

$$\gamma(\sigma) = \exp\left(-\frac{\sigma^2}{2\alpha^2} \right), \tag{5.29}$$

$$\lambda(\sigma) = \lambda_0 \exp\left(-\frac{\sigma^2}{2\beta^2} \right). \tag{5.30}$$

We optimise all parameters of the models: Contrast parameter λ and smoothing scale σ for EED, contrast parameter λ and multiscale weight parameters α for MS-EED, and α, β, and λ_0 for IAD. We do so by applying a nested golden section search. The configurations of the learned multiscale parameter evolutions remain unchanged.

Instead of the parabolic PDE in the denoising case, the inpainting setting considers an elliptic PDE. The EED model is solved with a semi-implicit scheme with an efficient conjugate gradient solver. For IAD, applying a conjugate gradient solver is hard as the resulting system matrix is dense due involving large neighbourhood regions around a pixel. Thus, we solve it with an explicit scheme accelerated by FSI [170](see Section 3.1.2). We use cycles of length 200 and run as many cycles as required for convergence. For the sake of efficiency during the parameter and mask optimisations, convergence is defined as the relative residual decreasing by a factor of $5 \cdot 10^{-4}$. The final results with optimised parameters are computed with a relative residual decrease factor of 10^{-6}. The MS-EED model is solved in the same way as the IAD model. All models use the standard discretisation obtained for $\alpha = 0$ in the stencil (3.51) of Weickert et al. [379].

RANDOM MASK INPAINTING In our first experiment, we perform an inpainting on the *trui* [402] image with a random mask of 5%

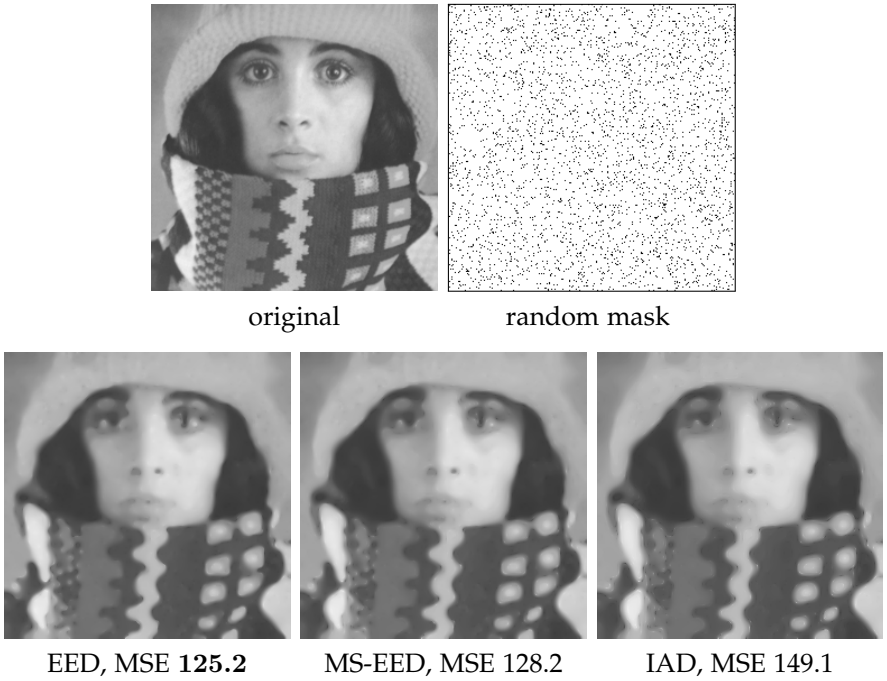


Figure 5.7: Comparison of inpainting results on a random mask with 5% density. Mask points are shown in black, and the mask image is framed for better visibility. The pure EED model performs best, followed by MS-EED and IAD.

density with all three models. The results are depicted in Figure 5.7. We see that the EED model yields the best quality, with MS-EED following closely behind. The IAD model produces a significantly worse inpainting. However, all three results are visually comparable despite their differences in MSE. The multiscale models, and IAD in particular, produce artefacts at mask points, similar to the singularities that are observed for homogeneous diffusion inpainting.

The optimal parameters of the IAD model are $\alpha = 2.95, \beta = 0.32$, and $\lambda_0 = 1.3$. In particular this means that the contrast parameters decay very quickly. Already at $\sigma = 1$, the contrast parameter is smaller than 10^{-3}. Thus, significant diffusion tensors that contribute to the inpainting live essentially only on the few finest scales. To preserve anisotropic information within the multiscale structure tensor, the weighting of scales with $\alpha = 2.95$ incorporates essentially the full range of available scales. This in turn assigns weights to the smoothing of gradient and divergence outside of the diffusion tensor, which deteriorates the result.

This becomes apparent when comparing the IAD model to the MS-EED model: The MS-EED model is free to choose the weighting in the multiscale structure tensor without introducing additional smoothing outside of it. With a much smaller weight of $\alpha = 1.83$ and a contrast parameter $\lambda = 2.58$, it can achieve a result close in quality to that of EED.

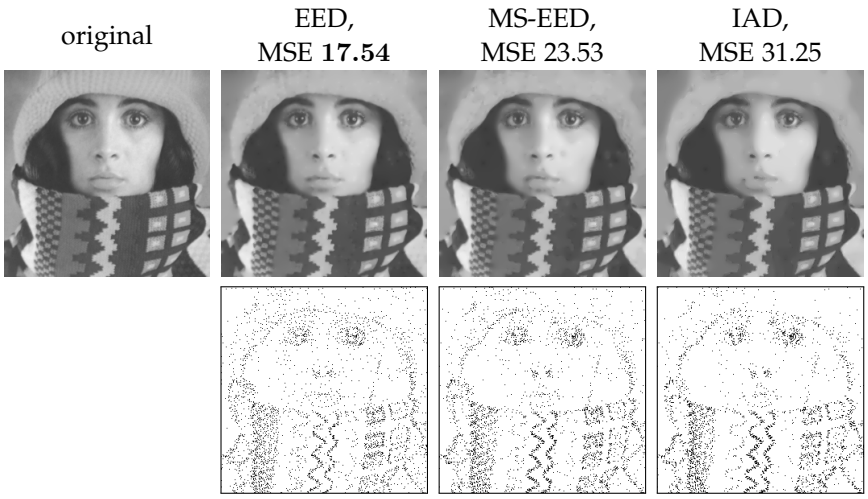


Figure 5.8: Comparison of inpainting results on masks with 5% density, optimised with probabilistic sparsification. Mask points are shown in black, and mask images are framed for better visibility. As in the random mask case, the pure EED model performs best, followed by MS-EED and IAD.

However, EED yields the best inpainting result. With $\lambda = 0.49$ and $\sigma = 1.25$ it smooths the gradient information on a single scale. We conjecture that in contrast to the denoising case, there exists a single optimal scale in the inpainting setting which is why EED remains superior.

OPTIMISED MASK INPAINTING This trend continues when investigating the inpainting of optimal masks. Instead of prescribing a random mask, we interleave the parameter optimisation with a probabilistic sparsification of the inpainting mask [246] (see Section 3.6.2). We use candidate fractions $p = 0.1$ and $q = 0.05$, and alternate sparsification and parameter optimisation until the inpainting error stabilises. The mask density of 5% remains unchanged. The resulting reconstructions and the masks can be found in Figure 5.8.

While the gaps between the models become smaller, the overall ranking remains the same. The IAD model yields worse quality than its competitors, which for example can be seen in the hat and in the more subtle patterns of the scarf. The masks look very similar, however the IAD masks tend to cluster mask points slightly more than those for EED and MS-EED.

The same parameter trend as in the random mask case can be observed. With $\alpha = 6.4, \beta = 0.32$, and $\lambda_0 = 1.49$, the IAD model suffers from the same effects as previously described. The MS-EED model with $\alpha = 2.64$ and $\lambda = 1.79$ remedies some of the problems of the IAD model, however the EED model with $\lambda = 0.86, \sigma = 0.89$ remains unbeaten.

AN OUTLOOK ON IAD INPAINTING It seems that without major modifications, the IAD model is not more suitable for sparse image inpainting than EED. In particular, considering the dramatic increase in complexity for the IAD model, EED should also be preferred from an efficiency viewpoint: The computational effort of an explicit IAD step is about a factor L larger than that of EED. This does not consider the additional computations resulting from larger Gaussian presmoothing stencils.

Moreover, solving any type of scheme involving the inverse of the system matrix is cumbersome for IAD: The smoothing of the outer differential operators transforms the nonadiagonal system matrix of EED into a dense matrix. Thus, sparse solvers are not applicable.

That being said, it is worthwhile to further investigate IAD for inpainting. One option is to retrain the parameter evolutions specifically for inpainting. This requires backpropagation through a sophisticated solver, e.g. a multigrid approach or a surrogate U-net as presented in Chapter 10. There is no guarantee that the parameter evolutions for denoising are also suitable for inpainting. For denoising, the finest scale is the most important one after all. For inpainting, however, there might be an optimal non-zero scale as EED suggests, with a decreasing weight of both coarser and finer scales.

Another option is to transform IAD into a more efficient inpainting strategy. A clever time-dynamic selection of scales may increase convergence speed: Starting with a coarse set of scales and gradually decreasing them with more explicit steps could mimic a coarse-to-fine strategy which can help to efficiently compute the required long-range interactions.

In any case, the connection between inner and outer weighted smoothing operations seems to be problematic in the inpainting setting. This connection arose from the goal of deriving IAD from a variational energy. However, the scale-adaptive penalisation in the diffusion tensor prevents us from designing such an energy. Therefore, one would not give up any important model properties by decoupling the weights and investigating whether a different set of outer smoothing weights is more appropriate for the inpainting setting.

5.5 CONCLUSIONS

This chapter has shown that one can elegantly introduce multiscale information within an anisotropic diffusion process by considering integrodifferential models. Similar as in the previous chapter, we used learning to uncover the optimal scale dynamics of the resulting model.

Apart from their evident merits of improving anisotropic diffusion methods, our findings are of more fundamental nature in two aspects: We have seen that integrodifferential equations are hitherto hardly explored but very promising extensions of differential equations. They

are not only more general, but also allow new possibilities for data adaptation, e.g. by anisotropy through multiscale integration.

On a more general note, this part of the thesis has shown the potential of learning with maximal model reduction. Such approaches can benefit from the best of two worlds: the transparency and mathematical foundation of model-based techniques and performance improvements by data-driven, learning-based strategies.

Part II

MATHEMATICALLY FOUNDED NEURAL NETWORKS

6

MATHEMATICAL MODELS AND RESIDUAL NETWORKS

This chapter is the starting point of realising the second vision of the thesis: More transparent neural networks with mathematical foundations.

An *analytic* way towards this ambitious goal is to express successful CNN architectures with well-founded mathematical concepts. However, deep learning offers a plethora of design possibilities, and modern architectures may involve hundreds of layers and millions of parameters. Thus, this way of reducing a highly complex system to a simple and transparent mathematical model is not only very burdensome, but also bears the danger to lose performance critical CNN features along the way.

An alternative, *synthetic* way uses well-established models that offer deep mathematical insights to build simple components of neural architectures which inherit these qualities.

We adopt the latter strategy. As a simple first step, we consider the simple prototypical problem of signal denoising and interpret numerical approximations of classical methods as a residual network (ResNet) architecture.

Our considerations in this chapter are purely theoretical. Since we work with very simple models, we do not embed them yet into an experimental framework. However, we can gain theoretical insights that can be useful to suggest neural architectures that are simpler, more compact, involve less parameters, and benefit from provable stability guarantees.

We establish the first comprehensive framework that allows to translate diffusion methods, wavelet approaches, and variational techniques simultaneously into a ResNet architecture. The reason for choosing three denoising techniques lies in the intrinsic stability of denoising: Noise is a perturbation of the input data, which is not supposed to change the denoised output substantially. To maximise transparency and notational simplicity, we restrict ourselves to the one-dimensional setting and choose particularly simple representatives in each class: Perona–Malik diffusion, Haar wavelet shrinkage, smooth first order variational models, and a single ResNet block. We show that discrete formulations of all three denoising approaches can be expressed as a specific ResNet block. It inherits its stability directly from the three denoising algorithms. Thus, a ResNet consisting only of these blocks is stable for any number of layers.

Whereas typical CNNs learn convolution weights and fix the nonlinear activation function to a simple design, we proceed in the opposite way: We fix the convolution kernels and study various nonlinear activation functions that are inspired by the diffusivities, shrinkage functions, and variational regularisers. For researchers from the diffusion, wavelet or variational communities, this introduces a dictionary that allows them to translate their methods directly into CNN architectures. Deep learning researchers will find hitherto unexplored nonmonotone activation functions and new motivations for existing ones.

Our results question two architectural principles behind CNNs that are usually taken for granted. One of our findings is the fact that antisymmetric activation functions can occur naturally. More importantly, we also show that nonmonotone activation functions do not contradict even the most restrictive notions of stability and well-posedness.

RELATED WORK A prominent avenue to establish mathematical foundations of CNNs is through their analysis in terms of stability. This can be achieved by studying their invertibility properties [37, 68], by exploiting sparse coding concepts [304], and by interpreting deep learning as a parameter identification or optimal control problem for ordinary differential equations [167, 352, 412]. CNNs can also be connected to flows of diffeomorphisms [309] and to parabolic or hyperbolic PDEs [112, 113, 231, 240], where it is possible to transfer L^2 stability results [319]. This thesis focuses on diffusion PDEs. In this chapter, we establish stricter stability notions such as L^∞ stability and sign stability, before relaxing these conditions in the following chapters.

We have already highlighted the connections between both wavelets and variational models and CNNs in Section 3.5. In this chapter, however, we focus on more traditional connections of wavelets [264] and variational models [326] to diffusion. These works help us to extend our findings on diffusion and residual networks to paint a bigger picture.

We argue for shifting the focus of CNN models towards more sophisticated and also nonmonotone activation functions. The CNN literature offers only few examples of training activation functions [214] or designing them in a well-founded and flexible way [361]. Nonmonotone activation functions have been suggested already before the advent of deep learning [96, 254], but fell into oblivion afterwards. We revitalise this idea by providing a natural justification from the theory of diffusion filtering.

PUBLICATION INFORMATION The contents of this chapter can be found in an unpublished technical report of Alt et al. [15]. It reflects our

first results on connections between PDEs and CNNs, and constitutes the foundation of the following chapters.

ORGANISATION OF THE CHAPTER This chapter is structured as follows. We briefly recap the basic models that we consider in Section 6.1. In Section 6.2, we discuss numerical approximations for three instances of the classical models. We interpret them in terms of a specific residual network architecture, for which we derive explicit stability guarantees. This leads to a dictionary for translating the nonlinearities of the three classical methods to activation functions, which is presented in Section 6.3. For a selection of the most popular nonlinearities, we derive their counterparts and discuss novel consequences for the design of neural networks in detail. Finally, we summarise our conclusions in Section 6.4.

6.1 REVIEW: BASIC APPROACHES

Let us now briefly specify the models that we consider. For a more detailed introduction into diffusion, wavelets, variational models, and residual networks, we refer to Chapter 3.

To ensure a consistent notation, all models in this section produce an output signal u from an input signal f. We define continuous one-dimensional signals u, f as mappings from a signal domain $\Omega = [a, b]$ to a codomain $[c, d]$. We employ reflecting boundary conditions on the signal domain boundaries a and b. The discrete signals $\boldsymbol{u}$, $\boldsymbol{f} \in \mathbb{R}^n$ are obtained by sampling the continuous functions at N equidistant positions with grid size h.

NONLINEAR DIFFUSION As a representative of a diffusion model, we consider one-dimensional Perona–Malik diffusion [279], which creates filtered versions $u(x, t)$ of an initial signal $f(x)$ with the evolution

$$\partial_t u = \partial_x \left(g\left((\partial_x u)^2 \right) \partial_x u \right), \tag{6.1}$$

with initial condition $u(x, 0) = f(x)$, diffusion time t, and reflecting boundary conditions. As highlighted in previous chapters, the central design choice lies in the diffusivity $g(s^2)$ which controls the amount of smoothing depending on the local structure of the evolving signal. We assume that it is nonnegative, nonincreasing, and bounded.

Choosing the constant diffusivity $g(s^2) = 1$ [195] leads to a homogeneous diffusion process that smooths the signal equally at all locations. A more sophisticated diffusivity such as the exponential Perona–Malik diffusivity $g(s^2) = \exp\left(-\frac{s^2}{2\lambda^2}\right)$ [279] inhibits smoothing around discontinuities where $|\partial_x u|$ is larger than the contrast parameter λ. This allows discontinuity-preserving smoothing.

WAVELET SHRINKAGE As a wavelet shrinkage model, we consider shift-invariant Haar wavelet shrinkage in one dimension (see Section 3.3). It can be seen as the one-dimensional counterpart of the baseline model of Chapter 4. This model computes the discrete reconstruction $\boldsymbol{u}$ from $\boldsymbol{f}$ as

$$\boldsymbol{u} = \widetilde{\boldsymbol{W}} S(\boldsymbol{W}\boldsymbol{f})\,. \tag{6.2}$$

The matrices $\boldsymbol{W}$ and $\widetilde{\boldsymbol{W}}$ represent the one-dimensional shift-invariant forward and backward Haar wavelet transformations. As we have seen, besides the choice of the wavelet basis, the result is strongly influenced by the shrinkage function $S(s)$.

The hard shrinkage function [248] eliminates all coefficients with a magnitude smaller than the threshold parameter, while the soft shrinkage function [107] additionally modifies the remaining coefficients equally.

VARIATIONAL REGULARISATION As a variational model we choose a functional with a quadratic data term and an increasing regulariser

$$E(u) = \int_\Omega \left((u-f)^2 + \alpha \Psi\left((\partial_x u)^2 \right) \right) dx\,. \tag{6.3}$$

where the data term $(u-f)^2$ drives u towards the input data f, and a first order regularisation $\Psi\left((\partial_x u)^2\right)$ enforces smoothness conditions on u by penalising variations in $\partial_x u$. We can control the balance between both terms by the regularisation parameter $\alpha > 0$. Choosing e.g. $\Psi(s^2) = s^2$ is called Whittaker–Tikhonov regularisation [354, 391].

RESIDUAL NETWORKS On the CNN side, we consider a single residual block [173] consisting of two convolutional layers with biases and nonlinear activation functions after each layer

$$\boldsymbol{u} = \varphi_2(\boldsymbol{f} + \boldsymbol{W}_2\, \varphi_1(\boldsymbol{W}_1 \boldsymbol{f} + \boldsymbol{b}_1) + \boldsymbol{b}_2) \tag{6.4}$$

with discrete convolution matrices $\boldsymbol{W}_1, \boldsymbol{W}_2$, activation functions φ_1, φ_2, and bias vectors $\boldsymbol{b}_1, \boldsymbol{b}_2$.

The crucial difference between residual networks and the three previous approaches is the design focus: The three classical methods consider complex nonlinear modelling functions, while CNNs mainly focus on learning convolution weights and use simple activation functions. We will see that by permitting more general activation functions, we can relate all four methods within a unifying framework.

6.2 TRANSLATION INTO RESIDUAL NETWORKS

Now we are in a position to discuss numerical approximations for the three classical models that allow to interpret them in terms of a residual network architecture.

FROM NONLINEAR DIFFUSION TO RESIDUAL NETWORKS In practice, the continuous diffusion process is discretised and iterated to approximate the continuous solution $u(x,T)$ for a stopping time T. With the help of the flux function $\Phi(s) = g(s^2)\, s$ we rewrite the diffusion equation (6.1) as

$$\partial_t u = \partial_x \left(\Phi(\partial_x u)\right). \tag{6.5}$$

For this equation, we perform a standard discretisation in the spatial and the temporal domain. This yields an explicit scheme which can be iterated. Starting with an initial signal $\boldsymbol{u}^0 = \boldsymbol{f}$, the evolving signal $\boldsymbol{u}^k$ at a time step k is used to compute $\boldsymbol{u}^{k+1}$ at the next step by

$$\frac{u_i^{k+1} - u_i^k}{\tau} = \frac{1}{h}\left(\Phi\left(\frac{u_{i+1}^k - u_i^k}{h}\right) - \Phi\left(\frac{u_i^k - u_{i-1}^k}{h}\right)\right). \tag{6.6}$$

Here the temporal derivative is discretised by a forward difference with time step size τ. We apply a forward difference to implement the inner spatial derivative operator and a backward difference for the outer spatial derivative operator. Both can be realised with a simple convolution.

To obtain a scheme which is stable in the L^∞ norm, one can show that the time step size must fulfil

$$\tau \leq \frac{h^2}{2 g_{\max}}, \tag{6.7}$$

where $g_{\max}$ is the maximum value that the diffusivity $g(s^2) = \frac{\Phi(s)}{s}$ can attain [369]. This guarantees a maximum–minimum principle, stating that the values of the filtered signal $\boldsymbol{u}^k$ do not lie outside the range of the original signal $\boldsymbol{f}$.

To achieve a substantial filter effect, one often needs a diffusion time T that exceeds the stability limit in (6.7). Then one concatenates m explicit steps with a time step size $\tau = \frac{T}{m}$ that satisfies (6.7).

In order to translate diffusion into residual networks, we rewrite the explicit scheme (6.6) in matrix-vector form:

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \tau\, \boldsymbol{D}_h^- \left(\boldsymbol{\Phi}\left(\boldsymbol{D}_h^+ \boldsymbol{u}^k\right)\right), \tag{6.8}$$

where $\boldsymbol{D}_h^+$ and $\boldsymbol{D}_h^-$ are convolution matrices denoting forward and backward difference operators with grid size h, respectively. In this notation, the resemblance to a residual block becomes apparent:

Theorem 1. *A diffusion step (6.8) is equivalent to a residual block (6.4) if*

$$\varphi_1 = \tau\, \boldsymbol{\Phi}, \quad \varphi_2 = \mathrm{Id}, \quad \boldsymbol{W}_1 = \boldsymbol{D}_h^+, \quad \boldsymbol{W}_2 = \boldsymbol{D}_h^-, \tag{6.9}$$

and the bias vectors $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ are set to $\boldsymbol{0}$.

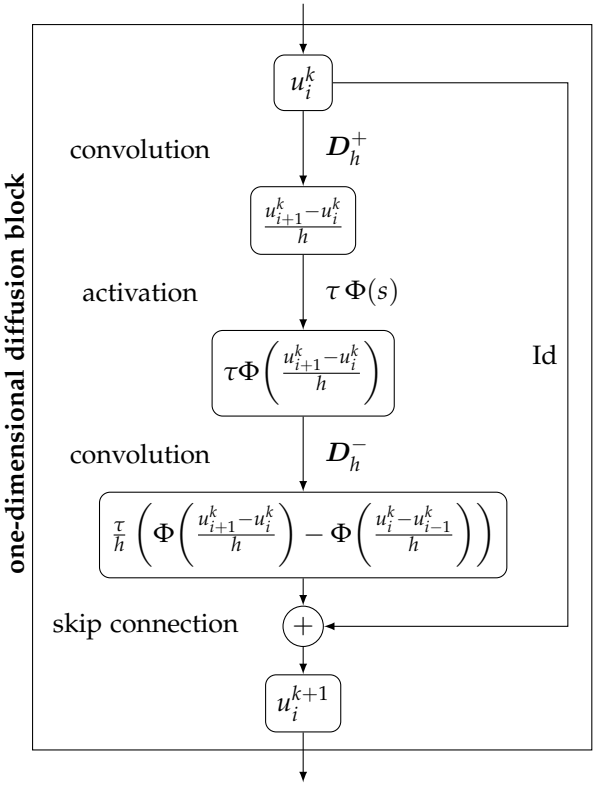


Figure 6.1: One-dimensional diffusion block for one explicit nonlinear diffusion step (6.6) with flux function $\Phi(s)$ and time step size τ.

We see that the convolutions implement forward and backward difference operators. Since we have $\boldsymbol{D}_h^+ = -\left(\boldsymbol{D}_h^-\right)^\top$, the convolutions are symmetric to each other. This will become an important property when generalising this result to arbitrary filters in Chapter 7.

Crucially, the inner activation function φ_1 corresponds to the flux function $\tau\boldsymbol{\Phi}$ rescaled by the time step size. The effect of the skip-connection in the residual block also becomes clear now: It is the central concept to realise a time discretisation. We call a block of this form a *one-dimensional diffusion block*.

Figure 6.1 visualises such a diffusion block. Graph nodes contain the current state of the signal at position i, while edges describe operations which are applied to proceed from one node to the next.

A related approach for learning parameters of nonlinear diffusion filters in the context of inverse problems has been proposed by Chen and Pock [75]. However, their work establishes a diffusion-reaction framework: The skip-connection rewards similarity to the original image in each step. Therefore, it cannot be translated directly into a residual block (6.4). On the contrary, we use a pure diffusion model. Our skip-connection rewards similarity to the image from the previous layer and is thus compatible with the residual block formulation.

A consequence of Theorem 1 is that a residual network chaining m diffusion blocks with activation function $\tau\Phi(s)$ approximates a nonlinear diffusion process with stopping time $T = m\tau$ and diffusivity $g(s^2) = \frac{\Phi(s)}{s}$. These insights enable us to translate a diffusivity g directly into an activation function through the flux function Φ:

$$\boxed{\varphi(s) \;=\; \tau\,\Phi(s) \;=\; \tau\,g(s^2)\,s} \tag{6.10}$$

While this translation from a diffusion step to a residual block appears simple, it will serve as the Rosetta stone in our dictionary: It also allows to connect wavelet methods and variational regularisation to residual networks, since both paradigms can be related to diffusion [264, 326]. Let us now sketch these correspondences.

FROM WAVELET SHRINKAGE TO NONLINEAR DIFFUSION The connection between diffusion PDEs and specific ResNets can be extended to wavelet shrinkage. This is possible due to various equivalence results between diffusion methods and wavelets; see e.g. [105, 243, 264, 385]. In our context, it is sufficient to focus on the results of Mrázek and Weickert [264].

To explore the connection between wavelet shrinkage and nonlinear diffusion, they consider shift-invariant Haar wavelet shrinkage on the finest scale. The missing multiscale structure is compensated by iterating this shrinkage. They show that one step with shrinkage function $S(s)$ is equivalent to an explicit diffusion step with diffusivity $g(s^2)$, grid size $h = 1$, and time step size τ if

$$g(s^2) = \frac{1}{4\tau}\left(1 - \frac{\sqrt{2}}{s} S\left(\frac{s}{\sqrt{2}}\right)\right). \tag{6.11}$$

The L^∞ stability condition (6.7) from the diffusion case translates into a condition on the shrinkage function

$$-s \leq S(s) \leq s \quad \text{for } s > 0, \tag{6.12}$$

which is less restrictive than the typical design principle

$$0 \leq S(s) \leq s \quad \text{for } s > 0. \tag{6.13}$$

Mrázek et al. show that the latter one leads to a sign stable process in the sense of Schönberg [332], i.e. the resulting signal shows not more sign changes than the input signal. This is a stronger stability notion than L^∞ stability. It limits the time step size to $\tau \leq \frac{h^2}{4g_{\max}}$. This is half the bound of (6.7).

FROM VARIATIONAL MODELS TO NONLINEAR DIFFUSION We make use of the connections between variational methods and nonlinear diffusion presented by Scherzer and Weickert [326]. They con-

sider an energy functional with a quadratic data term and a regulariser $\Psi(s^2)$:

$$E(u) = \int_\Omega \left((u-f)^2 + \alpha \, \Psi\left((\partial_x u)^2 \right) \right) dx. \tag{6.14}$$

The corresponding Euler–Lagrange equation for a minimiser u of the functional reads

$$\frac{u-f}{\alpha} = \partial_x \left(\Psi'\left((\partial_x u)^2 \right) \, \partial_x u \right). \tag{6.15}$$

This can be regarded as a fully implicit time discretisation for a nonlinear diffusion process with stopping time $T = \alpha$ and diffusivity $g(s^2) = \Psi'(s^2) = \frac{\partial}{\partial(s^2)} \Psi(s^2)$. This process can also be approximated by m explicit diffusion steps of type (6.6) with time step size $\tau = \frac{\alpha}{m}$ [292], where m is chosen such that the stability condition (6.7) holds.

A related interpretation which is equivalent to the concept of iterated regularisation is given by algorithm unrolling [217, 260, 345].

STABILITY GUARANTEES FOR OUR RESIDUAL NETWORK The connections established so far imply direct stability guarantees for networks consisting of diffusion blocks.

Theorem 2 (L^∞ Stability of Residual Networks with Diffusion Blocks)**.** *A residual network chaining any number of diffusion blocks with grid size h and activation function $\varphi(s) = \tau\Phi(s) = \tau g(s^2)s$ with finite Lipschitz constant L is stable in the L^∞ norm if*

$$\tau \leq \frac{h^2}{2L}, \tag{6.16}$$

It is also sign stable if the bound is chosen half as large.

Proof. Since $\Phi(s) = g(s^2)\, s$ and g is a nonincreasing symmetric diffusivity with bound $g_{\max}$, it follows that $L = g_{\max}$. Thus, (6.16) is the network analogue of the stability condition (6.7) for an explicit diffusion step. In the same way, the diffusion block inherits its sign stability from the sign stability condition of wavelet shrinkage. Stability of the full network follows by induction. □

Note that our results in terms of L^∞ or sign stability are stricter stability notions than the L^2 stability proposed by Ruthotto and Haber [319]. However, they investigate more general convolution filters. This will be our focus in the following chapter.

Contrary to the result of Ruthotto and Haber [319], our stability result does not require activation functions to be monotone. We will see that widely used diffusivities and shrinkage functions naturally lead to nonmonotone activation functions.

Table 6.1: Dictionary for diffusivities $g(s^2)$, regularisers $\Psi(s^2)$, wavelet shrinkage functions $S(s)$, and activation functions $\Phi(s)$. A nonlinearity from a row can be translated into a nonlinearity from a column with the respective equation.

from \ to	diffusivity	regulariser	shrinkage function	activation function
diffusivity	$g\left(s^2\right)$	$\Psi\left(s^2\right) = \int_0^{s^2} g(x)\ dx$	$S(s) = s\left(1 - 4\tau\, g\left(2s^2\right)\right)$	$\varphi(s) = \tau g\left(s^2\right)\, s$
regulariser	$g\left(s^2\right) = \Psi'\left(s^2\right)$	$\Psi(s^2)$	$S(s) = s - \sqrt{2}\alpha\, \Psi'\left(2s^2\right)$	$\varphi(s) = \tau \Psi'\left(s^2\right)\, s$
shrinkage function	$g\left(s^2\right) = \frac{1}{4\tau}\left(1 - \frac{\sqrt{2}}{s} S\left(\frac{s}{\sqrt{2}}\right)\right)$	$\Psi\left(s^2\right) = \frac{1}{4\alpha}\left(s^2 - 2\sqrt{2}\int_0^{s^2} S\left(\frac{x}{\sqrt{2}}\right) dx\right)$	$S(s)$	$\varphi(s) = \frac{1}{4}\left(s - \sqrt{2}\, S\left(\frac{s}{\sqrt{2}}\right)\right)$
activation function	$g\left(s^2\right) = \frac{\varphi(s)}{\tau\, s}$	$\Psi\left(s^2\right) = \frac{1}{\tau}\int_0^{s^2} \frac{\varphi(x)}{x}\, dx$	$S(s) = s - 2\sqrt{2}\, \varphi\left(\sqrt{2}s\right)$	$\varphi(s)$

6.3 DICTIONARY OF ACTIVATION FUNCTIONS

MAIN RESULT Exploiting the Equations (6.10), (6.11), and (6.15), we are now in the position to present a general dictionary which can be used to translate arbitrary diffusivities, wavelet shrinkage functions, and variational regularisers into activation functions. This dictionary is displayed in Table 6.1.

On one hand, our dictionary provides a blueprint for researchers acquainted with diffusion, wavelet shrinkage or regularisation to build a residual network for a desired model while preserving important theoretical properties. This can help them to develop rapid prototypes of the corresponding filters without the need to pay attention to implementational details. Also parallelisation for GPUs is readily available. Last but not least, these methods can be gradually refined by learning.

On the other hand, also CNN researchers can benefit. The dictionary shows how to restrict CNN architectures or parts thereof to models which are well-motivated, provably stable, and can benefit from the rich research results for diffusion, wavelet shrinkage and regularisation. Lastly, it can inspire CNN practitioners to use more sophisticated activation functions, in particular antisymmetric and nonmonotone ones.

WHAT WE CAN LEARN FROM POPULAR METHODS Let us now apply our general dictionary to prominent diffusivities, shrinkage functions, and regularisers in order to identify their activation functions.

We visualise these functions in Tables 6.2 and 6.4 and display their mathematical formulas in Tables 6.3 and 6.5. For our examples, we choose a grid size of $h = 1$. As we have $g_{\max} = 1$ for all cases considered, we set $\tau = \alpha = \frac{1}{4}$. This fulfils the sign stability condition (6.16).

We generally observe that all resulting activation functions $\tau\Phi(s) = \tau g(s^2)\, s$ are antisymmetric, since they involve the product of a symmetric diffusivity $g(s^2)$ and the antisymmetric identity function s. This is very natural in the diffusion case, where the argument of the flux function is the signal derivative $\partial_x u$. It reflects a desired invariance axiom of denoising: Signal negation and filtering are commutative.

Still, the discussed antisymmetric activation functions can be expressed with typical ReLU functions [266]. The truncated total variation activation relying on the flux function

$$\Phi(s) = \begin{cases} s, & |s| \leq \sqrt{2}\,\theta, \\ \sqrt{2}\,\theta \operatorname{sgn}(s), & |s| > \sqrt{2}\,\theta, \end{cases} \tag{6.17}$$

provides a simple example as it can be rewritten as

$$\Phi(s) = s - \operatorname{ReLU}\left(s - \sqrt{2}\,\theta\right) + \operatorname{ReLU}\left(-s - \sqrt{2}\,\theta\right). \tag{6.18}$$

Other activation functions which are not piecewise linear can be approximated with a series of feedforward layers [190]. By allowing more advanced antisymmetric activation functions, we can end up with fewer layers.

Our six examples fall into two classes: The first class in Tables 6.2 and 6.3 comprises diffusion filters with constant [195] and Charbonnier diffusivities [70], as well as soft wavelet shrinkage [107] which involves a Huber regulariser [193] and a truncated total variation (TV) diffusivity [16, 313].

These methods have strictly convex regularisers, and their shrinkage functions do not approximate the identity function for $s \to \pm\infty$. Most importantly, their activation functions are monotonically increasing. This is compatible with the standard scenario in deep learning where the ReLU activation function dominates [266]. On the diffusion side, the corresponding increasing flux functions act contrast reducing. Strictly convex regularisers have unique minimisers, and popular minimisation algorithms such as gradient descent converge globally.

The second class in Tables 6.4 and 6.5 is much more exciting. Its representatives are given by Perona–Malik diffusion [279] and two wavelet shrinkage methods: garrote shrinkage [136] and hard shrinkage [248]. Garrote shrinkage corresponds to the truncated balanced forward–backward (BFB) diffusivity of Keeling and Stollberger [210], while hard shrinkage has a truncated quadratic regulariser which is used in the weak string model of Geman and Geman [138].

Approaches of the second class have nonconvex regularisers, which may lead to multiple energy minimisers. Their shrinkage functions converge to the identity function for $s \to \pm\infty$. The flux function of the diffusion filter is nonmonotone. While this was considered somewhat problematic for continuous diffusion PDEs, it has been shown that their discretisations are well-posed [372], in spite of the fact that they may act contrast enhancing. Since the activation function is a rescaled flux function, it is also nonmonotone. This is very unusual for CNN architectures. Although there were a few very early proposals in the neural network literature arguing that such networks offer a larger storage capacity [96] and have some optimality properties [254], they had no impact on modern CNNs.

Our results motivate the idea of nonmonotone activation functions from a different perspective. Since it is well-known that nonconvex variational methods can outperform convex ones, it appears promising to incorporate nonmonotone activations into CNNs in spite of some challenges that have to be mastered.

6.4 CONCLUSIONS

We have seen that CNNs and classical methods have much more in common than many would expect: Focusing on three classical de-

Table 6.2: Plots of selected functions resulting in monotone activation functions. Names of known functions are written above the graphs. Bold font indicates the best known function for each row. Axes and parameters are individually scaled for optimal qualitative inspection.

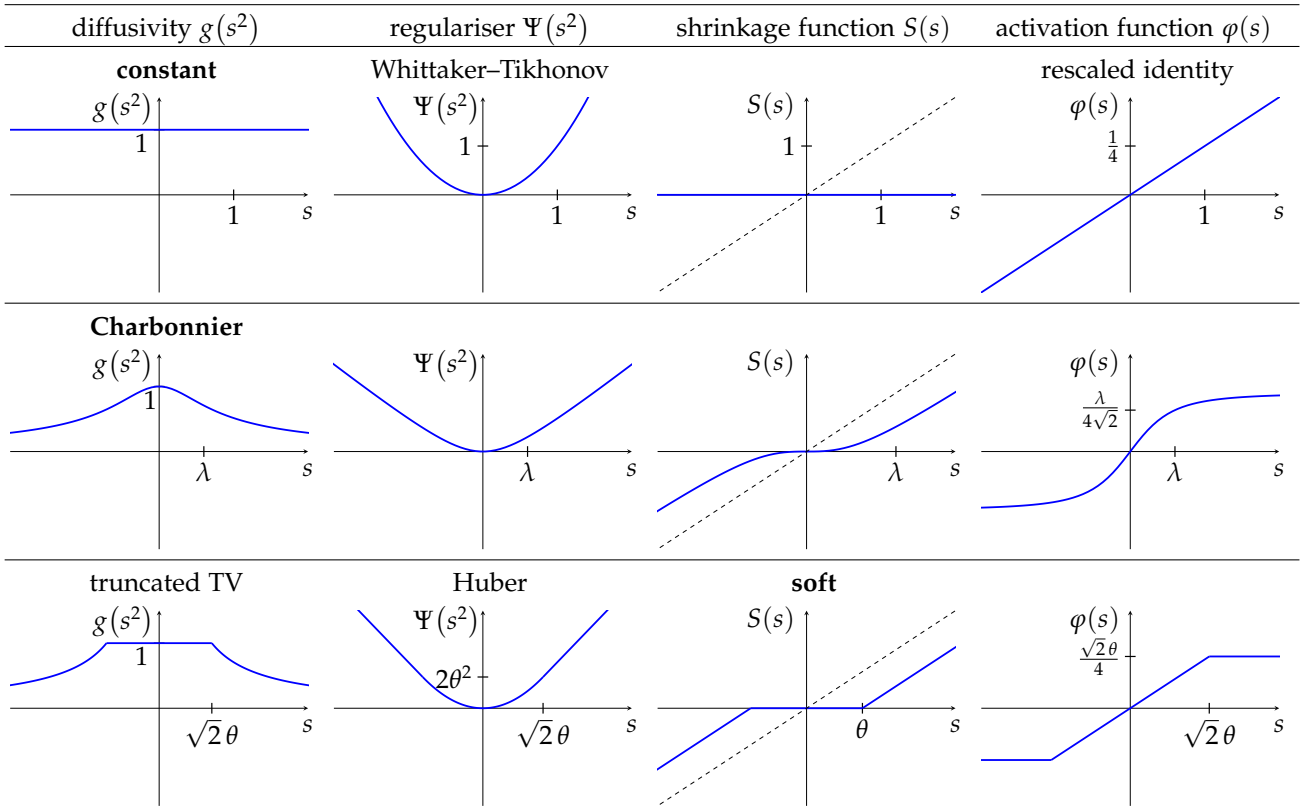


Table 6.3: Formulas for the function plots in Table 6.2. The names of known functions are written above the equations. Bold font indicates the best known function for each row.

diffusivity $g(s^2)$	regulariser $\Psi(s^2)$	shrinkage function $S(s)$	activation function $\varphi(s)$
constant	Whittaker–Tikhonov		rescaled identity
$g\left(s^2\right) = 1$	$\Psi\left(s^2\right) = s^2$	$S(s) = 0$	$\varphi(s) = \frac{s}{4}$
Charbonnier			
$g\left(s^2\right) = \frac{1}{\sqrt{1+\frac{s^2}{\lambda^2}}}$	$\Psi\left(s^2\right) = 2\lambda^2\sqrt{1+\frac{s^2}{\lambda^2}} - 2\lambda^2$	$S(s) = s\left(1 - \frac{1}{\sqrt{1+\frac{2s^2}{\lambda^2}}}\right)$	$\varphi(s) = \frac{s}{4\sqrt{1+\frac{s^2}{\lambda^2}}}$
truncated TV	Huber	**soft**	
$g\left(s^2\right) = \begin{cases} 1, & \lvert s\rvert \le \sqrt{2}\,\theta \\ \frac{\sqrt{2}\,\theta}{\lvert s\rvert}, & \lvert s\rvert > \sqrt{2}\,\theta \end{cases}$	$\Psi\left(s^2\right) = \begin{cases} s^2, & \lvert s\rvert \le \sqrt{2}\,\theta \\ 2\theta\left(\sqrt{2}\,\lvert s\rvert - \theta\right), & \lvert s\rvert > \sqrt{2}\,\theta \end{cases}$	$S(s) = \begin{cases} 0, & \lvert s\rvert \le \theta \\ s - \theta\,\mathrm{sgn}(s), & \lvert s\rvert > \theta \end{cases}$	$\varphi(s) = \begin{cases} \frac{s}{4}, & \lvert s\rvert \le \sqrt{2}\,\theta \\ \frac{1}{4}\sqrt{2}\,\theta\,\mathrm{sgn}(s), & \lvert s\rvert > \sqrt{2}\,\theta \end{cases}$

Table 6.4: Plots of selected functions resulting in nonmonotone activation functions. Names of known functions are written above the graphs. Bold font indicates the best known function for each row. Axes and parameters are individually scaled for optimal qualitative inspection.

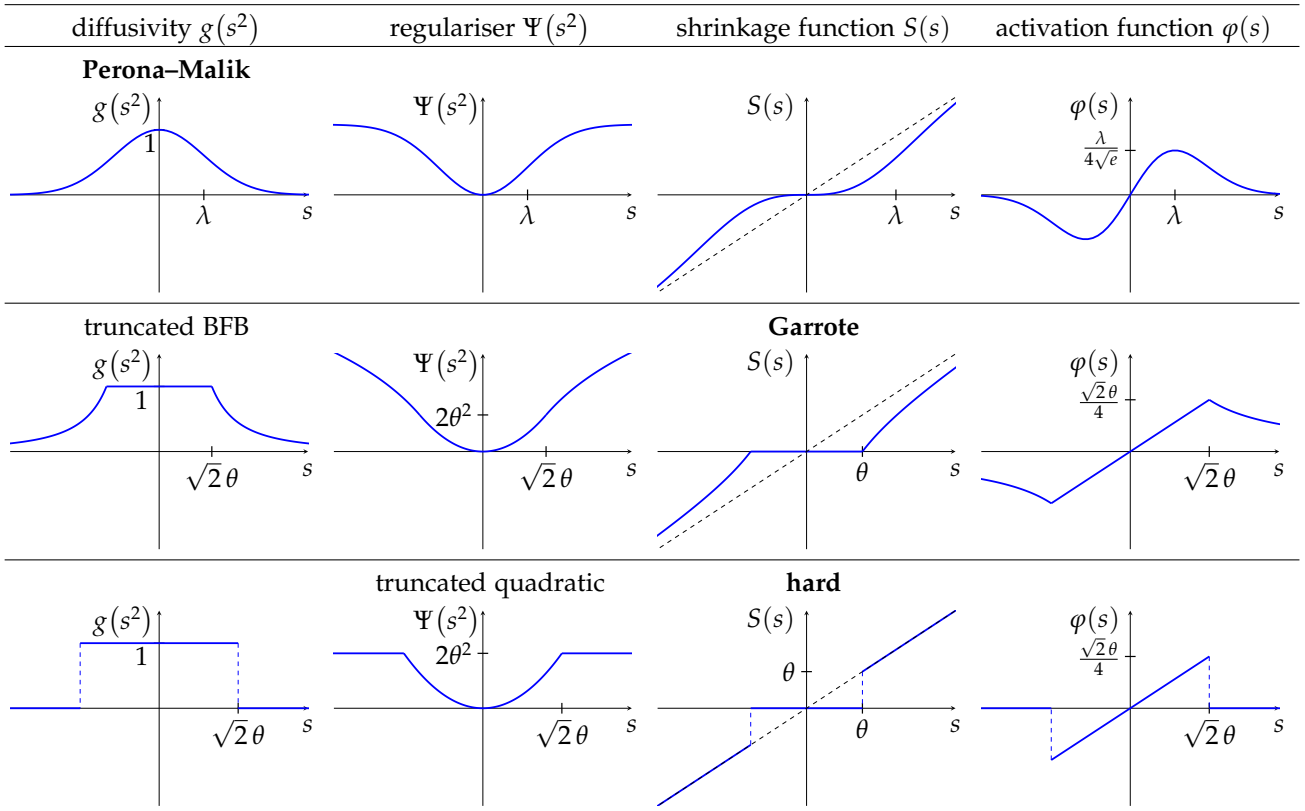


Table 6.5: Formulas for the function plots in Table 6.4. The names of known functions are written above the equations. Bold font indicates the best known function for each row.

diffusivity $g(s^2)$	regulariser $\Psi(s^2)$	shrinkage function $S(s)$	activation function $\varphi(s)$
Perona–Malik			
$g\left(s^2\right) = \exp\left(-\frac{s^2}{2\lambda^2}\right)$	$\Psi\left(s^2\right) = 2\lambda^2\left(1-\exp\left(-\frac{s^2}{2\lambda^2}\right)\right)$	$S(s) = s\left(1-\exp\left(-\frac{s^2}{\lambda^2}\right)\right)$	$\varphi(s) = \frac{s}{4}\exp\left(-\frac{s^2}{2\lambda^2}\right)$
truncated BFB		**Garrote**	
$g\left(s^2\right) = \begin{cases} 1, & \lvert s\rvert \le \sqrt{2}\,\theta \\ \frac{2\theta^2}{s^2}, & \lvert s\rvert > \sqrt{2}\,\theta \end{cases}$	$\Psi\left(s^2\right) = \begin{cases} s^2, & \lvert s\rvert \le \sqrt{2}\,\theta \\ 2\theta^2\left(\ln\left(\frac{s^2}{2\theta^2}\right)+1\right), & \lvert s\rvert > \sqrt{2}\,\theta \end{cases}$	$S(s) = \begin{cases} 0, & \lvert s\rvert \le \theta \\ s-\frac{\theta^2}{s}, & \lvert s\rvert > \theta \end{cases}$	$\varphi(s) = \begin{cases} \frac{s}{4}, & \lvert s\rvert \le \sqrt{2}\,\theta \\ \frac{\theta^2}{2s}, & \lvert s\rvert > \sqrt{2}\,\theta \end{cases}$
	truncated quadratic	**hard**	
$g\left(s^2\right) = \begin{cases} 1, & \lvert s\rvert \le \sqrt{2}\,\theta \\ 0, & \lvert s\rvert > \sqrt{2}\,\theta \end{cases}$	$\Psi\left(s^2\right) = \begin{cases} s^2, & \lvert s\rvert \le \sqrt{2}\,\theta \\ 2\theta^2, & \lvert s\rvert > \sqrt{2}\,\theta \end{cases}$	$S(s) = \begin{cases} 0, & \lvert s\rvert \le \theta \\ s, & \lvert s\rvert > \theta \end{cases}$	$\varphi(s) = \begin{cases} \frac{s}{4}, & \lvert s\rvert \le \sqrt{2}\,\theta \\ 0, & \lvert s\rvert > \sqrt{2}\,\theta \end{cases}$

noising approaches in a one-dimensional setting and on a ResNet architecture with simple convolutions, we have established a dictionary that allows to translate diffusivities, shrinkage functions, and regularisers into activation functions. This does not only yield strict stability results for specific ResNets with an arbitrary number of layers, but also suggests to invest more efforts into the design of activation functions. In particular, antisymmetric and nonmonotone activation functions warrant more attention.

Needless to say, our restrictions to a single dimension, to a single scale, and to denoising methods have been introduced mainly for didactic reasons. This chapter is considered a theoretical analysis of the connections between the models in the simplest setting. The following two chapters extend these basic results to more general frameworks: Chapter 7 analyses the one-dimensional connections in the presence of generalised filter matrices, and Chapter 8 is concerned with the two-dimensional setting and associated concepts such as rotation invariance.

7

NUMERICAL ALGORITHMS AND NEURAL ARCHITECTURES

In the previous chapter we have established first connections between diffusion and residual networks. Using further popular links between diffusion, variational methods, and wavelets, we were able to connect all four worlds in a unifying framework. This allowed us to focus on the links between different nonlinear design functions, while the convolution structure of the filters was fixed. Even though this led us to strict stability guarantees, this framework was more of tutorial value w.r.t. activation functions inspired from diffusivities, shrinkage functions, and variational regularisers.

This ignored the strengths of practical neural networks: Learning the filter kernels in combination with nonlinear activation functions makes neural networks so powerful. To this end, we extend our results to more general diffusion filters in this chapter, at the cost of a slightly weaker stability notion in the form of Euclidean stability.

Moreover, the previous chapter has shown that not the diffusion model alone determines the resulting network architecture, but also the choice of the numerical algorithm with which the model is solved. Thus, this chapter provides an in-depth discussion of different numerical algorithms for diffusion models and how they translate into neural architectures.

Similar to untrained neural networks, numerical algorithms can be applied to a multitude of problems in a general purpose fashion. The model at hand is then specified by the differential equation, which is approximated by training the neural network. Thus, we believe that the design principles of modern neural networks realise a small but powerful set of numerical strategies at their core.

In particular, we consider the neural counterparts of explicit schemes, acceleration strategies thereof [110, 170], implicit schemes, and multigrid approaches [53, 54]. We supplement our findings with an experimental evaluation of the proposed network architectures for denoising and inpainting tasks. Therein, we demonstrate the effectiveness of the architectures together with nonmonotone activation functions in practice.

As a starting point of our investigations, we consider a generalised nonlinear diffusion equation. Similar to the previous chapter, we stay within the one-dimensional setting for didactic purposes only, since all important concepts can already be translated in this simple setting. We show that also this diffusion model can be connected to ResNets when considering an explicit discretisation.

This connection inspires a more general ResNet architecture than the one in the previous chapter. It follows a symmetric structure which saves half the amount of network parameters compared to standard ResNets, and additionally allows to derive a theory for guaranteeing stability in the Euclidean norm. This chapter also provides empirical evidence that nonmonotone activation functions can be effective for trainable denoising.

By considering acceleration strategies for explicit schemes and solution strategies for implicit schemes, we justify the effectiveness of skip connections in neural networks. We show that Du Fort–Frankel schemes [110], fast semi-iterative (FSI) schemes [170], and fixed point iterations for implicit schemes motivate different architectural designs which all rely on skip connections as a foundation of their efficiency.

Finally, we consider the rich class of multigrid approaches [53, 54]. We show that a nonlinear full approximation scheme (FAS) can be cast in the form of the popular U-net [306] architecture. We think that at their core, U-nets realise a multigrid strategy, and we support this claim by proposing a U-net which realises a full multigrid strategy for an inpainting task.

The findings in this chapter do not only inspire new design criteria for stable neural architectures and show that uncommon design choices can perform well in practice. They also provide structural insights into the success of popular CNN architectures from the perspective of numerical algorithms.

RELATED WORK In contrast to many others, our philosophy of translating numerical concepts into neural architectures is shared only by few works [43, 229, 240, 275, 416]; see also Section 3.5. They motivate additional or modified skip connections based on numerical schemes for ODEs, such as Runge-Kutta methods or implicit Euler schemes. Our work provides additional motivations for such skip connections based on several numerical strategies for PDEs.

A common result is that ResNets with a symmetric filter structure can be shown to be stable in the Euclidean norm [166, 167, 309, 319, 412]. We motivate this result from a novel viewpoint based on diffusion processes. In contrast to previous results, this unique starting point allows us to present our stability result independently of the monotonicity of the activation function, inspiring the use of nonmonotone and trainable activation functions.

Nonmonotone activation functions are rarely found in standard CNNs, with some notable exceptions [75, 151, 272]. Recently, the so-called Swish activation [294] and modifications thereof [257, 417] have been found to empirically boost the classification performance of CNNs. While these activations are modifications of the ReLU activation which are nonmonotone around the zero position, the activations that arise from our diffusion interpretation are antisymmetric and

nonmonotone. Such functions have been analysed before the advent of deep learning [96, 254], but have not found their way into current CNN architectures. Our experiments, however, suggest that these activations can be advantageous in practice.

Multigrid ideas have been combined with CNNs already in the early years of neural network research [29, 30]. Current works use inspiration from multigrid concepts to learn restriction and prolongation operators of multigrid solvers [159, 209], to couple channels for parameter reduction [117] or to boost training with ideas from multigrid-in-time [163].

However, to the best of our knowledge, the only architectures that consequently implement a trainable multigrid approach are presented by He and Xu [172] and Hartmann et al. [171]. However, both works do not draw any connections to the popular U-net architecture [306], whereas we directly link both concepts.

PUBLICATION INFORMATION The contents in this chapter appeared in the conference paper of Alt et al. [10] at the Scale Space and Variational Methods in Computer Vision conference in 2021. An extended version has been published as an invited paper in the Journal of Mathematical Imaging and Vision [11]. The present chapter includes additional visualisations of multiple diffusion-inspired neural architectures.

ORGANISATION OF THE CHAPTER In Section 7.1, we review a generalised nonlinear diffusion model. We connect it to residual networks in Section 7.2 and analyse the implications in terms of stability and novel activation functions. Afterwards we motivate skip connections from different numerical algorithms in Section 7.3. We review multigrid approaches and U-nets in Section 7.4 before connecting both worlds in Section 7.5. Finally, we experimentally evaluate the proposed architectures in Section 7.6 and present a discussion and our conclusions in Section 7.7.

7.1 REVIEW: GENERALISED ONE-DIMENSIONAL DIFFUSION

In this section, we review generalised one-dimensional diffusion filters as the basic model for our first translation. We start by considering a generalised one-dimensional diffusion PDE of arbitrary high order. It produces signals $u(x,t) : [a,b] \times [0,\infty) \to \mathbb{R}$ evolving over time from an initial signal $f(x)$ on a domain $[a,b] \subset \mathbb{R}$ according to

$$\partial_t u = -\mathcal{D}^* \left(g\left(|\mathcal{D}u|^2\right) \mathcal{D}u \right) \tag{7.1}$$

with reflecting boundary conditions. We use a general differential operator

$$\mathcal{D} = \sum_{m=0}^{M} \alpha_m \partial_x^m \tag{7.2}$$

and its adjoint

$$\mathcal{D}^* = \sum_{m=0}^{M} (-1)^m \alpha_m \partial_x^m. \tag{7.3}$$

The operators consist of weighted derivatives up to order M with weights α_m of arbitrary sign, yielding a PDE of order $2M$.

Choosing e.g. $M = 1$ yields the second order PDE of Perona and Malik [279] from the previous chapter (cf. Equation (6.1)), while $M = 2$ leads to a one-dimensional version of the fourth order model of You and Kaveh [401] (cf. Equation (3.21)).

In analogy to the previous chapter, the diffusion PDE (7.1) is the gradient flow which minimises the energy functional

$$E(u) = \int_a^b \Psi(|\mathcal{D}u|^2)\, dx, \tag{7.4}$$

where the penaliser Ψ can be linked to the diffusivity with the connection $g = \Psi'$ [326]. The penaliser must be increasing, but not necessarily convex. This connection motivates the specific filter structure of the resulting ResNet blocks.

7.2 FROM DIFFUSION TO SYMMETRIC RESIDUAL NETWORKS

We are now in the position to show that explicit diffusion schemes realise a ResNet architecture with a symmetric filter structure. As before, we consider a ResNet block of the form

$$\boldsymbol{u} = \varphi_2(\boldsymbol{f} + \boldsymbol{W}_2\, \varphi_1(\boldsymbol{W}_1 \boldsymbol{f} + \boldsymbol{b}_1) + \boldsymbol{b}_2)\,. \tag{7.5}$$

In analogy to the previous chapter we can rewrite the generalised nonlinear diffusion equation (7.1) with the help of the flux function

$$\Phi(s) = g(s^2)\, s \tag{7.6}$$

as

$$\partial_t u = -\mathcal{D}^* \Phi(\mathcal{D}u)\,. \tag{7.7}$$

A discrete version of this equation is obtained by means of a standard finite difference scheme. We discretise the temporal derivative by a forward difference with time step size τ. The spatial derivative operator $\mathcal{D}$ is implemented by a convolution matrix $\boldsymbol{K}$. Consequently, the adjoint operator $\mathcal{D}^*$ is realised by a transposed convolution matrix $\boldsymbol{K}^\top$. The matrix transposition corresponds to mirroring the associated discrete convolution kernel.

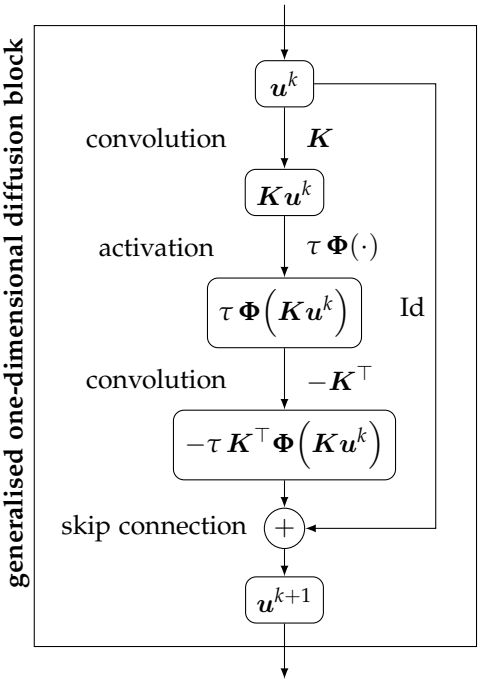


Figure 7.1: Generalised one-dimensional diffusion block for an explicit diffusion step (7.9) with flux function Φ, time step size τ, and a discrete derivative operator K.

This yields the discrete evolution equation

$$\frac{u^{k+1} - u^k}{\tau} = -K^\top \Phi\left(Ku^k\right). \tag{7.8}$$

Solving this expression for the new signal u^{k+1} yields the explicit scheme

$$u^{k+1} = u^k - \tau K^\top \Phi\left(Ku^k\right). \tag{7.9}$$

In contrast to the previous chapter, we are now free to choose the convolution filters K without being restricted to specific first order operators.

Thus, we can generalise our findings in Theorem 1 to trainable ResNet blocks.

Theorem 3 (Diffusion-inspired ResNets)**.** *An explicit step* (7.9) *of the generalised higher order diffusion scheme* (7.1) *can be expressed as a residual block* (7.5) *by*

$$\varphi_1 = \tau\,\Phi, \quad \varphi_2 = \mathrm{Id}, \quad W_1 = K, \quad W_2 = -K^\top, \tag{7.10}$$

with the bias vectors b_1, b_2 *set to* $\mathbf{0}$.

We call a ResNet block of this form a *generalised one-dimensional diffusion block*. Figure 7.1 visualises such a block in the form of a graph.

This insight confirms the observations from the previous chapter: The rescaled flux function $\tau\boldsymbol{\Phi}$ serves as the sole activation function φ_1, and the skip connection naturally arises from the discretisation of the temporal derivative. However, an additional structural property now becomes more apparent: The filters exhibit a negated symmetric filter structure $\boldsymbol{W}_2 = -\boldsymbol{W}_1^\top$. This is a natural consequence of the gradient flow structure of the diffusion process, and leads to provable stability guarantees for ResNets with such a filter structure, as we show in the following.

7.2.1 *Stability for Symmetric ResNets*

The structural connection between explicit schemes and ResNets allows us to transfer classical stability [101] and well-posedness [369] results of diffusion evolutions to a specific residual network architecture. By relaxing the maximum-minimum stability from the previous chapter to stability in the Euclidean norm, we obtain a good balance between degrees of freedom in the ResNet block and mathematical guarantees.

To this end, we consider ResNets which chain generalised diffusion blocks. Since we show that a key to stability of these networks is the symmetric filter structure, we refer to these architectures as *symmetric residual networks (SymResNets)*, following [319, 412].

For these networks, we prove Euclidean stability and well-posedness. Euclidean stability guarantees that the Euclidean norm of the signal is nonincreasing in each iteration, i.e. $\|\boldsymbol{u}^{k+1}\|_2 \leq \|\boldsymbol{u}^k\|_2$. Well-posedness ensures that the network output is a continuous function of the input data.

Theorem 4 (Euclidean Stability of Symmetric Residual Networks)**.** *Consider a symmetric residual network chaining any number of diffusion blocks (7.9) with convolutions represented by a convolution matrix $\boldsymbol{K}$ and activation function $\tau\boldsymbol{\Phi}$. Moreover, assume that the activation function can be expressed as a diffusion flux function $\Phi(s) = g(s^2)\, s$ and has a finite Lipschitz constant L. Then the symmetric residual network is well-posed and stable in the Euclidean norm if*

$$\tau \leq \frac{2}{L\|\boldsymbol{K}\|_2^2}. \tag{7.11}$$

Here, $\|\cdot\|_2$ denotes the spectral norm which is induced by the Euclidean norm.

Proof. The activation function $\varphi(s)$ can be expressed in terms of a diffusivity function by

$$\varphi(s) = \tau\Phi(s) = \tau g(s^2)\, s. \tag{7.12}$$

Thus, its application is equivalent to a rescaling with a diagonal matrix $\boldsymbol{G}(\boldsymbol{u}^k)$ with $g((\boldsymbol{K}\boldsymbol{u}^k)_i^2)$ as i-th diagonal element. Therefore, we can write (7.9) as

$$\boldsymbol{u}^{k+1} = \left(\boldsymbol{I} - \tau \boldsymbol{K}^\top \boldsymbol{G}(\boldsymbol{u}^k)\boldsymbol{K}\right)\boldsymbol{u}^k. \tag{7.13}$$

At this point, well-posedness follows directly from the continuity of the operator $\boldsymbol{I} - \tau \boldsymbol{K}^\top \boldsymbol{G}(\boldsymbol{u}^k)\boldsymbol{K}$, as the diffusivity g is assumed to be smooth [369].

We now show that the time step size restriction (7.11) guarantees that the eigenvalues of the operator always lie in the interval $[-1, 1]$. Then the explicit step (7.9) constitutes a contraction mapping which in turn guarantees Euclidean stability.

As the spectral norm is sub-multiplicative, we can estimate the eigenvalues of $\boldsymbol{K}^\top \boldsymbol{G}(\boldsymbol{u}^k)\boldsymbol{K}$ for each matrix separately. Since g is non-negative, the diagonal matrix $\boldsymbol{G}$ is positive semidefinite. The maximal eigenvalue of $\boldsymbol{G}$ is then given by the supremum of g, which can be bounded by the Lipschitz constant L of Φ:

$$L = \sup_s \left|\Phi'(s)\right| = \sup_s \left|g(s^2) + 2s^2 g'(s^2)\right| \geq \sup_s \left|g(s^2)\right|. \tag{7.14}$$

Consequently, the eigenvalues of $\boldsymbol{K}^\top \boldsymbol{G}(\boldsymbol{u}^k)\boldsymbol{K}$ lie within the interval $\left[0, \tau L \|\boldsymbol{K}\|_2^2\right]$. Then the operator $\boldsymbol{I} - \tau \boldsymbol{K}^\top \boldsymbol{G}(\boldsymbol{u}^k)\boldsymbol{K}$ has eigenvalues in $\left[1 - \tau L \|\boldsymbol{K}\|_2^2, 1\right]$, and the condition

$$1 - \tau L \|\boldsymbol{K}\|_2^2 \geq -1 \tag{7.15}$$

leads to the bound (7.11). □

Similar results have been obtained recently in [309, 319, 412], albeit with alternative justifications. However, our unique diffusion interpretation allows novel design concepts for CNNs such as nonmonotone activation functions (cf. Chapter 6).

7.2.2 *How General is Our Stability Result?*

While our focus on explicit diffusion schemes appears restrictive at first glance, our stability result is more general.

The fact that we use discrete differential operators as convolutions is no restriction, since any convolution matrix can be expressed as a weighted combination of discrete differential operators. Moreover, our proof does not even require a convolutional matrix structure.

A key requirement for stability is the symmetric structure $\boldsymbol{W}_2 = -\boldsymbol{W}_1^T$.

The symmetric convolution structure is an important structural difference to the original ResNet formulation [173]. It does not only yield a stable network, but also allows to reduce the amount of trainable parameters by 50%, since inner and outer convolution share their weights. Our experiments in Section 7.6 show that this is not necessarily a trade-off: The symmetric residual networks provide a stable alternative with the same performance as standard ResNets for the same amount of parameters when using few trainable network layers.

Moreover, the requirement of using a flux function as an activation function can be relaxed. As we have shown, one only requires the diagonal matrix $\boldsymbol{G}$ to be positive semidefinite. While this is naturally fulfilled for a diffusion flux function, other activations also adhere to this constraint. For example, the ReLU function multiplies positive arguments with one and negative ones with zero, yielding a binary positive semidefinite matrix $\boldsymbol{G}$. Thus, using the ReLU instead of a diffusion flux does not affect stability. This shows that diffusion algorithms inspire general, sufficient design criteria for stable networks.

In particular, we do not require any assumptions on the monotonicity of the activation function, in contrast to the results of Ruthotto and Haber [319].

7.2.3 *Enforcing Stability in Practice*

While the stability criterion (7.11) can be computed on the fly already during the training process of the network, evaluating the spectral radius of the operator $\boldsymbol{K}$ is costly. To this end, we suggest a simple rescaling to turn the stability bound (7.11) into an a priori criterion.

For a symmetric residual network with a single channel, one can directly use Gershgorin's circle theorem [143] to bound the maximum absolute eigenvalue of $\boldsymbol{K}$. More precisely, the eigenvalues of $\boldsymbol{K}$ lie in the union of circles around the diagonal entries $k_{i,i}$ with radii $r_i = \sum_{j \neq i} |k_{i,j}|$ corresponding to the absolute sums of the off-diagonal values. Thus, the maximal absolute eigenvalue of $\boldsymbol{K}$ is bounded by the largest absolute row sum of $\boldsymbol{K}$. If we simply rescale both inner and outer convolutions by this sum, we can guarantee $\|\boldsymbol{K}\|_2^2 \leq 1$. Then the stability condition (7.11) transforms into $\tau \leq \frac{2}{L}$. Since the Lipschitz constant L of the activation is known a priori, this simple rescaling allows to constrain the time step size to a fixed value, while not affecting the expressive power of the network.

However, most networks in practice are not concerned with only a single channel. To this end, we extend our stability result to symmetric ResNets with multiple channels.

For a diffusion block operating on a signal with C channels, the matrix $\boldsymbol{K}$ is a $C \times C$ block convolution matrix. As long as the transposed structure is realised, this is not problematic for the stability proof.

An extension of Gershgorin's circle theorem to block matrices [357] states that the eigenvalues of $\boldsymbol{K}$ lie in the union of circles which are centred around the eigenvalues of the diagonal blocks. The radii of the circles are given by the sum of the spectral norms of the off-diagonal blocks. If we rescale each block matrix as in the single channel case, we simply need to additionally divide the operator $\boldsymbol{K}$ by $\sqrt{C}$ to ensure that $\|\boldsymbol{K}\|_2^2 \leq 1$. With this, we obtain the same a priori criterion as in the single channel case.

This strategy constitutes an instance of the popular weight normalisation technique [323], and related spectral normalisations have shown to be successful for improving the performance and convergence speed of the training process [62, 157].

Our translation of explicit schemes is an example of a simple, direct correspondence which in turn allows for multiple novel insights. In the following, we explore variants of explicit schemes as well as implicit schemes which inspire changes to the skip connections of the symmetric ResNets, leading to more efficient architectures.

7.3 THE VALUE OF SKIP CONNECTIONS

So far, we have seen that the temporal discretisation of an explicit scheme naturally leads to skip connections. This, however, is just one of the many justifications for their use. Since their proposal, skip connections [173] have been adapted into numerically inspired networks in many different forms; see e.g. [191, 229, 240, 275, 416].

This motivates us to explore several other numerical algorithms which justify several types of skip connections from a numerical perspective. We explore unconditionally stable schemes, acceleration strategies for explicit schemes, and fixed point iterations for implicit schemes.

7.3.1 *Du Fort–Frankel Schemes*

While the classical explicit scheme (7.9) is only conditionally stable, there exist absolutely stable schemes which are still explicit. These schemes are not that popular in practice since they trade unconditional stability for conditional consistency. However, we will see that this is not problematic from the perspective of learning.

Du Fort and Frankel [110] propose to change the temporal discretisation of the explicit scheme (7.9) to a central difference and introduce a stabilisation term on the right hand side, corresponding to an approximation of $\partial_{tt} u$. A Du Fort–Frankel scheme for the generalised diffusion (7.1) evolution can be written as

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^{k-1}}{2\tau} = -\boldsymbol{K}^\top \boldsymbol{\Phi}\left(\boldsymbol{K}\boldsymbol{u}^k\right) - \alpha\left(\boldsymbol{u}^{k+1} - 2\boldsymbol{u}^k + \boldsymbol{u}^{k-1}\right), \tag{7.16}$$

where a positive constant α controls the influence of the stabilisation term.

Solving this scheme for $\boldsymbol{u}^{k+1}$ yields

$$\boldsymbol{u}^{k+1} = \frac{4\tau\alpha}{1+2\tau\alpha}\left(\boldsymbol{u}^k - \frac{1}{2\alpha}\boldsymbol{K}^\top \boldsymbol{\Phi}\Big(\boldsymbol{K}\boldsymbol{u}^k\Big)\right) + \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{u}^{k-1}. \tag{7.17}$$

For $\tau\alpha = \frac{1}{2}$, one obtains the explicit scheme (7.9).

The scheme involves the signals $\boldsymbol{u}^k$ and $\boldsymbol{u}^{k-1}$ at the current and the previous time level. The first term is nothing else than a rescaled diffusion block, where $\frac{1}{2\alpha}$ takes the role of the original time step size. Since the scalar factors $\frac{4\tau\alpha}{1+2\tau\alpha}$ and $\frac{1-2\tau\alpha}{1+2\tau\alpha}$ add up to one, this is simply an extrapolation of the result of an explicit step based on the signal at time level $k-1$.

If α is large enough, this scheme is unconditionally stable. Thus, one does not need to obey any stability condition, in contrast to the explicit case. Whereas classical proofs such as [156] consider only the linear case and typically work in the Fourier space, we are not aware of any proofs for the stability of nonlinear Du Fort–Frankel schemes. To this end, we prove stability of the nonlinear case in Appendix B.

However, this scheme is not unconditionally consistent. If the time step size τ is too large, the scheme (7.17) approximates a different PDE [110], namely a nonlinear variant of the telegrapher's equation. Such PDEs have also been used in image processing; see e.g. [297].

In the trainable setting, the conditional consistency is not an issue, but can even present a chance. It allows the network to learn a more suitable PDE for the problem at hand. In our experiments we show that indeed, the unconditional stability of the Du Fort–Frankel scheme can help to achieve better results when only few residual blocks are available.

This scheme can be realised with a small change in the original diffusion block from Figure 7.1 by adding an additional skip connection. The two skip connections are weighted by $\frac{4\tau\alpha}{1+2\tau\alpha}$ and $\frac{1-2\tau\alpha}{1+2\tau\alpha}$, respectively. The resulting architecture is visualised in Figure 7.2 as a *Du Fort–Frankel block*.

7.3.2 *Fast Semi-iterative Schemes*

Another numerical scheme which also leads to the same concept from a different motivation is based on acceleration strategies for explicit schemes. Hafner et al. [170] introduced fast semi-iterative (FSI) schemes to accelerate explicit schemes for diffusion processes; see e.g. Section 3.1.2.

In a similar manner as the Du Fort–Frankel schemes, FSI extrapolates the diffusion result at a fractional time step $k + \frac{\ell}{L}$ with the

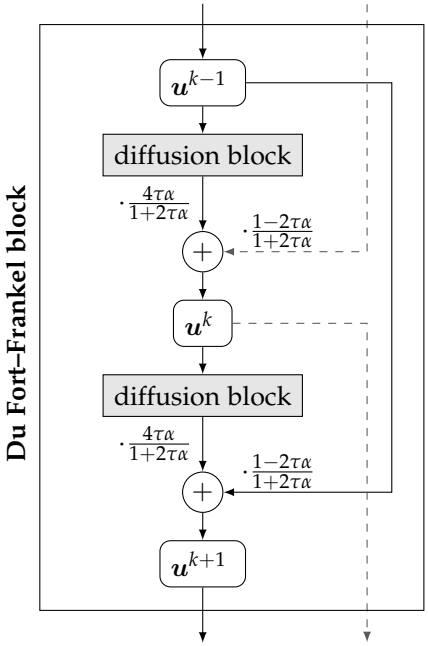


Figure 7.2: Du Fort–Frankel block for the acceleration of an explicit diffusion step (7.18) with extrapolation parameter α. The diffusion block employs a time step size of $\frac{1}{2\alpha}$.

previous fractional time step $k + \frac{\ell-1}{L}$ and a weight α_ℓ. For the explicit diffusion scheme (7.9), an FSI acceleration with cycle length L reads

$$\boldsymbol{u}^{k+\frac{\ell+1}{L}} = \alpha_\ell \left(\boldsymbol{u}^{k+\frac{\ell}{L}} - \tau \boldsymbol{K}^\top \boldsymbol{\Phi}\left(\boldsymbol{K} \boldsymbol{u}^{k+\frac{\ell}{L}} \right) \right) + (1-\alpha_\ell)\, \boldsymbol{u}^{k+\frac{\ell-1}{L}} \tag{7.18}$$

with fractional time steps $\ell = 0, \dots, L-1$ and extrapolation weights $\alpha_\ell := (4\ell+2)/(2\ell+3)$. One formally initialises with $\boldsymbol{u}^{k-\frac{1}{L}} := \boldsymbol{u}^k$.

The crucial difference to Du Fort–Frankel schemes is that FSI uses time-varying extrapolation coefficients instead of fixed ones. These coefficients are motivated by a box filter factorisation and allow a cycle to realise a super time step of size $\frac{L(L+1)}{3}\tau$. Thus, with one cycle involving L steps, one reaches a super step size of $\mathcal{O}(L^2)$ rather than $\mathcal{O}(L)$. This explains its remarkable efficiency [170].

Even though Du Fort–Frankel and FSI schemes have fundamentally different motivations, they lead to the same architectural changes, where additional weighted skip connections help to realise acceleration strategies. This is in line with observations in the CNN literature; see e.g. [240, 275]. We visualise this concept at the example of an *FSI block* in Figure 7.3.

FSI and Du Fort–Frankel schemes are just two representatives of a large class of extrapolation strategies; see e.g. [267, 287, 362]. The ongoing success of using momentum methods for training [317, 347] and constructing [240, 275, 416] neural networks warrants an extensive investigation of these strategies in both worlds.

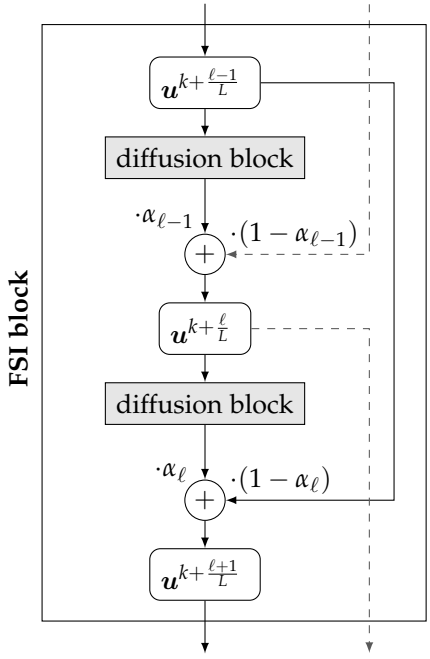


Figure 7.3: FSI block realising the acceleration of an explicit diffusion step (7.18) with time-varying extrapolation parameters α_ℓ.

7.3.3 *Implicit Schemes*

So far, we have investigated variants of explicit schemes and their neural counterparts. However, implicit discretisations constitute another important solver class. We now show that such a discretisation of the generalised diffusion equation can be connected to a recurrent neural network (RNN) (cf. Section 3.4.2). At the same time, this translation inspires yet another way of leveraging skip connections.

A fully implicit discretisation of the diffusion equation (7.1) is given by

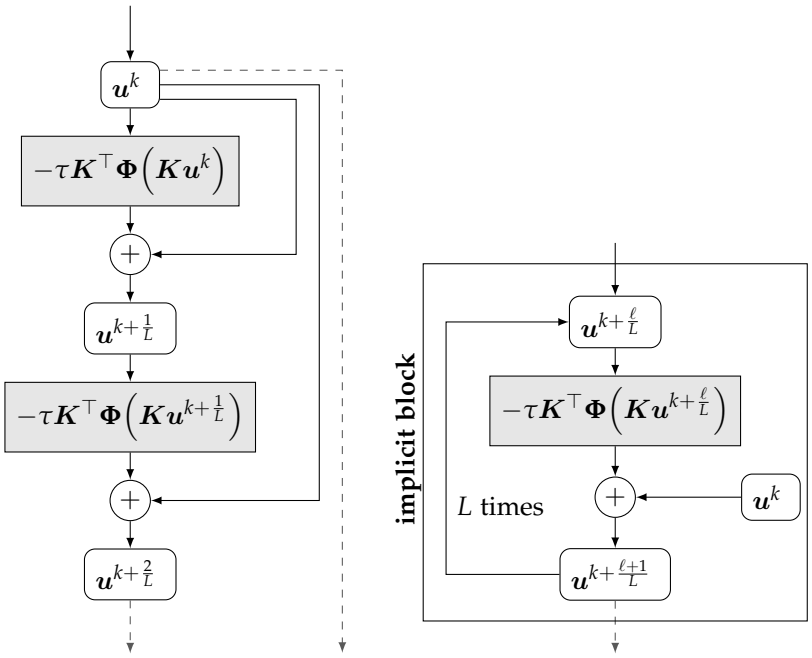
$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \tau \boldsymbol{K}^\top \boldsymbol{\Phi}\left(\boldsymbol{K}\boldsymbol{u}^{k+1}\right). \tag{7.19}$$

The crucial difference as opposed to the explicit scheme lies in using the new signal $\boldsymbol{u}^{k+1}$ within the flux function $\boldsymbol{\Phi}$. This yields a nonlinear system of equations, which we solve by means of a fixed point iteration with a cycle of length L:

$$\boldsymbol{u}^{k+\frac{\ell+1}{L}} = \boldsymbol{u}^k - \tau \boldsymbol{K}^\top \boldsymbol{\Phi}\left(\boldsymbol{K}\boldsymbol{u}^{k+\frac{\ell}{L}}\right), \tag{7.20}$$

where $\ell = 0, \dots, L-1$, and where we assume that τ is sufficiently small to yield a contraction mapping.

For $L = 1$, we obtain the explicit scheme (7.9) with its ResNet interpretation. For larger L, however, different skip connections arise. They connect the layer at time step k with all subsequent layers at steps $k + \frac{\ell}{L}$ with $\ell = 0, \dots, L-1$.



(a) Unrolled implicit scheme with skip connections.

(b) RNN interpretation.

Figure 7.4: Two equivalent representations of an implicit diffusion scheme (7.20).

There are two possible ways of interpreting this type of connection. One option is to regard this as a consequent extension of the extrapolation idea of FSI and Du Fort–Frankel schemes. Instead of only connecting a node to its two successors, the fixed point iteration above links a node to L of its successors. Similar ideas have been used in the popular DenseNet architecture [191], where each layer is connected to all subsequent ones. We present a corresponding visualisation in Figure 7.4(a). Another option is to interpret the repeated connection to $\boldsymbol{u}^k$ as a feedback loop, which in turn is closely related to a recurrent neural network (RNN) architecture [188]. This variant is depicted in Figure 7.4(b).

In their trainable nonlinear diffusion model, Chen and Pock [75] proposed a similar architecture. However, they explicitly supplement the diffusion process with an additional reaction term which results from the data term of the energy. Our feedback term is a pure numerical phenomenon of the fixed point solver.

We see that skip connections can implement a number of successful numerical concepts: forward difference approximations of the temporal derivative in explicit schemes, extrapolation steps to accelerate them e.g. via FSI or Du Fort–Frankel schemes, and recurrent connections within fixed point solvers for implicit schemes.

7.4 REVIEW: MULTIGRID SOLVERS AND U-NETS

Although the previously investigated numerical strategies and their neural counterparts can be efficient, they work on a single scale: The signal is considered at its original resolution at all points in time. However, using different signal resolutions in a clever combination can yield even higher efficiency.

This is the core idea of the large class of multigrid approaches [53, 54, 168]. They belong to the most efficient numerical methods for PDE-related problems and have been successfully applied to various tasks such as image denoising [57], inpainting [246], video compression [219], and image sequence analysis [59].

On the CNN side, architectures that work on multiple resolutions of the signal have become very successful. In particular, the shape of the popular U-net architecture [306] suggests that there is a structural connection between multigrid and CNN concepts.

By translating multigrid solvers into a U-Net architecture, we show that this is indeed the case, which serves as a basis for explaining the remarkable success of U-nets. Since both underlying concepts are not self-explanatory, we review them in the following before connecting them in the next section.

7.4.1 *Multigrid Solvers for Nonlinear Systems*

Multigrid methods [53, 54, 168] are designed to accelerate the convergence speed of standard numerical solvers such as the Jacobi or the Gauss–Seidel method [320]. These solvers attenuate high-frequent components of the residual error very quickly, while low-frequent error components are damped slowly. This causes a considerable drop in convergence speed after a few iterations.

Multigrid methods remedy this effect by transferring the low-frequent error to a coarser grid, transforming them into high-frequent components. This allows a coarse grid solver to attenuate them more efficiently. By correcting the fine grid approximation with coarse grid results, convergence speed can be significantly improved.

In the following, we review the so-called full approximation scheme (FAS) [53] for a nonlinear system of equations. We consider a two-grid cycle as the basic building block of more complex multigrid solvers.

We are interested in solving a nonlinear system of equations of the form

$$\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{b}, \tag{7.21}$$

with a nonlinear operator $\boldsymbol{A}$ and a right hand side vector $\boldsymbol{b}$ for an unknown coefficient vector $\boldsymbol{x}$.

The two-grid FAS involves two grids with different step-sizes: a fine grid of size h, and a coarse grid of size $H > h$. We denote the

respective grid by superscripts. The following six steps describe the two-grid FAS:

1. **Presmoothing Relaxation:** A standard solver is applied to the fine grid system $\boldsymbol{A}^h(\boldsymbol{x}^h) = \boldsymbol{b}^h$. Given an initialisation $\boldsymbol{x}_0^h$, it produces an approximation $\tilde{\boldsymbol{x}}^h$ to the solution with a reduced high frequency error.

2. **Restriction:** In order to approximate low-frequent components of the error more efficiently, one transfers the problem to the coarse grid with the help of a restriction operator $\boldsymbol{R}^{h\to H}$. One restricts both the residual $\boldsymbol{r}^h = \boldsymbol{A}^h(\boldsymbol{x}^h) - \boldsymbol{b}^h$ as well as the current approximation $\tilde{\boldsymbol{x}}^h$ to the coarse grid. One obtains two parts of the right hand side for the coarse grid problem: One part $\boldsymbol{b}^H = \boldsymbol{R}^{h\to H}\boldsymbol{r}^h$ which is used directly, and a second one which we denote by $\boldsymbol{y}^H = \boldsymbol{R}^{h\to H}\tilde{\boldsymbol{x}}^h$ serving as the argument for the nonlinear operator $\boldsymbol{A}^H$. The coarse grid problem then reads

$$\boldsymbol{A}^H\left(\boldsymbol{x}^H\right) = \boldsymbol{A}^H\left(\boldsymbol{y}^H\right) + \boldsymbol{b}^H. \tag{7.22}$$

   If we express the desired solution $\boldsymbol{x}^H$ in terms of the error $\boldsymbol{e}^H$ by $\boldsymbol{x}^H = \boldsymbol{y}^H + \boldsymbol{e}^H$, then we see that this equation is solved for the full approximation rather than the error alone, in contrast to a linear multigrid scheme. Hence, this scheme is called the full approximation scheme. If the operator $\boldsymbol{A}$ is linear, FAS reduces to a linear multigrid scheme.

   A standard choice for $\boldsymbol{R}^{h\to H}$ is a simple averaging. However, finding suitable restriction operators is a difficult task, which motivates researchers to even learn such operators; see e.g. [159, 209].

3. **Coarse Grid Computation:** Solving the coarse grid problem with a standard solver produces an error approximation $\tilde{\boldsymbol{x}}^H$.

4. **Prolongation:** The approximation on the coarse grid needs to be transferred to the fine grid again.To this end, one applies a prolongation operator $\boldsymbol{P}^{H\to h}$.

   Since a coarse grid solution $\tilde{\boldsymbol{x}}^H$ is a full approximation, we need to compute the approximation to the error by $\tilde{\boldsymbol{x}}^H - \boldsymbol{y}^H$. This error approximation is then transferred to the fine grid via $\boldsymbol{P}^{H\to h}$.

   A standard choice for $\boldsymbol{P}^{H\to h}$ is a nearest neighbour interpolation, but as for the restriction operator, finding a good prolongation operator is not easy.

5. **Correction:** The fine grid approximation $\tilde{\boldsymbol{x}}^h$ is corrected with the upsampled coarse grid error approximation $\boldsymbol{P}^{H\to h}\left(\tilde{\boldsymbol{x}}^H - \boldsymbol{y}^H\right)$ to produce a new approximation

$$\tilde{\boldsymbol{x}}^h_{\text{new}} = \tilde{\boldsymbol{x}}^h + \boldsymbol{P}^{H\to h}\left(\tilde{\boldsymbol{x}}^H - \boldsymbol{y}^H\right). \tag{7.23}$$

6. **Postsmoothing Relaxation:** Finally, one applies another solver on the fine grid to smooth high frequent errors which have been introduced by the correction step.

The two-grid FAS will serve as the starting point for translating multigrid concepts into the a U-net formulation. It is more general than the linear connections from our conference publication [10].

7.4.2 *Review: U-nets*

In Section 3.4.2 we have already introduced the U-net as a network architecture performing repeated down- and upsampling of input data. In the following, we introduce an abstract mathematical formulation of a simple U-net to compare it to a multigrid cycle.

A two-level U-net with fine grid size h and coarse grid size H has the following structure:

1. An input signal $\boldsymbol{f}^h$ is fed into a series of general convolutional layers which we denote by $\boldsymbol{C}_1^h(\cdot)$. The resulting output signal is denoted by $\tilde{\boldsymbol{f}}^{\boldsymbol{h}} = \boldsymbol{C}_1^h(\boldsymbol{f}^h)$.

   Originally, these layers are assumed to be feed-forward convolutional layers, but they can also be replaced by any other suitable layer type such as residual layers.

2. The fine grid signal $\tilde{\boldsymbol{f}}^{\boldsymbol{h}}$ is transferred to a coarser grid with a restriction operator $\boldsymbol{R}^{h\to H}$, yielding a coarse grid signal $\boldsymbol{f}^H = \boldsymbol{R}^{h\to H}\boldsymbol{f}$.

3. On the coarse grid, another series of convolutional layers $\boldsymbol{C}^H(\cdot)$ is applied to the signal, yielding a modified coarse grid signal $\tilde{\boldsymbol{f}}^H = \boldsymbol{C}^H(\boldsymbol{f}^H)$.

4. The modified coarse grid signal is upsampled with a prolongation operator $\boldsymbol{P}^{H\to h}$.

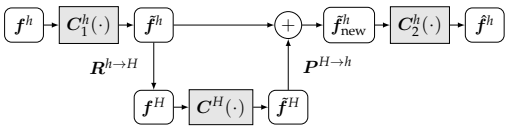
5. With the help of a skip connection, the modified fine grid signal $\tilde{\boldsymbol{f}}^h$ and the upsampled coarse grid signal $\boldsymbol{P}^{H\to h}\tilde{\boldsymbol{f}}^H$ are added together. This produces a new fine grid signal $\tilde{\boldsymbol{f}}^h_{\text{new}}$.

6. Lastly, another series of convolutional layers $\boldsymbol{C}_2^h(\cdot)$ is applied to the new fine grid signal, producing the final output signal $\hat{\boldsymbol{f}}^h$.

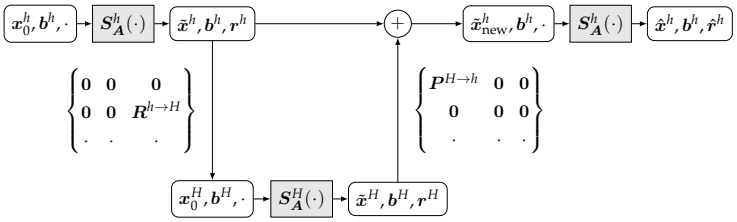
We visualise this architecture in Figure 7.5(a).

7.5 FROM MULTIGRID TO U-NETS

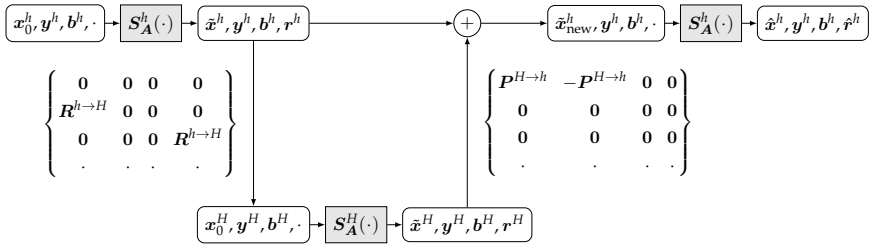
Now we show how one can express FAS in terms of a U-net architecture. For our U-net, we use multiple network channels which carry the variables required by FAS. Even though not all variables are used



(a) U-net architecture for an input $\boldsymbol{f}^h$.



(b) Linear two-grid cycle in the form of a U-net utilising three network channels. Besides the iteration variable $\boldsymbol{x}$, the network tracks variables $\boldsymbol{b}$ as the system's right hand side, as well as the residual $\boldsymbol{r}$.



(c) FAS two-grid cycle in the form of a U-net utilising four network channels. In addition to the linear scheme, the network tracks variables $\boldsymbol{b}$ and $\boldsymbol{y}$ as linear and nonlinear parts of the system's right hand side.

Figure 7.5: Architectures for a general U-net (a), a linear two-grid cycle (b) and an FAS two-grid cycle (c).

at each point in the network, we keep the channel number consistent for the sake of simplicity.

We track the FAS variables in dedicated channels only for didactic reasons, as a direct translation shows that a U-net architecture is sufficient for representing FAS. When practically implementing FAS, this overhead can be omitted.

Firstly, let us assume that we are given suitable solvers $\boldsymbol{S}_{\boldsymbol{A}}^h(\cdot), \boldsymbol{S}_{\boldsymbol{A}}^H(\cdot)$ for the nonlinear operators on the fine and coarse grid, respectively. To be able to use the two-grid cycle as a recursive building block, we assume that all solvers approximate solutions for nonlinear systems of the form $\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{A}(\boldsymbol{y}) + \boldsymbol{b}$, regardless of the grid.

To this end, we always keep track of the iteration variable $\boldsymbol{x}$, the nonlinear right hand side $\boldsymbol{y}$, and the linear right hand side $\boldsymbol{b}$. In addition, we track the residual $\boldsymbol{r}$. By appropriately modifying these variables, we can ensure that the solvers always act on the desired system, despite having a common specification.

1. **Presmoothing Relaxation:** The first instance of the fine grid solver obtains an initial iteration variable $\boldsymbol{x}_0^h$, which can be $\mathbf{0}$ or a more sophisticated guess. Since the first solver is supposed to solve $\boldsymbol{A}^h(\boldsymbol{x}^h) = \boldsymbol{b}^h$, we simply set $\boldsymbol{y}^h = \mathbf{0}$. In addition, we provide the linear right hand side $\boldsymbol{b}^h$. A residual is not needed as an input.

   The solver produces a preliminary approximation $\tilde{\boldsymbol{x}}^h$, and passes the right hand side components $\boldsymbol{y}^h$ and $\boldsymbol{b}^h$ through without changes. It also computes the residual $\boldsymbol{r}^h = \boldsymbol{b}^h - \boldsymbol{A}^h(\tilde{\boldsymbol{x}}^h)$ as an additional output.

2. **Restriction:** As the downsampling is now explicitly concerned with four channels, the corresponding operator in our U-net is a 4×4 block matrix. We apply the multigrid restriction operator only to certain channels.

   The coarse grid initialisation $\boldsymbol{x}_0^H$ can be set to $\mathbf{0}$, taking no information from the fine grid. The coarse, nonlinear right hand side $\boldsymbol{y}^H = \boldsymbol{R}^{h\to H}\tilde{\boldsymbol{x}}^h$ is given by the downsampled fine grid approximation. The corresponding linear right hand side $\boldsymbol{b}^H = \boldsymbol{R}^{h\to H}\boldsymbol{r}^h$ is the downsampled residual.

   In contrast to our linear correspondences in [10], the restriction step in FAS fits a U-net interpretation even better, since the approximation $\tilde{\boldsymbol{x}}^h$ itself is restricted, as is the case in the U-net.

3. **Coarse Grid Computation:** The coarse solver follows the same specification as the fine grid one. However, since $\boldsymbol{y}^H$ is not set to $\mathbf{0}$ at this point, the coarse solver actually solves the desired system $\boldsymbol{A}^H(\boldsymbol{x}^H) = \boldsymbol{A}^H(\boldsymbol{y}^H) + \boldsymbol{b}^H$. It produces a coarse approximation $\tilde{\boldsymbol{x}}^H$ and a residual $\boldsymbol{r}^H$, while leaving the right hand side components unchanged.

4. **Prolongation:** The upsampling step allows to prepare the coarse grid variables in such a way that the skip connection automatically performs the correct additions.

   The first row of the matrix operator ensures that we upsample the correction $\boldsymbol{P}^{H\to h}\tilde{\boldsymbol{x}}^H - \boldsymbol{P}^{H\to h}\boldsymbol{y}^H$. Note that this is equivalent to the FAS formulation $\boldsymbol{P}^{H\to h}(\tilde{\boldsymbol{x}}^H - \boldsymbol{y}^H)$ if the prolongation operator is linear. This is no limitation, however, since for the nonlinear case we can require the solvers to directly output the difference $\tilde{\boldsymbol{x}}^H - \boldsymbol{y}^H$.

   The right hand side components $\boldsymbol{y}^H$ and $\boldsymbol{b}^H$ are not used in the upsampling, as the fine grid right hand side is supposed to be passed on. The same holds for the residual, as it is not relevant to the second fine grid solver. It is only needed in case one adds another coarser level to the cycle.

5. **Correction:** In the correction step, the fine approximation $\tilde{\boldsymbol{x}}^h$ is appropriately corrected, and the fine grid right hand side components $\boldsymbol{y}^h$ and $\boldsymbol{b}^h$ are forwarded.

6. **Postsmoothing Relaxation:** Another instance of the fine grid solver solves the problem $\boldsymbol{A}^h(\boldsymbol{x}^h) = \boldsymbol{b}^h$. The nonlinear part $\boldsymbol{y}^h$ of the right hand side is still set to $\mathbf{0}$, ensuring that the correct system is solved.

The resulting architecture is visualised in Figure 7.5(c). For the sake of completeness, we also depict a linear multigrid cycle in its U-net formulation in Figure 7.5(b). To this end, one only needs to drop the nonlinear right hand side $\boldsymbol{y}$ and adapt the operators accordingly.

This shows that U-nets share essential structural properties with multigrid methods. In particular, employing multiple image resolutions connected through pooling and upsampling operations, as well as horizontal skip connections which realise correction steps are the keys for the success of both methods. This leads us to believe that at their core, U-nets realise a sophisticated multigrid strategy.

7.5.1 *V-Cycles, W-Cycles and Full Multigrid*

Our connections between two-grid FAS and U-nets are the basic building block for more advanced multigrid strategies.

So-called V-cycles arise from recursively stacking the two-grid FAS. Moreover, W-cycles can be built by concatenating several V-cycles. Optimising the depth and length of these cycles can lead to vast efficiency gains over direct solution strategies.

On the CNN side, the corresponding concept of U-nets with more levels as well as concatenations thereof is successful in practice: Typical U-nets work on multiple resolutions [306], and so-called stacked hourglass models [268] arise by concatenating multiple V-cycle architectures.

A full multigrid (FMG) strategy solves a problem on multiple grids by successively concatenating V- and W-cycles, usually starting at the coarsest grid and progressing towards the finest one. In our experiments in Section 7.6.2 we will construct a trainable FMG model based on the two-grid FAS network to approximate the solution of an inpainting problem. This shows that our model reduction of the full U-net is successful in practice and inspires new design strategies for U-nets.

7.6 EXPERIMENTAL EVALUATIONS

Let us now show that our findings are also of practical relevance. Our experiments are divided into two parts. First, we evaluate the

proposed symmetric ResNet architectures, along with their variations and nonmonotone activation functions for a denoising problem.

In a second experiment, we make use of our connections between multigrid and U-nets to learn an efficient solver for diffusion-based sparse inpainting, based on a trainable FAS architecture.

7.6.1 *Symmetric ResNets and Nonmonotone Activations*

Since we motivate our network designs through numerical algorithms for a diffusion problem, we start with an elementary comparison on a denoising problem. We deliberately choose a denoising problem, since it is a prime example of a well-posed problem, for which the presented numerical schemes can be easily applied.

In a second step, we refine the simple network structures to more and more complex ones, approaching the standard neural network design. This shows the extent to which our networks can compete with off-the-shelf ResNets.

EXPERIMENTAL SETUP We compare symmetric ResNets and their Du Fort–Frankel and FSI extensions with the original ResNet architecture [173]. As activation functions we allow the ReLU [266], Charbonnier [70], and rational Perona–Malik [279] activation functions.

The original ResNets train two filter kernels per ResNet block, along with two bias terms. The symmetric ResNets on the other hand only train one filter kernel per block, without any bias terms. We only consider kernels of width three. For maximal transparency, we do not use any additional optimisation layers such as batch normalisation.

When using Charbonnier and Perona–Malik activations, we always train the corresponding contrast parameter λ. The Du Fort–Frankel networks also learn the extrapolation parameter α, and the FSI networks train individual extrapolation parameters of each block.

In addition, all models train their numerical parameters such as time step size and extrapolation parameters. We restrict the time step size τ to our stability condition (7.11) to obtain a stable symmetric ResNet model. In the case of the Du Fort–Frankel extension we restrict the extrapolation parameter α to the bound in Appendix B, thus also yielding a stable scheme. For FSI, we restrict the extrapolation parameters α_ℓ to the range $[0, 2]$. This preserves the extrapolation character of the scheme. However, no stability theory is available in the case of learned extrapolation parameters.

We evaluate the networks on a synthetic dataset of one-dimensional signals which are piecewise affine, with jumps between the segments. This design highlights the ability of the different approaches to preserve signal discontinuities. The signals are of length 256 and are composed of linear segments that span between $\frac{1}{10}$ and $\frac{1}{2}$ of the signal length. Their values lie within the interval $[0, 255]$.

Finally, we add Gaussian noise of standard deviation $\sigma = 10$ to the signals, without clipping out of bounds values. This yields pairs of corrupted and ground truth signals. The training dataset contains 10000 such pairs, and the test and validation datasets contain 1000 pairs each. As a measure of denoising quality, we choose the peak-signal-to-noise ratio (PSNR), where higher values indicate better denoising performance.

For a fair comparison, we train all network configurations in the same fashion. We use the Adam optimiser [212] with a learning rate of 0.001 for at most 2000 training epochs, and choose the average mean square error (MSE) over the training dataset as an optimisation objective.

The filter weights are initialised according to a uniform random distribution with a range of $[-0.1, 0.1]$. The contrast parameters λ, the time step sizes τ, and the extrapolation weights α are initialised with fixed values of 15, 1.0, and 1.0, respectively. Out of several random initialisations, we choose the best performing one.

EVALUATION OF MODEL COMPONENTS We first evaluate the potential of the proposed network blocks on an individual level. To this end, we train the architectures for varying amounts of residual blocks. However, all blocks share their weights, and we also use only a single network channel.

This configuration is closest to the interpretation of explicit schemes, and it allows us to investigate the approximation qualities of the different architectures within a tightly controlled frame.

Figure 7.6 presents the denoising quality of the architectures in dependence of the number of residual blocks. Each plot is concerned with a different activation function.

Firstly, we compare the different network architectures. As the symmetric ResNet is guaranteeing stability and uses less than half of the parameters of the standard ResNet, it performs slightly worse. This is not surprising, since there is a natural trade-off between performance (high approximation quality) and stability, as is well-known in the field of numerical analysis. Nevertheless, when enough blocks are provided, the symmetric ResNet catches up to the standard one.

The acceleration methods of Du Fort–Frankel and FSI outperform the symmetric ResNet and yield comparable performance to the standard ResNet. The trainable extrapolation parameters help both methods to achieve better quality especially when not enough residual blocks are provided to reach a sufficient denoising result. This is in full accordance with our expectations. When enough steps are provided and no extrapolation is required, both methods are on par with the standard and symmetric ResNets.

A side-by-side comparison yields insights into the performance of different activation functions. We observe that the performance

of ReLU networks is only slightly better than classical linear diffusion [195]. This shows that the ReLU is not suited for our denoising problem, regardless of the network architecture. After as few as three network blocks, the improvement of deeper networks is only marginal.

In contrast, both the Charbonnier and the Perona–Malik activations are much more suitable. The nonmonotone Perona–Malik activation function yields the best denoising performance, as our diffusion interpretation suggests. When using the original ResNet with a diffusion-inspired activation function, tremendous performance gains in comparison to the ReLU activation can be achieved. This shows that in this experimental setting, the activation is the key to a good performance.

OPTIMALITY OF DIFFUSION PROCESSES Interestingly, the standard ResNet with a diffusion activation naturally learns a symmetric filter structure with biases close to zero.

For example, the ResNet with Perona–Malik activation, a grid size of $h = 1$, and 20 shared residual blocks learns kernels

$$\boldsymbol{k}_1 = (0.922, -0.917, 0.006)^\top \tag{7.24}$$

$$\boldsymbol{k}_2 = (0.051, 0.437, -0.489)^\top, \tag{7.25}$$

and biases $b_1 = 1.6 \cdot 10^{-1}$ and $b_2 = 1.2 \cdot 10^{-5}$. If we factor out the time step size limit of $\tau = 0.5$ for this setting from the outer kernel $\boldsymbol{k}_2$, we see that it transforms into $\widetilde{\boldsymbol{k}_2} = (0.102, 0.874, -0.978)^\top$. It becomes apparent that the kernels approximately fulfil the negated symmetric filter structure.

Moreover, the kernels closely resemble rescaled standard forward and backward difference discretisations. This is surprising, as a kernel of width three allows to learn derivative operators of second order, but a first order operator appears to yield already optimal quality.

This shows that in this constrained setting, second order diffusion processes are an optimal model which is naturally learned by a residual network.

TIME DYNAMIC CASE In a practical setting, the residual blocks typically do not share their weights, but train them independently. If the parameters evolve smoothly over the blocks, we can interpret this as an approximation of a time dynamic PDE model.

To investigate the performance of the proposed architectures in this setting, we train the parameters of each block individually, but enforce a certain smoothness between them. If the parameter vector of a block at time level k is given by $\boldsymbol{\theta}^k$, we add a regulariser

$$\beta \sum_{k=1}^{K} \tau \left(\frac{\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k-1}}{\tau} \right)^2 \tag{7.26}$$

to the loss function. Here, a smoothness parameter β controls the amount of smoothness between the blocks, with higher values of β

leading to smoother evolutions. This expression approximates the continuous temporal regulariser

$$\beta \int_0^T (\partial_t \boldsymbol{\theta}(t))^2 \, dt, \tag{7.27}$$

which enforces smoothness of the continuous parameter evolution $\boldsymbol{\theta}(t)$. The regularisation ensures that the learned filters change smoothly throughout the layers. This is essential for the numerical scheme to be consistent with the continuous limit case where the step size τ tends towards zero; see also [319].

For the residual network, where no time step size τ is learned explicitly, we set the time step size to the inverse of the number of blocks. This requires to use a different smoothness parameter β. We tune the smoothness parameters for all architectures such that their parameters exhibit similarly smooth evolutions over time. Numerical parameters such as time step size and extrapolation parameters are not affected by the regulariser.

Figure 7.7 presents the performance of time dynamic architectures with a single channel. We use $\beta = 5$ for the standard ResNets, and set $\beta = 10$ for all other architectures. In contrast to the previous comparisons, we now compare the denoising quality against the number of network parameters. This allows us to measure performance against model complexity.

For the symmetric ResNet we observe the same behaviour as in the setting without a temporal dynamic. The overall best performance still on par with the respective classical diffusion process for the Charbonnier and Perona–Malik activation functions.

However, the time dynamic allows this model to achieve better denoising quality for fewer blocks. For example, the symmetric ResNet with Perona–Malik activation and seven residual blocks can achieve a denoising quality of 36.51 dB if weights are shared, but already 37.08 dB with a regularised temporal dynamic. Similar observations for classical diffusion processes with a time dynamic diffusivity function can be found in [148]. Yet, this effectiveness comes at the price of additional parameters.

The Du Fort–Frankel and FSI networks allow for higher efficiency at the cost of more network parameters. Especially in the case of FSI, the trainable extrapolation parameters help to achieve significantly better performance when more residual blocks are provided. This is in accordance with observations in the literature [240].

The proposed architectures outperform the standard ResNet for the same amount of parameters when using Charbonnier and Perona–Malik activations. This shows that the model reduction to a symmetric convolution structure is indeed fruitful. Moreover, the ranking of activation functions remains the same in most cases.

None of the architectures significantly outperform the classical Perona–Malik diffusion process. This supports our claim that these

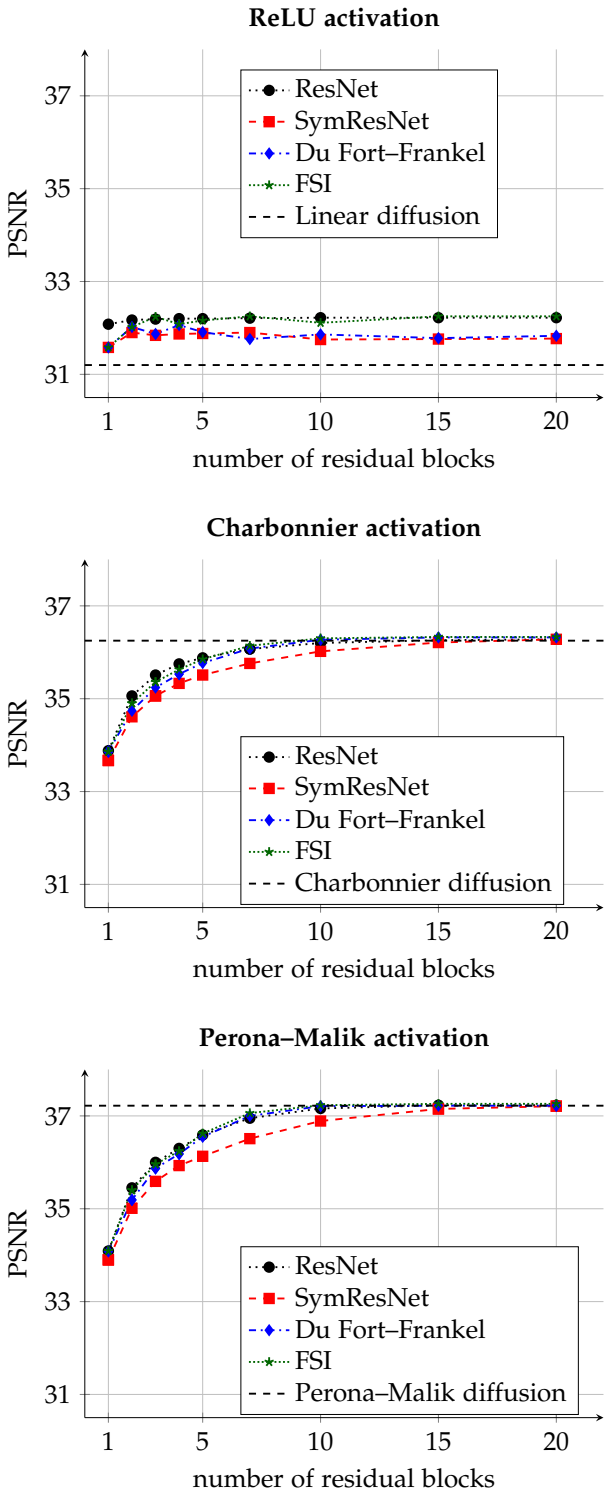


Figure 7.6: Denoising quality of network architectures with varying depth. We use a single channel, and weights between all blocks are shared. Each plot is concerned with a different activation function. Architectures with Perona–Malik activation perform best, while the ReLU activation is not suitable in this setting. Due to the tight network constraints, the architectures reproduce the performance of classical diffusion filters.

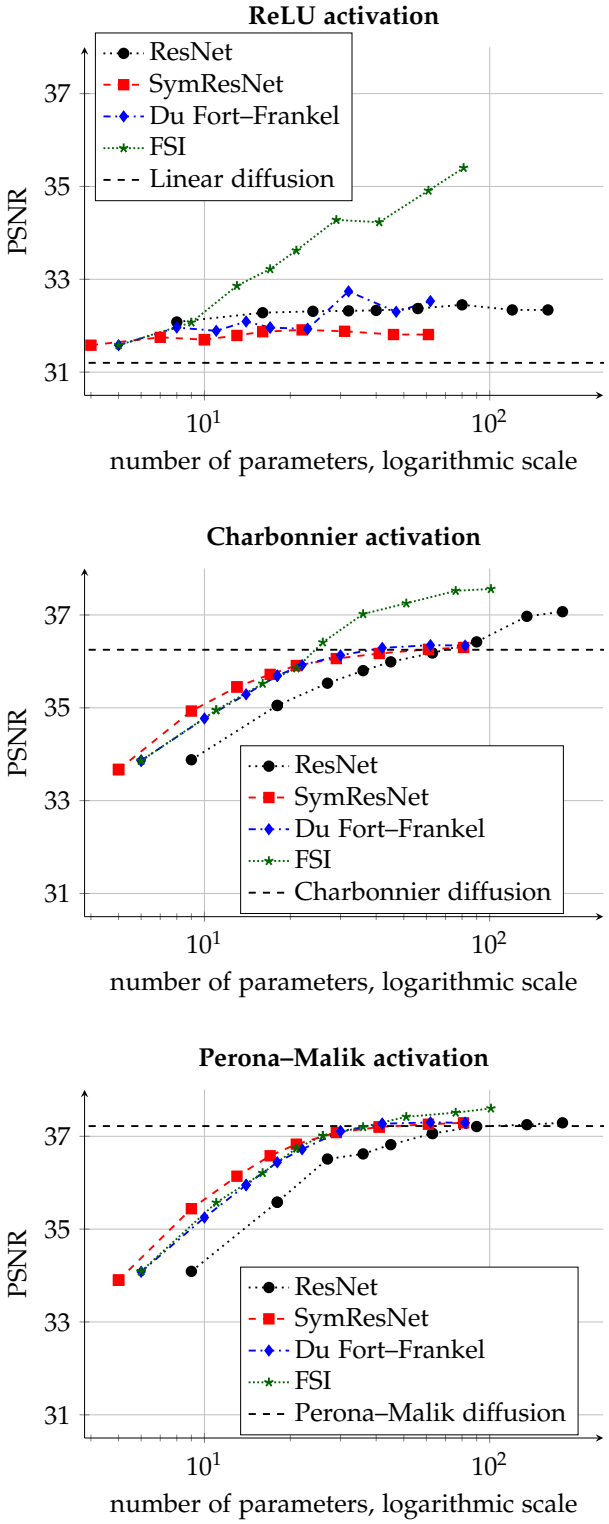


Figure 7.7: Denoising quality of network architectures with varying depth and a single channel. The parameters are smoothly changing between the residual blocks. Each plot is concerned with a different activation function. The proposed architectures can outperform the standard ResNet by saving a large amount of parameters.

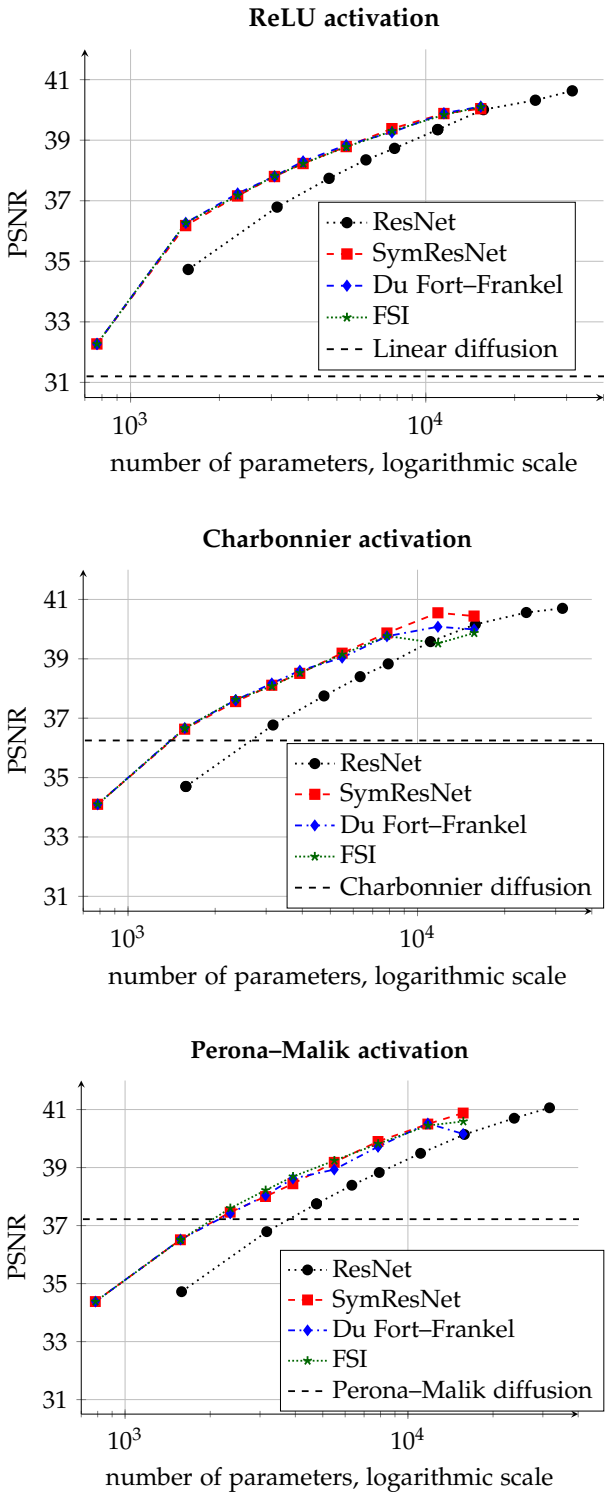


Figure 7.8: Denoising quality of network architectures with varying depth and $C = 16$ network channels. Each plot is concerned with a different activation function. The proposed architectures outperform the standard ResNet for the same amount of parameters. In this setting, the acceleration strategies are on par with the symmetric ResNet, and the margin between activation functions becomes smaller.

tightly constrained networks realise a numerical algorithm at their core. Different architectures can solve the problem with varying efficiency, but they converge towards the same result. This will only change when we allow for more flexibility within the network architecture, e.g. by using multiple network channels.

TOWARDS LARGER NETWORKS So far, we have only considered architectures with a single network channel. However, typical CNNs show their full potential when using multiple channels. In this case, the simplicity of the ReLU activation is compensated by a rich set of convolutions between channels.

We now extend our evaluations to architectures with multiple network channels. For simplicity, we leave the number of channels constant throughout the network. To this end, we copy the input signal into C channels. The output signal is computed as the average over the individual channel results. In between, we employ the proposed symmetric residual blocks which now use $C \times C$ block convolution matrices with the appropriate stability constraints. These convolutions can be interpreted as ensembles of differential operators which are applied to the signal. The channels are then activated individually, and convolved again with the adjoint counterparts of the differential operators.

To allow for maximum flexibility and performance, we remove the temporal parameter regularisation in this experiment.

The denoising performance of the networks are visualised in Figure 7.8 for the case of $C = 16$ channels.

This experiment is the first instance where all architectures significantly outperform their respective classical diffusion counterparts. The multi-channel architecture allows to approximate a more sophisticated denoising model, as information in the various channels is exchanged by means of multi-channel convolution operators.

All proposed architectures can still outperform the standard ResNet for the same amount of parameters. In this case, the extrapolation methods are on par with the symmetric ResNet. Interestingly, the ranking of activation functions remains the same, albeit with a much smaller margin. The symmetric ResNet with 20 residual blocks yields PSNR values of 40.04 dB, 40.44 dB and 40.88 dB for the ReLU, Charbonnier and Perona–Malik activations, respectively. We conclude that the more complex the network, the less the activation function matters for performance. On the contrary, this means that networks might be drastically reduced in size when trading network size for sophisticated activation function design.

7.6.2 *Learning a Multigrid Solver for Inpainting*

So far, it is not clear if our interpretation of the two-grid FAS is a reasonable model reduction of a full U-net. To prove that this interpretation is indeed of practical relevance, we show how it can be used to learn a multigrid solver for EED inpainting of images [135, 368, 378].

As a proof of concept, we construct a network implementing a full multigrid structure. We replace the prescribed nonlinear solvers by trainable feed-forward layers that learn to approximate EED inpainting by minimising the residual of the inpainting equation

$$(1 - c(\boldsymbol{x}))\boldsymbol{\nabla}^\top(\boldsymbol{D}\boldsymbol{\nabla}u) - c(\boldsymbol{x})(u - f) = 0, \tag{7.28}$$

where we use the Charbonnier diffusivity [70] within the diffusion tensor $\boldsymbol{D}$; see also Section 3.6.2.

EXPERIMENTAL SETUP Our trainable FMG architecture is designed as follows: Instead of prescribing nonlinear solvers on each grid, we employ a series of convolutional layers with trainable weights. The remainder of the architecture is fixed: We set the restriction operators to a simple averaging over a 2×2 pixel neighbourhood, and the prolongation operators to nearest neighbour interpolation.

Since FMG employs the same solver on each grid, we realise this idea also in our network by sharing the weights between all solvers for a specific grid. This drastically reduces the amount of parameters and incites that an iterated application of the solvers performs the correct computations.

Instead of training our network by minimizing a Euclidean loss between ground truth data and inpainting reconstructions, we use the absolute residual of a discretisation of the inpainting equation (7.28) as a loss function. This is closely related to the idea of deep energies [150], where one chooses a variational energy as a loss function.

Since we do not have such an energy available for EED inpainting, we resort to minimising the absolute residual of the associated Euler–Lagrange equation, which is given by (7.28). This guarantees that the trained architecture realises EED inpainting as efficiently as possible. To discretise the Euler–Lagrange equation, we employ the standard discretisation from [379]. This approach is generalisable to many more classical models beyond EED inpainting. In fact, it should be applicable to any elliptic PDE problem.

Whereas classical solution methods for the inpainting problem specify the known data $u = f$ on $\Omega \backslash K$ by means of Dirichlet boundary conditions, we leave it to the network to reproduce also the known data. We have found that this leads to a better approximation quality.

The inpainting masks consist of randomly sampled pixels with a density d as a percentage of the number of image pixels. Since the

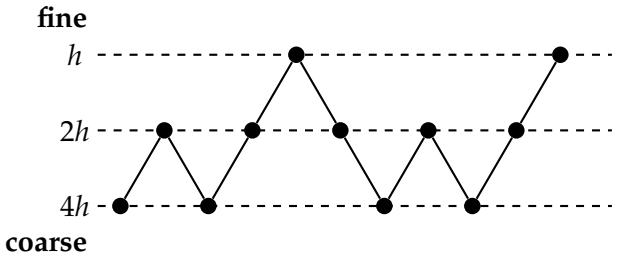


Figure 7.9: Visualization of the full multigrid strategy which we employ in our experiments. Dashed horizontal lines denote the three grids, and grids become coarser from top to bottom. Each circle denotes an instance of a solver.

masks are also required on coarser grids to compute the residual within the FMG architecture, we downsample them by putting defining a coarse pixel as known, if at least one pixel in the 2×2 cell on the fine grid is known.

We train the architecture on a subset of 1000 images of the ImageNet dataset [318] with the Adam optimiser [212] with standard settings.

EVALUATION OF THE FULL MULTIGRID NETWORK We construct a full multigrid network using three grids of size h, $2h$, and $4h$. The order in which the problem is solved on different grids is given by $[4h, 2h, 4h, 2h, h, 2h, 4h, 2h, 4h, 2h, h]$. This is the simplest FMG strategy that can be employed in a setting involving three grids and serves as a proof-of-concept architecture. We visualise this strategy in Figure 7.9. Thus, we employ 11 solvers, each using 12 feed-forward convolutional layers with 20 channels and ReLU activations. Weights are shared for solvers on the same grid.

We train this network on an EED inpainting problem with random masks of 20% density. The EED parameters $\lambda = 0.93, \sigma = 0.97$ have been optimised for inpainting quality with a simple grid search.

To show how the FMG network can benefit from the multigrid structure, we compare it against two networks with the same amount of parameters. One network solves the problem only on a single grid by using 25 layers and 24 channels. Moreover, we compare our FMG network to a standard U-net with addition with three scales, 17 channels and two layers per scale. All three models contain 1.2×10^5 trainable parameters.

Figure 7.10 shows the inpainting results for both networks at the example of a 128×128 version of the image *trui*. Moreover, we present the true inpainting result obtained from a conjugate gradient (CG) solver for EED inpainting.

In contrast to the single grid network, the FMG network and the U-net are able to approximate the EED inpainting result. The FMG and U-net results are visually comparable, while the residual of the U-net is slightly better. This does not only show that the multigrid

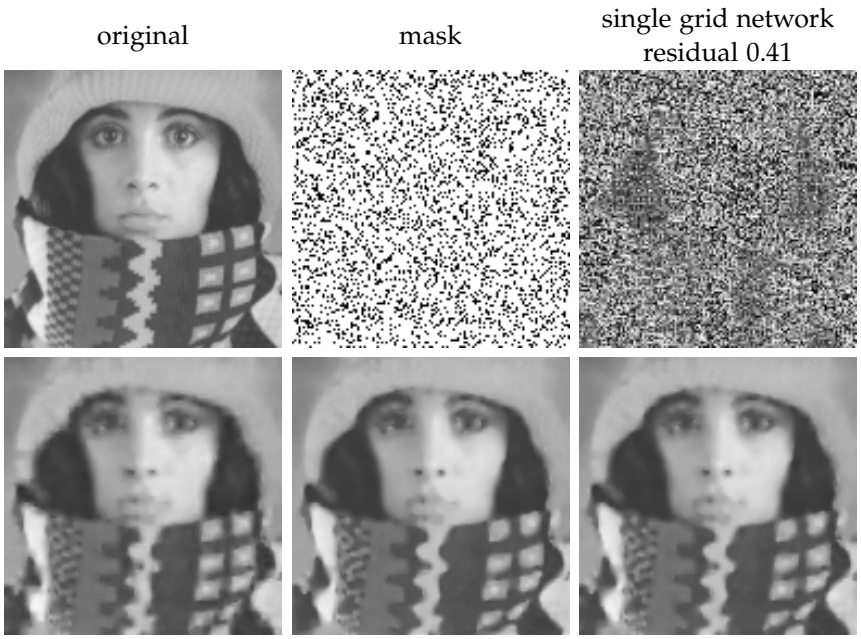


Figure 7.10: Reconstruction quality of a single grid network, a full multigrid network, a standard U-net, and a classical CG solver for the EED inpainting problem. All results use $\lambda = 0.93, \sigma = 0.97$ and the same random mask with 20% density. Both the full multigrid network and the standard U-net approximate an EED inpainting result, while a single grid network fails to do so.

structure is an adequate network design, but also that a standard U-net is not able to obtain a much better solution in this case. From the perspective of numerical algorithms, this is expected, since we know that multigrid methods are highly efficient for these problems.

The advantage of the FMG network is that adding further solvers on the three scales will not inflate the number of parameters as these solvers are shared. This is in contrast to the U-net, where any addition increases the trainable parameter set. The architectural design of the FMG network suggests that U-nets should also be constructed in a similar way. In practice, already concatenating multiple U-nets [268] is a successful idea. Instead of a single down- and upsampling pass, multiple alternating computations on different resolutions should be beneficial.

7.7 CONCLUSIONS

We have shown that numerical algorithms for diffusion evolutions share structural connections to CNN architectures and inspire novel design concepts.

Table 7.1: Overview of connections between numerical and neural concepts which we have encountered.

Numerical Concept	Neural Concept	
numerical algorithm	neural network architecture	**interpretation**
evolution equation	trained neural network	
specification of nonlinear dynamics	training	
explicit scheme	residual network	**coarse connections**
implicit scheme	recurrent network	
multigrid techniques	U-net architectures	
time level	residual block	**detailed connections**
flux function	activation function	
spatial derivative	convolution kernel	
temporal derivative	skip connection	
acceleration strategies	weighted skip connections	

Explicit diffusion schemes yield a specific form of residual networks with a symmetric filter structure, for which one can prove Euclidean stability. Moreover, this architecture saves half of the network parameters, and its stability constraint is easy to implement without affecting its performance. Answering an open question from the previous chapter, we have shown that nonmonotone activations perform well for a denoising task, even when using them within standard architectures in a plug-and-play fashion.

By investigating accelerated explicit schemes and implicit schemes we have justified the effectiveness of skip connections in neural networks. They realise time discretisations in explicit schemes, extrapolation terms to increase their efficiency, and recurrent connections in implicit schemes with fixed point structure. In practice, the resulting architectures are particularly useful when training networks with small amounts of layers.

Lastly, our connection between multigrid concepts and U-net architectures serves as a basis for explaining their success. We have shown that a U-net architecture is able to implement a full multigrid strategy, which allows to learn efficient solutions for PDEs which are typically hard to solve. By directly using the residual norm as a loss function, we can guarantee that the network approximates the PDE at hand. This suggests to extend the standard U-net architecture in a full multigrid fashion.

This concludes our considerations in the one-dimensional setting. Our direct translations of classical methods and their numerical implementations have yielded structural insights into popular neural networks and inspired well-founded neural building blocks with practical relevance. An overview of the detailed connections that we have encountered so far is presented in Table 7.1.

Our numerical perspective on neural networks differs from most viewpoints in the literature. However, it provides a blueprint for directly translating a plethora of numerical strategies into well-founded and practically relevant neural building components. In the following chapter, we extend these considerations to the two-dimensional setting. Therein, we discuss the central question of guaranteeing rotational invariance within neural networks. Our established connections between explicit schemes and ResNets provide the foundation of this endeavour.

8 ROTATIONALLY INVARIANT NEURAL NETWORKS

In this chapter, we transfer our insights from the one-dimensional setting to the two-dimensional domain. In particular, we focus on guaranteed rotation invariance for neural networks.

PDEs and variational methods are again an excellent starting point, as such models often achieve invariance under transformations such as translations and rotations by design. These invariances reflect the physical motivation of the models: Transforming the input should lead to an equally transformed output.

Due to their convolution structure, CNNs are shift invariant by design. However, they lack inherent rotation invariance. Proposed adaptations often inflate the network structure and rely on complex filter design with large stencils; see e.g. [382].

As a core novelty we propose activation functions which couple network channels by combining information from several oriented filters. This guarantees rotation invariance within the basic building blocks of the networks while still allowing for directional filtering. The resulting neural architectures are inherently rotationally invariant. With only a few small filters, they can achieve the same invariance as existing techniques which require a fine-grained sampling of orientations.

Since in the literature, multiple notions of rotation invariance exist, we define our terminology in the following. We call an operation rotationally invariant, if rotating its input yields an equally rotated output. Thus, rotation and operation are interchangeable. This notion follows the classical definition of rotation invariance for differential operators. Note that recent CNN literature refers to this concept as equivariance.

We start with simple two-dimensional diffusion models for greyscale images. Extending the connections from the previous chapters between explicit schemes for these models and ResNets leads to activation functions which couple network channels. Their result is based on a rotationally invariant measure involving specific channels representing differential operators.

By exploring multi-channel and multiscale diffusion models, we generalise the concept of coupling to ResNeXt [398] architectures as an extension of the ResNet. Activations which couple all network channels preserve rotation invariance, but allow to design anisotropic models with a directional filtering.

We derive three central design principles for rotationally invariant neural network design, discuss their effects on practical CNNs and evaluate their effectiveness within an experimental evaluation.

RELATED WORK A simple option to learn a rotationally invariant model is to perform data augmentation [339], where the network is trained on randomly rotated input data. This strategy, however, only approximates rotation invariance and is heavily dependent on the data at hand.

An alternative is to design the filters themselves in a rotationally invariant way, e.g. by weight restriction [74]. However, the resulting rotation invariance is too fine-grained: The filters as the smallest network component are not oriented. Thus, the model is not able to perform a directional filtering.

Other works [123, 221] create a set of rotated input images and apply filters with weight sharing to this set. Depending on the amount of sampled orientations, this can lead to large computational overhead.

An elegant solution for inherent rotation invariance is based on symmetry groups. Gens and Domingos [139] as well as Dieleman et al. [102] propose to consider sets of feature maps which are rotated versions of each other. This comes at a high memory cost as four times as many feature maps need to be processed. Marcos et al. [249] propose to rotate the filters instead of the features, with an additional pooling of orientations. However, the pooling reduces the directional information too quickly. A crucial downside of all these approaches is that they only use four orientations. This only yields a coarse approximation of rotation invariance.

This idea has been generalised to arbitrary symmetry groups by Cohen and Welling [84] through the use of group convolution layers. Group convolutions lift the standard convolution to other symmetry groups which can also include rotations, thus leading to rotation invariance by design. However, also there, only four rotations are considered. This is remedied by Weiler et al. [380, 382] who make use of steerable filters [129] to design a larger set of oriented filters. Duits et al. [112] go one step further by formulating all layers as solvers to parametrised PDEs. Similar ideas have been implemented with wavelets [337] and circular harmonics [395], and the group invariance concept has also been extended to higher dimensional data [83, 289, 381]. However, processing multiple orientations in dedicated network channels inflates the network architecture, and discretising the large set of oriented filters requires the use of large stencils.

We provide an alternative by means of a more sophisticated activation function design. By coupling specific network channels, we can achieve inherent rotation invariance without using large stencils or group theory, while still allowing for models to perform directional filtering. In a similar manner, we have seen in Chapter 4 how wavelet shrinkage can be made rotationally invariant by using a coupling wavelet shrinkage function [262]. However, to the best of our knowledge coupling activation functions have not been considered in CNNs so far.

PUBLICATION INFORMATION This chapter is based on the contents of Alt et al. [12] which has been published within the special issue 'PDE Methods for Machine Learning' of the Research in the Mathematical Sciences journal.

ORGANISATION OF THE CHAPTER We motivate our view on rotationally invariant design with a tutorial example in Section 8.1. In Section 8.2, we connect various diffusion models and their associated energies to their neural counterparts and identify central concepts for rotation invariance. We summarise our findings and discuss their practical implementation in Section 8.3 and conduct experiments on rotation invariance in Section 8.4. We finish this chapter with our conclusions in Section 8.5.

8.1 TWO VIEWS ON ROTATIONAL INVARIANCE

To motivate our viewpoint on rotationally invariant model design, we review a nonlinear diffusion filter of Weickert [367] for image denoising and enhancement. It achieves anisotropy by integrating one-dimensional diffusion processes over all directions. This integration model creates a family of greyscale images $u(\boldsymbol{x}, t) : \Omega \times [0, \infty) \to \mathbb{R}$ on an image domain $\Omega \subset \mathbb{R}^2$ according to the integrodifferential equation

$$\partial_t u = \frac{2}{\pi} \int_0^\pi \partial_{e_\theta} \left(g\left(|\partial_{e_\theta} u_\sigma|^2 \right) \partial_{e_\theta} u \right) \, d\theta \tag{8.1}$$

where ∂_{e_θ} is a directional derivative along the orientation of an angle θ. The evolution is initialised as $u(\cdot, 0) = f$ with the original image f, and reflecting boundary conditions are imposed. The model integrates one-dimensional nonlinear diffusion processes with different orientations θ. All of them share a diffusivity function g which steers the diffusion in dependence of the local directional image structure $|\partial_{e_\theta} u_\sigma|^2$.

As this model diffuses more along low contrast directions than along high contrast ones, it is anisotropic. It is still rotationally invariant, since it combines all orientations of the one-dimensional processes with equal importance. However, this concept comes at the cost of an elaborate discretisation. First, one requires a large amount of discrete rotation angles for a reasonable approximation of the integration. Discretising the directional derivatives in all these directions with a sufficient order of consistency requires the use of large filter stencils; cf. also [48]. The design of rotationally invariant networks such as [382] faces similar difficulties. Processing the input by applying several rotated versions of an oriented filter requires large stencils and many orientations.

A much simpler option arises when considering the closely related EED model [368]

$$\partial_t u = \boldsymbol{\nabla}^\top \left(\boldsymbol{D}(\boldsymbol{\nabla} u_\sigma) \, \boldsymbol{\nabla} u \right). \tag{8.2}$$

As discussed in Section 3.1.1, a diffusion tensor $\boldsymbol{D}(\boldsymbol{\nabla} u_\sigma)$ adapts the diffusion process to local directional information by smoothing along, but not across dominant image structures.

Discretising the EED model (8.2) is much more convenient. For example, a discretisation of the divergence term with good rotation invariance can be performed on a 3×3 stencil, which is the minimal size for a consistent discretisation of a second order model [379]; see Section 3.1.2.

This illustrates a central insight: *One can replace a complex discretisation by a sophisticated design of the nonlinearity*. This motivates us to investigate how rotationally invariant design principles of diffusion models translate into novel activation function designs.

8.2 FROM DIFFUSION PDES AND VARIATIONAL MODELS TO ROTATIONALLY INVARIANT NETWORKS

8.2.1 *Isotropic Diffusion on Greyscale Images*

We first consider the simplest setting of isotropic diffusion models for images with a single channel. By selecting three diffusion models from Section 3.1.1, we identify the common concepts for rotation invariance, and find a unifying neural network interpretation.

We start with the second order isotropic diffusion model of Perona and Malik [279], which is given by the PDE

$$\partial_t u = \boldsymbol{\nabla}^\top \left(g\left(|\boldsymbol{\nabla} u|^2 \right) \boldsymbol{\nabla} u \right) \tag{8.3}$$

with reflecting boundary conditions.

A reformulation of the variational counterpart of this model helps us to identify the cause of its rotation invariance. An energy for the Perona–Malik model can be written in the following way which allows a different generalisation:

$$E(u) = \int_\Omega \Psi\left(\operatorname{tr}\left(\boldsymbol{\nabla} u \boldsymbol{\nabla} u^\top \right) \right) \, d\boldsymbol{x} \tag{8.4}$$

with an increasing regulariser function Ψ and $g = \Psi'$ [326].

Note that we restrict ourselves to energy functionals with only a regularisation term and interpret the gradient descent to the energy as a parabolic diffusion PDE. The variational framework is the simplest setting for analysing invariance properties, as these are automatically transferred to the corresponding diffusion process.

The argument of the regulariser is the trace of the so-called structure tensor [127], here without Gaussian regularisation, which reads

$$\boldsymbol{\nabla} u \boldsymbol{\nabla} u^\top = \begin{pmatrix} u_x^2 & u_x u_y \\ u_x u_y & u_y^2 \end{pmatrix}. \tag{8.5}$$

This structure tensor is a 2×2 matrix with eigenvectors $v_1 \parallel \nabla u$ and $v_2 \perp \nabla u$ parallel and orthogonal to the image gradient. The corresponding eigenvalues are given by $\nu_1 = |\nabla u|^2$ and $\nu_2 = 0$, respectively. Thus, the eigenvectors span a local coordinate system where the axes point across and along dominant structures of the image, and the larger eigenvalue describes the magnitude of image structures.

The use of the structure tensor is the key to rotation invariance. A rotation of the image induces a corresponding rotation of the structure tensor and the structural information that it encodes: Its eigenvectors rotate along, and its eigenvalues remain unchanged. Consequently, the trace as the sum of the eigenvalues is rotationally invariant.

In the following, we explore other ways to design the energy functional based on rotationally invariant quantities and investigate how the resulting diffusion model changes.

The fourth order model of You and Kaveh [401] relies on the Hessian matrix. The corresponding energy functional reads

$$E(u) = \int_\Omega \Psi\Big((\operatorname{tr}(\boldsymbol{H}(u)))^2 \Big) \, d\boldsymbol{x}. \tag{8.6}$$

Here, the regulariser takes the squared trace of the Hessian matrix $\boldsymbol{H}(u)$ as an argument. Since the trace of the Hessian is equivalent to the Laplacian Δu, the gradient flow of (8.6) can be written as

$$\partial_t u = -\Delta \left(g\Big((\Delta u)^2 \Big) \, \Delta u \right). \tag{8.7}$$

This is a fourth order counterpart to the Perona–Malik model. Instead of the gradient operator, one considers the Laplacian Δ. This change was motivated as one remedy to the staircasing effect of the Perona–Malik model [401].

The rotationally invariant matrix at hand is the Hessian $\boldsymbol{H}(u)$. In a similar manner as the structure tensor, the Hessian describes local structure and thus follows a rotation of this structure. Also in this case, the trace operation reduces the Hessian to a scalar that does not change under rotations.

To avoid speckle artefacts of the model of You and Kaveh, Lysaker et al. [242] propose to combine all entries of the Hessian in the regulariser. They choose the Frobenius norm of the Hessian $\|\boldsymbol{H}(u)\|_F^2$ together with a total variation regulariser. For more general regularisers, this model reads [101]

$$E(u) = \int_\Omega \Psi\big(\|\boldsymbol{H}(u)\|_F^2 \big) \, d\boldsymbol{x}. \tag{8.8}$$

Its gradient descent yields a diffusion process of the form

$$\partial_t u = -\mathcal{D}^\top \big(g\big(\|\boldsymbol{H}(u)\|_F^2 \big) \, \mathcal{D} u \big), \tag{8.9}$$

where the differential operator $\mathcal{D}$ induced by the Frobenius norm reads

$$\mathcal{D} = \big(\partial_{xx}, \partial_{xy}, \partial_{yx}, \partial_{yy} \big)^\top. \tag{8.10}$$

This shows another option how one can use the rotationally invariant information of the Hessian matrix. While the choice of a Frobenius norm instead of the trace operator changes the associated differential operators in the diffusion model, it does not destroy the rotation invariance property: The squared Frobenius norm is the sum of the squared eigenvalues of the Hessian, which in turn are rotationally invariant.

8.2.2 *Coupled Activations for Operator Channels*

In the following, we extend the connections between residual networks and explicit schemes from the previous chapters in order to transfer rotation invariance concepts to neural networks. To this end, we consider the generalised diffusion PDE

$$\partial_t u = -\mathcal{D}^*\left(g\left(|\mathcal{D}u|^2\right)\mathcal{D}u\right). \tag{8.11}$$

Here, we use a generalised differential operator $\mathcal{D}$ and its adjoint $\mathcal{D}^*$. This PDE is the two-dimensional counterpart of Equation (7.1) and subsumes the diffusion models (8.3), (8.7), and (8.9). Since the diffusivities take a scalar argument, we can express the diffusivity as $g(|\mathcal{D}u|^2)$. The differential operator $\mathcal{D}$ is induced by the associated energy functional.

As before, to connect the generalised model (8.11) to a ResNet architecture, we rewrite (8.11) by means of the now vector-valued flux function

$$\boldsymbol{\Phi}(\boldsymbol{s}) = g(|\boldsymbol{s}|^2)\,\boldsymbol{s} \tag{8.12}$$

as

$$\partial_t u = -\mathcal{D}^*\left(\boldsymbol{\Phi}(\mathcal{D}u)\right). \tag{8.13}$$

Let us now consider an explicit discretisation for this diffusion PDE. Following the same reasoning as in the previous chapter, we obtain an explicit scheme for (8.13) as

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \tau \boldsymbol{K}^\top \boldsymbol{\Phi}\left(\boldsymbol{K}\boldsymbol{u}^k\right). \tag{8.14}$$

In contrast to the one-dimensional case, the matrix $\boldsymbol{K}$ implements a set of two-dimensional convolutions. However, due to the flexibility of the residual block, this does not change its connections to the explicit scheme. As before, one can connect the two by identifying convolutions $\boldsymbol{W}_1, \boldsymbol{W}_2$ and activation functions φ_1, φ_2 with

$$\varphi_1 = \tau\boldsymbol{\Phi}, \quad \varphi_2 = \mathrm{Id}, \quad \boldsymbol{W}_1 = \boldsymbol{K}, \quad \boldsymbol{W}_2 = -\boldsymbol{K}^\top, \tag{8.15}$$

and setting the bias vectors to $\mathbf{0}$ [10, 319, 412].

In contrast to the one-dimensional considerations in Chapter 7, the connection between flux and activation in the two-dimensional setting yields additional, novel design concepts for activation functions. This yields the first design principle for neural networks.

Design Principle 1 (Coupled Activations for Rotational Invariance)**.** *Activation functions which couple network channels can be used to design rotationally invariant networks. At each position of the image, the channels of the inner convolution result are combined within a rotationally invariant quantity which determines the nonlinear response.*

The coupling effect of the diffusivity and the regulariser directly transfers to the activation function. This is apparent when the differential operator $\mathcal{D}$ contains multiple components. For example, consider an operator $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)^\top$ with two components and its discrete variant $\boldsymbol{K} = (\boldsymbol{K}_1, \boldsymbol{K}_2)^\top$.

The application of the operator $\boldsymbol{K}$ transforms the single-channel signal $\boldsymbol{u}^k$ into a signal with two channels. Then the activation function couples the information from both channels within the diffusivity g. For each pixel position i, j, we have

$$\boldsymbol{\Phi}\begin{pmatrix} (\boldsymbol{K}_1\boldsymbol{u}^k)_{i,j} \\ (\boldsymbol{K}_2\boldsymbol{u}^k)_{i,j} \end{pmatrix} = g\left(\left|\boldsymbol{K}_1\boldsymbol{u}^k\right|^2_{i,j} + \left|\boldsymbol{K}_2\boldsymbol{u}^k\right|^2_{i,j}\right)\begin{pmatrix} (\boldsymbol{K}_1\boldsymbol{u}^k)_{i,j} \\ (\boldsymbol{K}_2\boldsymbol{u}^k)_{i,j} \end{pmatrix}. \tag{8.16}$$

Afterwards, the application of $\boldsymbol{K}^\top$ reduces the resulting two-channel signal to a single channel again.

In the general case, the underlying differential operator $\mathcal{D}$ determines how many channels are coupled. The choice

$$\mathcal{D} = (\partial_{xx}, \partial_{xy}, \partial_{yx}, \partial_{yy})^\top \tag{8.17}$$

of Lysaker et al. [242] induces a coupling of four channels containing second order derivatives. This shows that a central condition for rotation invariance is that the convolution $\boldsymbol{K}$ implements a rotationally invariant differential operator. We discuss the effects of this condition on the practical filter design in Section 8.3.

We call a block of the form (8.14) a *generalised two-dimensional diffusion block*. It is visualised in Figure 8.1 in graph form. We denote the channel coupling by a shaded connection to the activation function.

The coupling effect is natural in the diffusion case. However, in standard networks, the input is quickly deconstructed into multiple channels, each one concerned with different, specific image features. Each channel is activated independently, and information is exchanged through trainable convolutions. While this makes networks flexible, it does not take into account concepts such as rotation invariance. Thus, we see that allowing coupled activations can serve as an alternative way to guarantee built-in rotation invariance within a network. To the best of our knowledge, this concept has not been proposed for CNNs in the context of rotation invariance.

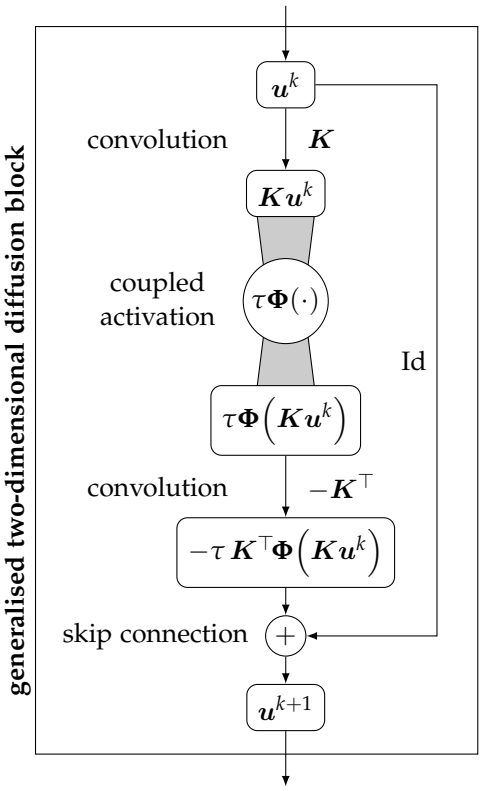


Figure 8.1: Generalised two-dimensional diffusion block for an explicit diffusion step (8.14) with activation function $\tau\Phi$, time step size τ, and convolution filters K. The activation function couples the channels of the operator K.

8.2.3 *Diffusion on Multi-channel Images*

So far, the considered models were isotropic. However, we will see that anisotropic models inspire another form of activation function which combines directional filtering with rotation invariance.

To this end, we move to diffusion on multi-channel images. While there are anisotropic models for single-channel images [369], they require a presmoothing as in the EED model. However, such models do not have a conventional energy formulation [384]. The multi-channel setting allows one to design anisotropic models that do not require a presmoothing and arise from a variational energy.

In the following we consider images $u = (u_1, u_2, \dots, u_M)^\top$ with M channels. To distinguish them from the previously considered channels of the differential operator, we refer to image channels and operator channels in the following.

A naive extension of the Perona–Malik model (8.3) to multi-channel images would treat each image channel separately. Consequently, the energy would consider a regularisation of the trace of the structure

tensor for each individual channel. This in turn does not respect the fact that structural information is correlated in the channels.

To incorporate this correlation, Gerig et al. [141] proposed to sum up structural information from all channels. An energy functional for this model reads

$$E(\boldsymbol{u}) = \int_{\Omega} \Psi\left(\operatorname{tr} \sum_{m=1}^{M} \boldsymbol{\nabla} u_m \boldsymbol{\nabla} u_m^{\top}\right) d\boldsymbol{x}. \tag{8.18}$$

Here we also use the trace formulation. It shows that this model makes use of a colour structure tensor, which goes back to Di Zenzo [98]. It is the sum of the structure tensors of the individual channels. In contrast to the single-channel structure tensor without Gaussian regularisation, no closed form solutions for its eigenvalues and eigenvectors are available. Still, the sum of structure tensors stays rotationally invariant.

The corresponding diffusion process is described by the coupled PDE set

$$\partial_t u_m = \boldsymbol{\nabla}^{\top}\left(g\left(\sum_{n=1}^{M} |\boldsymbol{\nabla} u_n|^2\right) \boldsymbol{\nabla} u_m\right) \qquad (m = 1, \dots, M) \tag{8.19}$$

with reflecting boundary conditions. As trace and summation are interchangeable, the argument of the regulariser corresponds to a sum of channel-wise gradient magnitudes. Thus, the diffusivity considers information from all channels. It allows to steer the diffusion process in all channels depending on a joint structure measure.

Interestingly, a simple change in the energy model (8.18) incorporates directional information [373] such that the model becomes anisotropic. Switching the trace operator and the regulariser yields the energy

$$E(\boldsymbol{u}) = \int_{\Omega} \operatorname{tr} \Psi\left(\sum_{m=1}^{M} \boldsymbol{\nabla} u_m \boldsymbol{\nabla} u_m^{\top}\right) d\boldsymbol{x}. \tag{8.20}$$

Now the regulariser acts on the colour structure tensor in the sense of a power series. Thus, the regulariser modifies the eigenvalues ν_1, ν_2 to $\Psi(\nu_1), \Psi(\nu_2)$ and leaves the eigenvectors unchanged. For the 2×2 colour structure tensor we have

$$\Psi\left(\sum_{m=1}^{M} \boldsymbol{\nabla} u_m \boldsymbol{\nabla} u_m^{\top}\right) = \Psi(\nu_1)\, \boldsymbol{v}_1 \boldsymbol{v}_1^{\top} + \Psi(\nu_2)\, \boldsymbol{v}_2 \boldsymbol{v}_2^{\top}. \tag{8.21}$$

The eigenvalues are treated individually. This allows for an anisotropic model, as each eigenvalue determines the local image contrast along its corresponding eigenvector. Still, the model is rotationally invariant as the colour structure tensor rotates accordingly. Consequently, the trace of this regulariser is equivalent to the sum of the regularised eigenvalues:

$$\operatorname{tr} \Psi\left(\sum_{m=1}^{M} \boldsymbol{\nabla} u_m \boldsymbol{\nabla} u_m^{\top}\right) = \Psi(\nu_1) + \Psi(\nu_2). \tag{8.22}$$

This illustrates the crucial difference to the isotropic case, where we have

$$\Psi\left(\operatorname{tr}\sum_{m=1}^{M}\nabla u_m \nabla u_m^\top\right) = \Psi(\nu_1+\nu_2)\,, \tag{8.23}$$

Both eigenvalues of the structure tensor are regularised jointly and the result is a scalar, which shows that no directional information can be involved. At this point, the motivation for using the structure tensor notation in the previous models becomes apparent: Switching the trace operator and the regulariser changes an isotropic model into an anisotropic one. This is the same connection that we used to obtain the IID model from the IAD model in Chapter 5.

The gradient descent of the energy (8.20) is an anisotropic nonlinear diffusion model for multi-channel images [373]:

$$\partial_t u_m = \nabla^\top\left(g\left(\sum_{n=1}^{M}\nabla u_n \nabla u_n^\top\right)\nabla u_m\right) \qquad (m=1,\dots,M)\,. \tag{8.24}$$

The diffusivity inherits the matrix-valued argument of the regulariser. Thus, it is applied in the same way and yields a 2×2 diffusion tensor. In contrast to single-channel diffusion, this creates anisotropy as its eigenvectors do not necessarily coincide with ∇u. Thus, the multi-channel case does not require Gaussian presmoothing.

We have seen that the coupling effect within the diffusivity goes beyond the channels of the differential operator. It combines both the operator channels as well as the image channels within a joint measure. Whether the model is isotropic or anisotropic is determined by the shape of the diffusivity result: Isotropic models use scalar diffusivities, while anisotropic models require matrix-valued diffusion tensors. In the following, we generalise this concept and analyse its influence on the ResNet architecture.

8.2.4 *Coupled Activations for Image Channels*

A generalised formulation of the multichannel diffusion models (8.19) and (8.24) is given by

$$\partial_t u_m = -\mathcal{D}^*\boldsymbol{\Phi}(\boldsymbol{u}, \mathcal{D}u_m) \qquad (m=1,\dots,M)\,. \tag{8.25}$$

As the flux function uses more information than only $\mathcal{D}u_m$, we switch to the notation $\boldsymbol{\Phi}(\boldsymbol{u}, \mathcal{D}u_m)$. An explicit scheme for this model is derived in a similar way as before, yielding

$$\boldsymbol{u}_m^{k+1} = \boldsymbol{u}_m^k - \tau \boldsymbol{K}^\top \boldsymbol{\Phi}\left(\boldsymbol{u}^k, \boldsymbol{K}\boldsymbol{u}_m^k\right) \qquad (m=1,\dots,M)\,. \tag{8.26}$$

The activation function now couples more than just the operator channels, it couples all its input channels. In contrast to Design Principle 1, this coupling is more general and provides a second design principle.

Design Principle 2 (Fully Coupled Activations for Image Channels)**.** *Activations which couple both operator channels and image channels can be used to create anisotropic, rotationally invariant models. At each position of the image, all operator channels for all image channels are combined within a rotationally invariant quantity which determines the nonlinear response.*

The different coupling effects serve different purposes: Coupling the image channels accounts for structural correlations and can be used to create anisotropy. Coupling the channels of the differential operators guarantees rotation invariance.

This design principle becomes apparent when explicitly formulating the activation function. Isotropic models use a scalar diffusivity within the flux function

$$\boldsymbol{\Phi}\left(\boldsymbol{u}^k, \left(\boldsymbol{K}\boldsymbol{u}_m^k\right)_{i,j}\right) = g\left(\sum_{m=1}^{M} \left|\boldsymbol{K}\boldsymbol{u}_m^k\right|_{i,j}^2\right)\left(\boldsymbol{K}\boldsymbol{u}_m^k\right)_{i,j} \qquad (m = 1,\dots,M)\,, \tag{8.27}$$

which couples all channels of $\boldsymbol{u}$ at the position i,j, as well as all components of the discrete operator $\boldsymbol{K}$. Anisotropic models require a matrix-valued diffusion tensor in the flux function

$$\boldsymbol{\Phi}\left(\boldsymbol{u}^k, \left(\boldsymbol{K}\boldsymbol{u}_m^k\right)_{i,j}\right) = g\left(\sum_{m=1}^{M} \left(\boldsymbol{K}\boldsymbol{u}_m^k\right)_{i,j}\left(\boldsymbol{K}\boldsymbol{u}_m^k\right)_{i,j}^{\top}\right)\left(\boldsymbol{K}\boldsymbol{u}_m^k\right)_{i,j} \qquad (m = 1,\dots,M)\,, \tag{8.28}$$

This concept is visualised in Figure 8.2 in the form of a *fully coupled multi-channel diffusion block*. To clarify the distinction between image and operator channels, we explicitly split the image into its channels. We see that all information of the inner filter passes through a single activation function and influences all outgoing results in the same manner.

Design Principle 2 shows that coupling cannot only be used for rotationally invariant design, but also makes sense for implementing modelling aspects such as anisotropy. This is desirable as anisotropic models often exhibit higher performance through better adaptivity to data.

8.2.5 *Integrodifferential Diffusion*

The previous models work on the finest scale of the image. However, as shown in Part I, generating a structural measure which incorporates information from multiple image scales can be beneficial.

To this end, we consider variants of the integrodifferential models from Chapter 5. In analogy to the multi-channel diffusion setting, these models inspire a full coupling of scale information for a variation of residual networks.

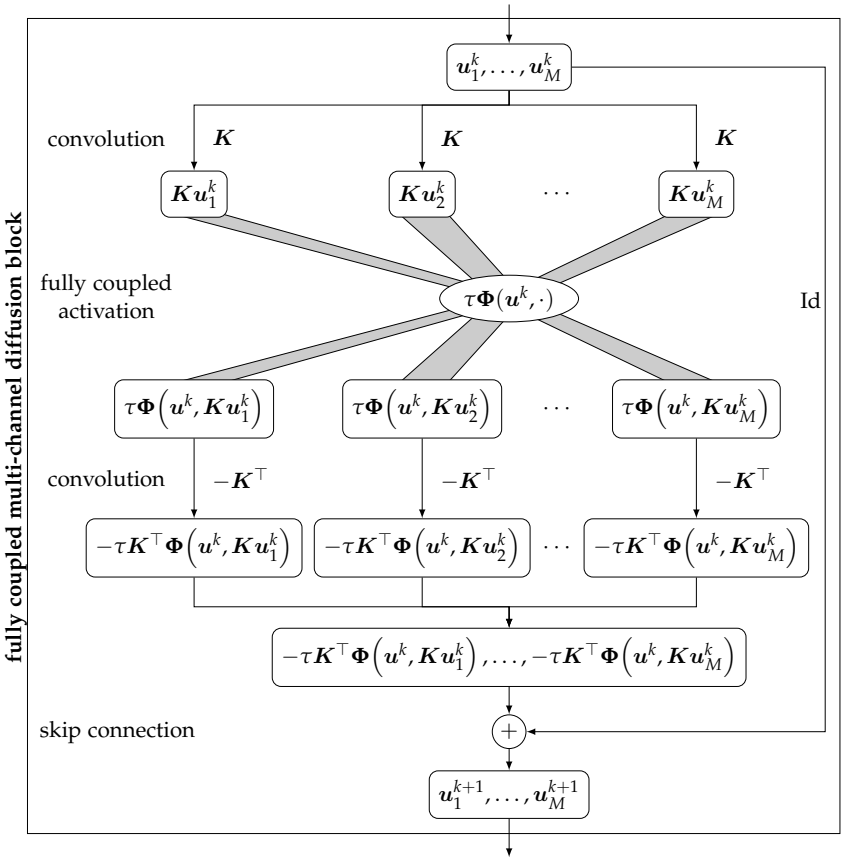


Figure 8.2: Fully coupled multi-channel diffusion block for an explicit step (8.26) with a fully coupled activation function $\tau\Phi$ and convolution filters K. The activation function couples all operator and image channels of its input jointly. Depending on the design of the activation, the resulting model can be isotropic or anisotropic.

We start with the energy functional

$$E(u) = \int_\Omega \Psi\left(\operatorname{tr} \int_0^\infty \left(\mathcal{D}^{(\sigma)} u\right) \left(\mathcal{D}^{(\sigma)} u\right)^\top d\sigma\right) dx. \tag{8.29}$$

We denote the scale parameter by σ and assume that the differential operators $\mathcal{D}^{(\sigma)}$ are dependent on the scale. This can be realised for example by an adaptive presmoothing of an underlying differential operator; see e.g. [14, 100].

Instead of summing structure tensors over image channels, this model integrates generalised structure tensors $\left(\mathcal{D}^{(\sigma)} u\right) \left(\mathcal{D}^{(\sigma)} u\right)^\top$ over multiple scales. This results in a multiscale structure tensor [14] which contains a semi-local measure for image structure. If $\mathcal{D}^{(\sigma)}$ are rotationally invariant operators, then the multiscale structure tensor is also invariant.

The corresponding diffusion model reads

$$\partial_t u = -\int_0^\infty \mathcal{D}^{(\sigma)*}\left(g\left(\int_0^\infty \left|\mathcal{D}^{(\gamma)}u\right|^2 d\gamma\right)\mathcal{D}^{(\sigma)}u\right) d\sigma, \tag{8.30}$$

where $g = \Psi'$ and one employs reflecting boundary conditions. Due to the chain rule, one obtains two integrations over the scales: The outer integration combines diffusion processes on each scale. The inner integration, where the scale variable has been renamed to γ, accumulates multiscale information within the diffusivity argument.

This model is a modification of the IID model from Chapter 5. Therein, however, the diffusivity uses a scale-adaptive contrast parameter. Thus, it does not arise from an energy functional.

As for the multi-channel diffusion models, switching trace and regulariser yields an anisotropic model, which is described by the energy

$$E(u) = \int_\Omega \operatorname{tr}\Psi\left(\int_0^\infty \left(\mathcal{D}^{(\sigma)}u\right)\left(\mathcal{D}^{(\sigma)}u\right)^\top d\sigma\right) d\boldsymbol{x}. \tag{8.31}$$

In analogy to the multi-channel model, the regulariser is applied directly to the structure tensor, which creates anisotropy. Consequently, the resulting diffusion process is a modification of the IAD model from Chapter 5:

$$\partial_t u = -\int_0^\infty \mathcal{D}^{(\sigma)*}\left(g\left(\int_0^\infty \left(\mathcal{D}^{(\gamma)}u\right)\left(\mathcal{D}^{(\gamma)}u\right)^\top d\gamma\right)\mathcal{D}^{(\sigma)}u\right) d\sigma. \tag{8.32}$$

The anisotropic regularisation is inherited by the diffusivity and results in a flux function that implements a matrix-vector multiplication.

8.2.6 *Coupled Activations for Image Scales*

Both the isotropic and the anisotropic multiscale models can be summarised by the flux formulation

$$\partial_t u = -\int_0^\infty \mathcal{D}^{(\sigma)*}\left(\boldsymbol{\Phi}\left(u, \mathcal{D}^{(\sigma)}u\right)\right) d\sigma. \tag{8.33}$$

To discretise this model, we now require a discretisation of the scale integral. To this end, we select a set of L discrete scales $\sigma_1, \sigma_2, \dots, \sigma_L$. On each scale σ_ℓ, we employ discrete differential operators $\boldsymbol{K}_\ell$. This yields an explicit scheme for the continuous model (8.33) which reads

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \tau \sum_{\ell=1}^{L} \omega_\ell \boldsymbol{K}_\ell^\top \boldsymbol{\Phi}\left(\boldsymbol{u}^k, \boldsymbol{K}_\ell \boldsymbol{u}^k\right). \tag{8.34}$$

Here, ω_ℓ is a step size over the scales, discretising the infinitesimal quantity $d\sigma$. It is dependent on the scale to allow a nonuniform sampling of scales σ_ℓ. A simple choice is $\omega_\ell = \sigma_{\ell+1} - \sigma_\ell$; see Chapter 5.

Interestingly, an extension of residual networks called ResNeXt [398] provides the corresponding neural architecture to this model. Therein, the authors consider a sum of transformations of the input signal together with a skip connection. We restrict ourselves to the following formulation:

$$\boldsymbol{u} = \varphi_2 \left(\boldsymbol{f} + \sum_{\ell=1}^{L} \boldsymbol{W}_{2,\ell} \, \varphi_\ell(\boldsymbol{W}_{1,\ell} \boldsymbol{f} + \boldsymbol{b}_{1,\ell}) + \boldsymbol{b}_{2,\ell} \right). \tag{8.35}$$

This ResNeXt block modifies the input image $\boldsymbol{f}$ within L independent paths, and sums up the results before the skip connection. Each path may apply multiple, differently shaped convolutions. Choosing a single path with $L = 1$ yields the ResNet model.

We can identify an explicit multiscale diffusion step (8.34) with a ResNeXt block (8.35) by

$$\varphi_{1,\ell} = \tau \, \boldsymbol{\Phi}, \quad \varphi_2 = \mathrm{Id}, \quad \boldsymbol{W}_{1,\ell} = \boldsymbol{K}_\ell, \quad \boldsymbol{W}_{2,\ell} = -\omega_\ell \boldsymbol{K}_\ell^\top, \tag{8.36}$$

and all bias vectors $\boldsymbol{b}_{1,\ell}, \boldsymbol{b}_{2,\ell}$ are set to $\boldsymbol{0}$, for all $\ell = 1, \dots, L$.

In contrast to the previous ResNet relation (8.15), we apply different filters $\boldsymbol{K}_\ell$ in each path. Their individual results are summed up before the skip connection, which approximates the scale integration. While the ResNeXt block allows for individual activation functions in each path, we use a common activation with a full coupling for all of them. This constitutes a variant of Design Principle 2, where one now couples image scales.

Design Principle 3 (Fully Coupled Activations for Image Scales)**.** *Activations which couple both operator channels and image scales can be used to create anisotropic, rotationally invariant multiscale models. At each position of the image, all operator channels for all image scales are combined within a rotationally invariant quantity which determines the nonlinear response.*

Also in this case, the combined coupling serves different purposes. Coupling the operator channels yields rotation invariance, and coupling of scales allows to obtain a more global representation of the image structure. Isotropic models employ a coupling with a scalar diffusivity in the flux function

$$\boldsymbol{\Phi}\left(\boldsymbol{u}^k, \left(\boldsymbol{K}_\ell \boldsymbol{u}^k \right)_{i,j} \right) = g\left(\sum_{\ell=1}^{L} \left| \boldsymbol{K}_\ell \boldsymbol{u}^k \right|_{i,j}^2 \right) \left(\boldsymbol{K}_\ell \boldsymbol{u}^k \right)_{i,j}, \tag{8.37}$$

and a matrix-valued diffusion tensor in the flux function

$$\boldsymbol{\Phi}\left(\boldsymbol{u}^k, \left(\boldsymbol{K}_\ell \boldsymbol{u}^k \right)_{i,j} \right) = g\left(\sum_{\ell=1}^{L} \left(\boldsymbol{K}_\ell \boldsymbol{u}^k \right)_{i,j} \left(\boldsymbol{K}_\ell \boldsymbol{u}^k \right)_{i,j}^\top \right) \left(\boldsymbol{K}_\ell \boldsymbol{u}^k \right)_{i,j} \tag{8.38}$$

can be used to create anisotropic models.

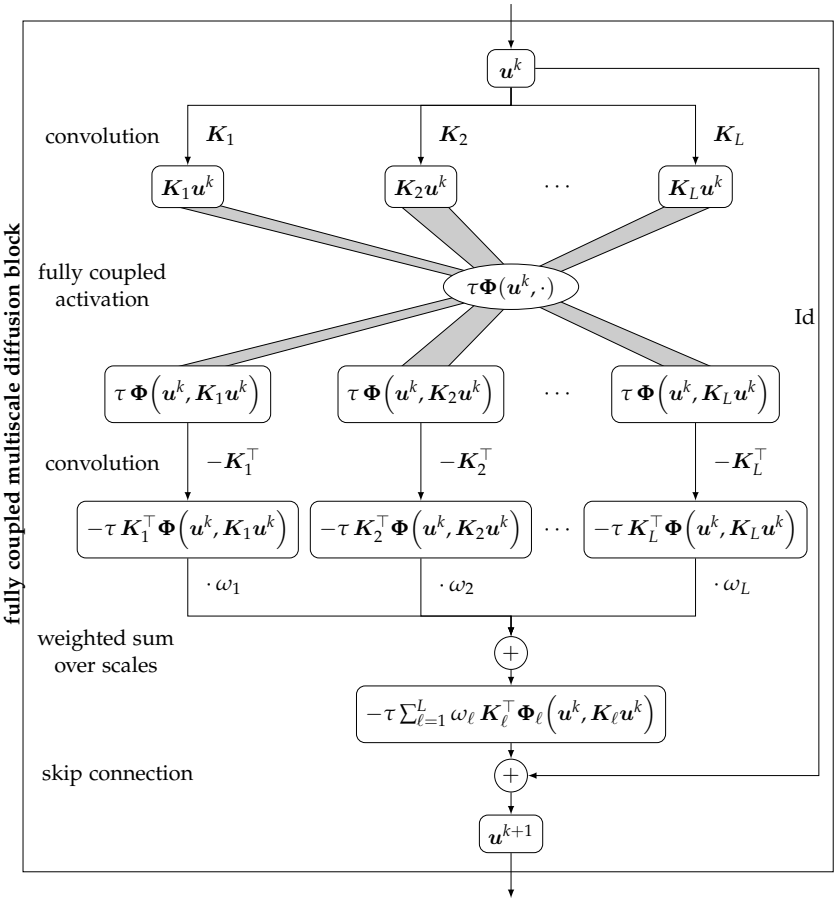


Figure 8.3: Fully coupled multiscale diffusion block for an explicit multiscale diffusion step (8.34) with a single activation function $\tau\Phi$, time step size τ, and convolution filters K_ℓ on each scale. The activation function couples all inputs jointly. Depending on the design of the activation, the resulting model can be isotropic or anisotropic.

We call a block of the form (8.34) a *fully coupled multiscale diffusion block*. This block is visualised in Figure 8.3. Comparing its form to that of the multichannel diffusion block in Figure 8.2, one can see that the different architectures use the same activation function design, however with different motivations.

8.3 DISCUSSION

We have seen that shifting the design focus from convolutions to activation functions can yield new insights into CNN design. We summarise all models that we have considered in Table 8.1 as a convenient overview.

All variational models are rotationally invariant, as they rely on a structural measure which accounts for rotations. This directly transfers

to the diffusion model, its explicit scheme, and thus also its network counterpart, resulting in Design Principle 1. Moreover, the different coupling options for models with multiple scales and multiple channels show how a sophisticated activation design can steer the model capacity. This has led to the additional Design Principles 2 and 3.

The coupling effects are naturally motivated for diffusion, but are hitherto unexplored in the CNN world. While activation functions such as maxout [151] and softmax introduce a coupling of their input arguments, they only serve the purpose of reducing channel information. Even though some works focus on using trainable and more advanced activations [75, 124, 272], the coupling aspect has not been considered so far.

The rotation invariance of the proposed architectures can be approximated efficiently in the discrete setting. For example for second order models, Weickert et al. [379] present L^2 stable discretisations with good practical rotation invariance that only require a 3×3 stencil. For models of second order, this is the smallest possible discretisation stencil which still yields consistent results.

In a practical setting with trainable filters, one is not restricted to the differential operators that we have encountered so far. To guarantee that the learned filter corresponds to a rotationally invariant differential operator, one has several options. For example, one can design the filters based on a dictionary of operators which fulfil the rotation invariance property, which are then combined into more complex operators through trainable weights; see e.g. [201]. In a similar manner, one can employ different versions of a base operator which arise from a rotationally invariant operation, e.g. a Gaussian smoothing. We pursue this strategy in our experiments in the following section in analogy to Chapter 5.

Apart from the coupling aspect, the underlying network architecture is not modified. This is a stark contrast to the CNN literature where a set of orientations is discretised, requiring much larger stencils for a good approximation of rotation invariance. We neither require involved discretisations, nor a complicated lifting to groups. Moreover, the proposed architectures allow for a Euclidean stability theory, as we prove in Appendix C. Thus, we regard the proposed activation function design as a promising alternative to the directional splitting idea.

8.4 EXPERIMENTS

In the following, we present an experimental evaluation to support our theoretical considerations. To this end, we design trainable multiscale diffusion models for denoising. We compare models with and without coupling activations, and evaluate their performance on differently

Table 8.1: The considered diffusion models, along with their variational energies and the resulting network architectures.

model	variational energy	diffusion PDE	explicit scheme / network block	activation coupling
Perona–Malik [279], single-channel, isotropic	$E(u) = \int_\Omega \Psi\Big(\operatorname{tr}\Big(\boldsymbol{\nabla} u \boldsymbol{\nabla} u^\top\Big)\Big)\, d\boldsymbol{x}$	$\partial_t u = \boldsymbol{\nabla}^\top \Big(g\Big(\lvert\boldsymbol{\nabla} u\rvert^2\Big)\, \boldsymbol{\nabla} u\Big)$	$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \tau \boldsymbol{K}^\top \boldsymbol{\Phi}\Big(\boldsymbol{K}\boldsymbol{u}^k\Big)$	structure tensor, scalar multiplication
You and Kaveh [401], single-channel, isotropic	$E(u) = \int_\Omega \Psi\Big(\big(\operatorname{tr}(\boldsymbol{H}(u))\big)^2\Big)\, d\boldsymbol{x}$	$\partial_t u = -\Delta\, \Big(g\Big((\Delta u)^2\Big)\, \Delta u\Big)$		Hessian, scalar multiplication
Lysaker et al. [242], single-channel, isotropic	$E(u) = \int_\Omega \Psi\Big(\lVert\boldsymbol{H}(u)\rVert_F^2\Big)\, d\boldsymbol{x}$	$\partial_t u = -\mathcal{D}^*\, \Big(g\Big(\lVert\boldsymbol{H}(u)\rVert_F^2\Big)\, \mathcal{D} u\Big)$ with $\mathcal{D} = \big(\partial_{xx}, \partial_{xy}, \partial_{yx}, \partial_{yy}\big)^\top$		Hessian, scalar multiplication
Gerig et al. [141], multi–channel, isotropic	$E(\boldsymbol{u}) = \int_\Omega \Psi\left(\operatorname{tr} \sum_{m=1}^{M} \boldsymbol{\nabla} u_m \boldsymbol{\nabla} u_m^\top\right) d\boldsymbol{x}$	$\partial_t u_m = \boldsymbol{\nabla}^\top \left(g\left(\sum_{n=1}^{M} \lvert\boldsymbol{\nabla} u_n\rvert^2\right) \boldsymbol{\nabla} u_m\right)$	$\boldsymbol{u}_m^{k+1} = \boldsymbol{u}_m^k - \tau \boldsymbol{K}^\top \boldsymbol{\Phi}\Big(\boldsymbol{u}^k, \boldsymbol{K}\boldsymbol{u}_m^k\Big)$	multi-channel structure tensor, scalar multiplication
Weickert and Brox [373], multi–channel, anisotropic	$E(\boldsymbol{u}) = \int_\Omega \operatorname{tr} \Psi\left(\sum_{m=1}^{M} \boldsymbol{\nabla} u_m \boldsymbol{\nabla} u_m^\top\right) d\boldsymbol{x}$	$\partial_t u_m = \boldsymbol{\nabla}^\top \left(g\left(\sum_{n=1}^{M} \boldsymbol{\nabla} u_n \boldsymbol{\nabla} u_n^\top\right) \boldsymbol{\nabla} u_m\right)$		multi-channel structure tensor, matrix multiplication
Alt and Weickert [14], multiscale, isotropic	$E(u) = \int_\Omega \Psi\left(\operatorname{tr} \int_0^\infty \Big(\mathcal{D}^{(\sigma)} u\Big) \Big(\mathcal{D}^{(\sigma)} u\Big)^\top d\sigma\right) d\boldsymbol{x}$	$\partial_t u = -\int_0^\infty \mathcal{D}^{(\sigma)*} \left(g\left(\int_0^\infty \Big\lvert\mathcal{D}^{(\gamma)} u\Big\rvert^2 d\gamma\right) \mathcal{D}^{(\sigma)} u\right) d\sigma$	$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \tau \sum_{\ell=1}^{L} \omega_\ell\, \boldsymbol{K}_\ell^\top \boldsymbol{\Phi}\Big(\boldsymbol{u}^k, \boldsymbol{K}_\ell \boldsymbol{u}^k\Big)$	multiscale structure tensor, scalar multiplication
Alt and Weickert [14], multiscale, anisotropic	$E(u) = \int_\Omega \operatorname{tr} \Psi\left(\int_0^\infty \Big(\mathcal{D}^{(\sigma)} u\Big) \Big(\mathcal{D}^{(\sigma)} u\Big)^\top d\sigma\right) d\boldsymbol{x}$	$\partial_t u = -\int_0^\infty \mathcal{D}^{(\sigma)*} \left(g\left(\int_0^\infty \Big(\mathcal{D}^{(\gamma)} u\Big) \Big(\mathcal{D}^{(\gamma)} u\Big)^\top d\gamma\right) \mathcal{D}^{(\sigma)} u\right) d\sigma$		multiscale structure tensor, matrix multiplication

rotated datasets. This shows that the Design Principle 1 is indeed necessary for rotation invariance.

8.4.1 *Experimental Setup*

We train the isotropic (8.30) and anisotropic (8.32) multiscale diffusion models. Both perform a full coupling of all scales, i.e. they implement Design Principles 1 and 3. As a counterpart, we train the same multiscale diffusion model with the diffusivity applied to each channel of the discrete derivative operator separately. Thus, the activation is applied independently in each direction. This violates Design Principle 1. Hence, the model should yield worse rotation invariance than the coupled models. Still, all models implement Design Principle 3 by integrating multiscale information.

The corresponding explicit scheme for the considered models is given by

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \tau \sum_{\ell=1}^{L} \omega_\ell \, \boldsymbol{K}_\ell^\top \boldsymbol{\Phi}\left(\boldsymbol{u}^k, \boldsymbol{K}_\ell \boldsymbol{u}^k\right) \tag{8.39}$$

The choice for ω_ℓ is set to $\sigma_{\ell+1} - \sigma_\ell$.

As differential operators $\boldsymbol{K}_\ell$ we choose weighted and Gaussian smoothed gradients $\beta_\ell \boldsymbol{\nabla}_{\sigma_\ell}$ on each scale σ_ℓ. The application of a smoothed gradient to an image via $\boldsymbol{\nabla}_\sigma u = G_\sigma * \boldsymbol{\nabla} u$ is equivalent to computing a Gaussian convolution with standard deviation σ of the image gradient. Moreover, we weight the differential operators on each scale by a scale-adaptive, trainable parameter β_ℓ.

A discrete set of $L = 8$ scales is determined by an exponential sampling between a minimum scale of $\sigma_{\text{min}} = 0.1$ and a maximum one of $\sigma_{\text{max}} = 10$ in analogy to Chapter 5. This yields discrete scales $[0.1, 0.18, 0.32, 0.56, 1.0, 1.77, 3.16, 5.62]$. As a diffusivity, we choose the exponential Perona–Malik [279] diffusivity with a trainable contrast parameter λ. Moreover, we train the time step size τ and we use 10 explicit steps with shared parameter sets. This amounts to a total number of 10 trainable parameters: τ, λ, and β_1 to β_8.

In the practical setting, a discretisation with good rotation invariance is crucial. We use the nonstandard finite difference discretisation of Weickert et al. [379]; see Section 3.1.2. For isotropic models, it has a free parameter $\alpha \in \left[0, \frac{1}{2}\right]$ which can be tuned for rotation invariance, with an additional parameter $\gamma \in [0, 1]$ for anisotropic ones. We found that in the denoising case, both parameter choices constitute a trade-off between performance and rotation invariance.

We train all models on a synthetic dataset which consists of greyscale images of size 256×256 with values in the range $[0, 255]$. Each image contains 20 randomly placed white rectangles of size 140×70 on a black background. The rectangles are all oriented along a common direction, which creates a directional bias within the dataset. The training set contains 100 images and is oriented with an angle of 30°

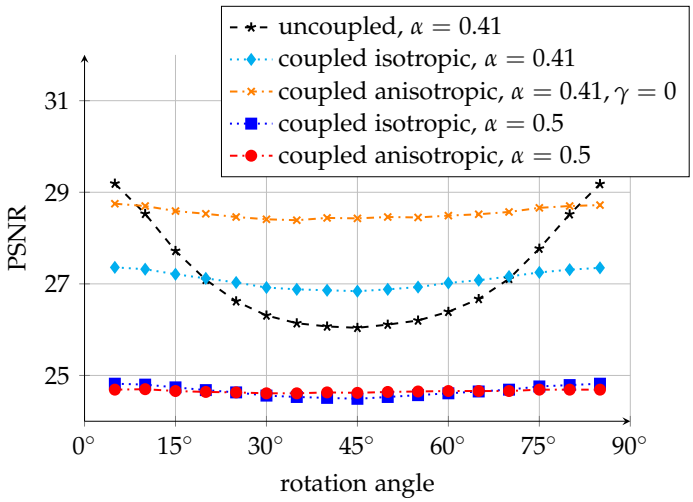


Figure 8.4: Analysis of rotation invariance in terms of denoising quality on differently rotated versions of the test dataset. The models have been trained on a dataset with 30° orientation. The models with coupling approximate rotation invariance significantly better than the uncoupled model.

from the x-axis. As test datasets, we consider rotated versions of a similar set of 50 images. The rotation angles are sampled between 0° and 90° in steps of 5°. To avoid an influence of the image sampling on the evaluation, we exclude the axis-aligned datasets.

To train the models for the denoising task, we add noise of standard deviation 60 to the clean training images and minimise the Euclidean distance to the ground truth images. We measure the denoising quality in terms of peak-signal-to-noise ratio (PSNR). All models are trained for 400 epochs with the Adam optimiser [212] with standard settings and a learning rate of 0.001. One training epoch requires 50 seconds on an *NVIDIA GeForce GTX 1060 6GB*, and the evaluation on one of the test sets requires 7 seconds.

8.4.2 *Evaluation*

A rotationally invariant model should produce the same PSNR on all rotations of the test dataset. Thus, in Figure 8.4 we plot the PSNR on the test datasets against their respective rotation angle.

We see that the fluctuations within both anisotropic and isotropic coupled models are much smaller than those within the uncoupled model. A choice of $\alpha = 0.41$ and $\gamma = 0$ yields a good balance between performance and rotation invariance for all models. However, rotation invariance can be driven to the extreme: A choice of $\alpha = 0.5$, which renders the choice of γ irrelevant, eliminates rotational fluctuations almost completely, but also drastically reduces the quality. The reason for this is given by Weickert et al. [379]: A value of $\alpha = 0.5$ decouples

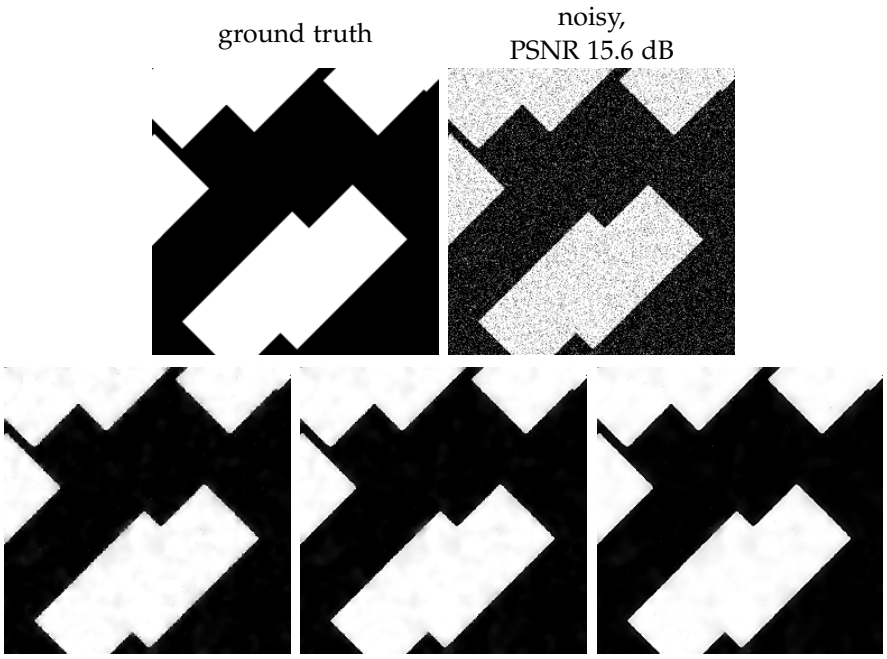


Figure 8.5: Visual comparison of denoising results for a rotation angle of 45°. The coupled models with $\alpha = 0.41$ achieve better denoising quality as they generalise better to the rotated data.

the image grid into two decoupled checkerboard grids which do not communicate except at the boundaries.

For the balanced choice of $\alpha = 0.41$, the anisotropic model consistently outperforms the isotropic one, as it can smooth along oriented structures. As the uncoupled model can only do this for structures which are aligned with the x- and y-axes, it performs better the closer the rotation is to 0° and 90°, respectively. Hence, it performs worst for a rotation angle of 45°. Thus, it does not achieve rotation invariance.

We measure the rotation invariance in terms of the variance of the test errors over the rotation angles. While the isotropic and anisotropic coupled models with $\alpha = 0.41$ achieve variances of 0.035 dB and 0.014 dB, the uncoupled model suffers from a variance of 1.25 dB. The extreme choice of $\alpha = 0.5$ even reduces the variances of the coupled models to 0.013 dB and $8.7 \cdot 10^{-4}$ dB, respectively.

A visual inspection of the results in Figure 8.5 supports this trend. Therein, we present the denoised results on an example from the test data set with 45° orientation. The uncoupled model suffers from ragged edges as the training on the differently rotated dataset has introduced a directional bias. The coupled isotropic model preserves the edges far better, and the coupled anisotropic model can even smooth along them to obtain the best reconstruction quality.

These findings show that the coupling effect leads to significantly better rotation invariance properties.

8.5 CONCLUSIONS

We have seen that the two-dimensional extension of our established connections between diffusion and neural networks allows to bring novel concepts for rotation invariance to the world of CNNs. The considered models inspire different activation function designs, which we summarise in Table 8.1.

The central design principle for rotation invariance is a coupling of operator channels. Diffusion models and their associated variational energies apply their respective nonlinear design functions to rotationally invariant quantities based on a coupling of multi-channel differential operators. Thus, the activation function as their neural counterpart should employ this coupling, too. Moreover, coupling image channels or scales in addition allows to create anisotropic models with better measures for structural information.

This strategy provides an elegant and minimally invasive modification of standard architectures. Thus, coupling activation functions constitute a promising alternative to the popular network designs of splitting orientations and group methods in orientation space.

This chapter concludes Part II. Overall, we have shown that in particular novel design ideas for activation functions can be justified from diffusion models and their numerical implementations, and that these ideas can be fruitful in a practical setting. We hope that our considerations provide one further contribution to the mathematical foundations of deep learning. In the following and final part, we show how ideas from diffusion and deep learning can help to improve inpainting-based image compression strategies.

Part III

IMAGE INPAINTING WITH HYBRID MODELS

9

INPAINTING WITH ANISOTROPIC SHEPARD INTERPOLATION

The present chapter initiates the final part of this thesis. It deals with an image inpainting model which combines the efficiency of a classical interpolation method with the reconstruction quality of anisotropic diffusion. Even though it does not involve any learning as the exception in this thesis, it nevertheless already constitutes an instance of a hybrid model as described in our third and final vision. We show that combining ideas from two research areas that have for a long time lived in separation can yield performant models without introducing too much additional complexity.

In the previous chapter, we have already extensively investigated the architectural differences between isotropic and anisotropic diffusion models. The present chapter is concerned with an anisotropic extension of a classical interpolation method. Similar to the approaches in Part I, our goal is to design a model with solid mathematical foundations and only few transparent parameter choices. The only difference is that we do not make use of any concepts from machine learning.

The starting point of this chapter is Shepard interpolation [336], which is a fast and simple method for interpolating sparse data. Particularly its low complexity makes it attractive in the context of compression codecs where interactive computation times are desirable [280].

Shepard interpolation computes a reconstruction from known data by a normalised weighted averaging. Typically, the weighting is determined by an inverse distance function which is rotationally invariant. However, for the case where the mask data are distributed regularly, additional directional information is encoded in the known data for free. This motivates us to combine the speed and simplicity of Shepard interpolation with ideas from edge-enhancing diffusion [368].

To this end, we introduce anisotropic Shepard interpolation, which allows the weighting function to adapt to the local directional structure of the available data. In particular, when these data are arranged in a regular fashion, it allows us to compute gradients and structure tensors on the respective regular mask which yields approximations of local orientations. In the same fashion as EED, we use this information to guide the influence function accordingly, thus achieving an anisotropic distribution of information. In an extensive evaluation, we show that this strategy does not only improve the inpainting quality, but also helps boost the performance of lossy image compression codecs significantly.

RELATED WORK Shepard interpolation [336] has originally been proposed as a general purpose interpolation strategy. An equivalent concept was suggested under the name of *normalised convolutions* by Knutsson and Westin [215]. Therein, a set of known sampling positions is used to reconstruct unknown positions by an inverse distance weighting: Each known point contributes to each unknown one according to their respective distance. Typically, the importance of points with larger distances decreases rapidly.

To improve the efficiency of the original formulation, Franke and Nielson [128] present a *modified Shepard's method* wherein only a fixed number of neighbouring points contribute to the reconstruction of an unknown position. The characteristics of the weighting function remain mostly unchanged. The seminal work of Renka [299] extends this idea by locally varying the radius of the weighting functions, increasing local adaptivity and thus reconstruction quality. Li et al. [232] compute the weights based on a measure of local redundancy: The less points are clustered around a known data point, the more it contributes to the reconstruction. Such isotropic modifications allow to control the influence of known points locally and are one important building block for our model.

As the underlying true function from which the known data points are sampled exhibits some directional structure, many works come up with strategies to estimate this structure from the known data. Tomczak [356] computes an *effective distance* between data points which incorporates the ratio of local anisotropy of the data. The effective distance is used in the inverse distance function and artificially moves points closer together when they are aligned along the locally dominant anisotropic axis, thus assigning higher weights to these points. In a similar manner, Ringaby et al. [300] compute an anisotropic distance measure in the context of image rectification. In a higher-dimensional setting, Lorenzi et al. [239] use such local distance measures to define the local interpolation regions as high-dimensional ellipsoids that are elongated along principal directions of the data.

The latter strategy is arguably the one which is related to our idea the most. However, we greatly benefit from assuming that known data points are sampled on a regular grid, whereas all previous references assume irregularly spaced data. This allows us to come up with a much simpler anisotropic measure based on an approximation of the structure tensor. Thus, we can easily control the size and the shape of the anisotropic weighting functions. We show that, in a pure inpainting setting, our anisotropic Shepard interpolation model elegantly and efficiently extends the set of anisotropic Shepard modifications.

The first instance where Shepard interpolation was used for image compression is the *regular grid codec with joint inpainting and prediction (RJIP)* codec of Peter [280]. He leverages the efficiency of Shepard interpolation to create a highly efficient inpainting-based compression

codec as an alternative to diffusion-based ones [135, 327]. The work of Mohideen et al. [258] extends the RJIP codec to colour images and constitutes the baseline of our image compression experiments. Within a simplified variant of their codec, we replace the standard isotropic Shepard interpolation by our anisotropic modification and show that it can boost the compression performance significantly, with only a moderate increase of computation time.

PUBLICATION INFORMATION The contents in this chapter have not appeared in a publication so far. A manuscript to be submitted to a journal is in preparation; see also Appendix D.

ORGANISATION OF THE CHAPTER In Section 9.1, we briefly review isotropic Shepard interpolation. Afterwards, in Section 9.2 we present our anisotropic modification and evaluate its effect on inpainting quality in Section 9.3. We then integrate our model into a compression codec in Section 9.4 to judge its performance in an actual compression setting in Section 9.5. Finally, we summarise our findings in Section 9.6.

9.1 REVIEW: ISOTROPIC SHEPARD INTERPOLATION

We consider a continuous greyscale image $f : \Omega \to \mathbb{R}$ which contains known data at a set of positions $K \subset \Omega$ on a rectangular image domain $\Omega \subset \mathbb{R}^2$. Shepard interpolation [336] computes the reconstruction u by accumulating neighbouring available information as

$$u(\boldsymbol{x}) = \frac{\int_{\boldsymbol{y} \in K} w(\boldsymbol{x} - \boldsymbol{y}) f(\boldsymbol{y})\, d\boldsymbol{y}}{\int_{\boldsymbol{y} \in K} w(\boldsymbol{x} - \boldsymbol{y})\, d\boldsymbol{y}}, \tag{9.1}$$

where $w : \mathbb{R}^2 \to \mathbb{R}$ is a scalar weighting function.

Thus, the reconstructed value at some position is computed as a weighted mean of the available data. One option is to use a Gaussian weighting function with a standard deviation σ depending on the percentage of known data. To accelerate the process, one can constrain the weighting functions to a rectangular window and truncate them at a given precision as a factor of the standard deviation σ; see e.g.[4, 280].

The above formulation is a continuous interpretation of the original discrete formulation by Shepard [336]. A notable difference of our variant is that the values at mask points are computed with the same formula as non-mask points, while the original expression fixes the values of known points. We have found that our modification helps with overall inpainting quality, in particular when performing tonal optimisation. Even though our method is technically an approximation instead of an interpolation, we still denote it as Shepard interpolation in the following for the sake of the original naming conventions.

9.2 ANISOTROPIC SHEPARD INTERPOLATION

It is obvious that the original Shepard interpolation constitutes an isotropic inpainting process if the weighting function is rotationally invariant. Yet, if the mask data are arranged in a regular fashion, it is easy to extract directional information of the original image at no additional storage cost.

We propose to modify the weighting function w based on structural information which is encoded in f. To this end, we adapt the weighting function at each mask position $\boldsymbol{y} \in K$ to obtain a set of functions $w_{\boldsymbol{y}}$. Our anisotropic Shepard interpolation computes the reconstruction u as

$$u(\boldsymbol{x}) = \frac{\int_{\boldsymbol{y}\in K} w_{\boldsymbol{y}}(\boldsymbol{x}-\boldsymbol{y}) f(\boldsymbol{y})\, d\boldsymbol{y}}{\int_{\boldsymbol{y}\in K} w_{\boldsymbol{y}}(\boldsymbol{x}-\boldsymbol{y})\, d\boldsymbol{y}}, \tag{9.2}$$

Note that the only difference to the isotropic variant (9.1) are the spatially varying weighting functions that depend on the local structure of the masked image f.

The gradient ∇f encodes the structural information of f which allows to define structure-adaptive weighting functions. As a weighting function we choose an oriented Gaussian with standard deviations σ_1, σ_2, and a rotation angle θ. The two standard deviations σ_1, σ_2 determine the major and minor directions of the Gaussian. The axis-aligned Gaussian is then rotated by the angle θ. For $\sigma_1 = \sigma_2$, we want to obtain a rotationally invariant Gaussian corresponding to the isotropic case.

A model which fulfils the above properties based on the structural information described by the image gradient $\nabla f = (f_x, f_y)^\top$ is given by

$$\sigma_1^2 = \sigma^2, \tag{9.3}$$
$$\sigma_2^2 = g\left(|\nabla f|^2\right) \sigma^2, \tag{9.4}$$
$$\theta = \arctan\left(-\frac{f_x}{f_y}\right), \tag{9.5}$$

where σ is an input parameter determining the base standard deviation of the Gaussian. In Figure 9.1, we visualise the relation between these parameters in a continuous setting.

The function $g(s^2)$ is a nonlinear diffusivity function. In our experiments, we found the rational Perona–Malik diffusivity [279] to be a suitable choice. Thus, it damps the variance σ^2 at image locations with dominant structures where the edge detector $|\nabla f|$ exceeds some contrast parameter λ. Choosing the constant diffusivity [195] yields the isotropic Shepard interpolation model.

The angle θ determines the rotation of the deformed Gaussian function. As ∇f points into the direction of the steepest ascent of f, the deformed Gaussian should be oriented along the orthogonal direction $\nabla^\perp f$. In two dimensions, a vector which is orthogonal to the

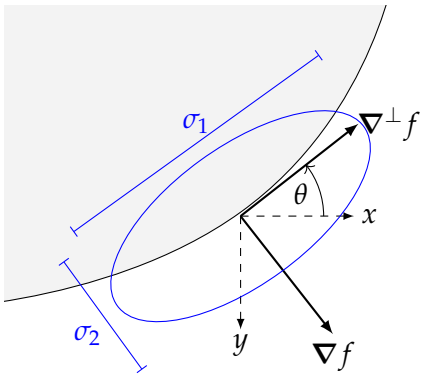


Figure 9.1: Local adaptation of the weighting function to the image structure in a continuous setting. The gradient ∇f spans a coordinate system rotated by θ w.r.t. the (x, y)-system. The gradient magnitude shrinks the level lines of the Gaussian kernel (blue) across dominant structures. This gives elliptic level lines with major and minor axes proportional to σ_1 and σ_2, respectively.

gradient can be easily constructed, e.g. as $\nabla^\perp f = (f_y, -f_x)^\top$. The angle of this vector in the (x, y)-coordinate system is given by (9.5).

In the $(\nabla^\perp f, \nabla f)$-coordinate system, the resulting Gaussian weighting function should scale the kernel along the principal directions by the standard deviations σ_1, σ_2 given above. Thus, we obtain

$$\boldsymbol{z}^\top \boldsymbol{\Sigma} \boldsymbol{z} = \begin{pmatrix} z_1 & z_2 \end{pmatrix} \begin{pmatrix} \frac{1}{2\sigma_1^2} & 0 \\ 0 & \frac{1}{2\sigma_2^2} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \tag{9.6}$$

as an argument for the Gaussian function, where $\boldsymbol{z}$ is the spatial difference between to positions in the $(\nabla^\perp f, \nabla f)$-coordinate system.

Let us now bring this argument to the (x, y)-coordinate system. To this end, we multiply a rotation by θ from the right, and its inverse from the left, yielding

$$\boldsymbol{R}_\theta^{-1} \boldsymbol{z}^\top \boldsymbol{\Sigma} \boldsymbol{z} \boldsymbol{R}_\theta = \boldsymbol{d}^\top \boldsymbol{R}_\theta^{-1} \boldsymbol{\Sigma} \boldsymbol{R}_\theta \boldsymbol{d}. \tag{9.7}$$

Here, $\boldsymbol{d}$ is the spatial distance between two positions in the (x, y)-coordinate system. Note that we used the relation $\boldsymbol{z} = \boldsymbol{R}_\theta \boldsymbol{d}$ to move the rotation matrices inside the expression.

Expressing the inner matrix as with parameters α, β, γ, we obtain

$$\begin{aligned} \begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix} &= \boldsymbol{R}_\theta^{-1} \Sigma \boldsymbol{R}_\theta \\ &= \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} \frac{1}{2\sigma_1^2} & 0 \\ 0 & \frac{1}{2\sigma_2^2} \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}, \end{aligned} \tag{9.8}$$

which, by additionally using the identity $\cos(\theta)\sin(\theta) = \frac{1}{2}\sin(2\theta)$ yields

$$\alpha = \frac{\cos^2(\theta)}{2\sigma_1^2} + \frac{\sin^2(\theta)}{2\sigma_2^2}, \tag{9.9}$$

$$\beta = -\frac{\sin(2\theta)}{4\sigma_1^2} + \frac{\sin(2\theta)}{4\sigma_2^2}, \tag{9.10}$$

$$\gamma = \frac{\sin^2(\theta)}{2\sigma_1^2} + \frac{\cos^2(\theta)}{2\sigma_2^2}. \tag{9.11}$$

The resulting Gaussian weighting function takes the distance between two spatial positions $\boldsymbol{d} = (d_1, d_2)^\top = \boldsymbol{x} - \boldsymbol{y}$ and computes

$$G(\boldsymbol{d}) = \exp\left(-\alpha d_1^2 + 2\beta d_1 d_2 - \gamma d_2^2\right). \tag{9.12}$$

This function forms the basis of the anisotropic Shepard interpolation strategy as it adapts to the structural information of the known data.

ADAPTATION TO COLOUR IMAGES So far, we assumed that the input is a grey value image. However, a simple adaptation of the gradient computation allows us to adapt our model to colour images. For an RGB image $f = (f_r, f_g, f_b)^\top$ with three channels, the colour structure tensor [98]

$$\boldsymbol{J} = \sum_{c \in \{r,g,b\}} \nabla f_c \nabla f_c^\top \tag{9.13}$$

helps to find the dominant direction over all channels. It is a 2×2 positive semi-definite matrix. Its normalised eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ span a coordinate system similar to that of ∇f, $\nabla^\perp f$ in the grey value case, and its eigenvalues $\nu_1, \nu_2 \geq 0$ measure the local contrast along the eigenvectors. Thus, we can simply use the entries of the dominant eigenvector $\boldsymbol{v}_1 = (v_{1,x}, v_{1,y})^\top$ and the corresponding eigenvalue ν_1 in the parameter computations:

$$\sigma_1^2 = \sigma^2, \tag{9.14}$$

$$\sigma_2^2 = g(\nu_1)\,\sigma^2, \tag{9.15}$$

$$\theta = \arctan\left(-\frac{v_{1,x}}{v_{1,y}}\right). \tag{9.16}$$

The channel-wise gradient approximations are computed as in the grey value case. Note that our colour adaptation is also compatible with the grey value setting: The single channel structure tensor $\nabla f \nabla f^\top$ has an eigenvector ∇f with eigenvalue $|\nabla f|^2$, and an eigenvector $\nabla^\perp f$ with eigenvalue zero.

Moreover, we have also implemented a luma preference (LP) mode inspired by diffusion-based colour image compression techniques [281] in analogy to [258]. Therein, the gradient approximation for a colour

image is computed only on the luma channel of a YCbCr version of the image. In the pure inpainting setting, without quantisation and different mask densities in each channel, the luma preference version performed negligibly worse than the RGB variant, which is why we do not consider it in the inpainting setting.

However, this result is promising for the compression context: An LP mode allows to choose different mask densities in the luma (Y), and the chroma (Cb and Cr) channels, as it is beneficial to sample structural information with higher rates than colour information. As the LP mode is on par with the RGB mode in the inpainting setting, we expect that an embedding into a compression framework can benefit from the higher sampling rates in the luma channel.

DISCRETE IMPLEMENTATION When using a discrete image $\boldsymbol{f} \in \mathbb{R}^{N_x \times N_y}$ which contains known data at a set of positions $K \subset \Omega$ on an image domain $\Omega = \{1, \dots, N_x\} \times \{1, \dots, N_y\}$, a discrete variant of the anisotropic Shepard interpolation computes the reconstruction $u_{i,j}$ at pixel position i, j as

$$u_{i,j} = \frac{\sum_{(p,q)\in K} w_{p,q}(i-p, j-q) f_{p,q}}{\sum_{(p,q)\in K} w_{p,q}(i-p, j-q)}, \tag{9.17}$$

where $w_{p,q}$ are the space-adaptive weighting functions at known pixel positions.

In this setting, the gradient approximation $(\nabla f)_{i,j}$ must be computed carefully. If the mask pixels obey uniform sampling distances d_x, d_y in x- and y-direction, then we can obtain a gradient approximation $(\nabla f)_{i,j} = ((f_x)_{i,j}, (f_y)_{i,j})^\top$ by central differences:

$$(f_x)_{i,j} \approx \frac{f_{i+d_x,j} - f_{i-d_x,j}}{2 d_x h_x}, \tag{9.18}$$

$$(f_y)_{i,j} \approx \frac{f_{i,j+d_y} - f_{i,j-d_y}}{2 d_y h_y}, \tag{9.19}$$

where h_x, h_y are the pixel sizes in x- and y-direction. The central difference in x-direction considers the two pixels which lie d_x positions away from i. This ensures that if we want to compute the gradient approximation at a mask position $(p, q) \in K$, the closest mask pixels are considered for the approximation. This yields consistent gradient approximations at mask pixels. Inconsistent approximations at pixels in unknown image regions can be safely ignored, as they are not required for the computation of the reconstruction.

FAST TONAL OPTIMISATION When inpainting sparse data, the given data are often not optimal. A slight manipulation of the given data can significantly improve reconstruction quality [183]. This has also been observed in the context of Shepard interpolation [280]. In contrast to diffusion-based inpainting methods, tonal optimisation for

Shepard interpolation can be extremely fast [280], and even closed-form optimal solutions are possible [258].

In the anisotropic setting, no closed-form solution is available. Still, we can make use of the superposition principle to obtain a fast tonal optimisation for our model. Changing the value $f_{p,q}$ at a mask point $(p,q) \in K$ changes the following data: The pixel value $f_{p,q}$ itself, but also the gradient approximation at (p,q) and all four neighbouring mask pixels.

In practice, we compute numerator and denominator of (9.2) separately, and truncate the Gaussian weighting functions. For the novel anisotropic setting, this gives a set of influence windows instead of a fixed one. This allows us to quickly update a reconstruction for a new tonal value $f_{p,q}^{\text{new}}$:

1. For each of the five affected pixels $f_{p,q}$, $f_{p+d_x,q}$, $f_{p-d_x,q}$, $f_{p,q+d_y}$, and $f_{p,q-d_y}$ we remove their influence on all reconstructed pixels within their respective influence windows. This is done by subtracting the associated terms in both numerator and denominator of (9.2).

2. We insert the new value $f_{p,q}^{\text{new}}$ and update the gradient approximations and structure tensors at the five affected positions.

3. Based on the new approximations, we update the corresponding anisotropic weighting functions $w_{p,q}$.

4. Within the updated influence windows, we add the refined contribution of the five pixels to the numerator and the denominator.

5. In the union of old and new influence windows, we re-compute the reconstruction and update the reconstruction error locally.

We repeat these steps for all mask pixels in a random order. Each time, we check if the reconstruction error is improved by changing the current pixel's value to both neighbouring quantised values. If this yields a better reconstruction, the value is kept, otherwise the change is reverted. One full pass through all mask pixels is denoted an epoch, and we stop the process once a fixed number of epochs is computed, or if the improvement becomes too small.

Compared to the isotropic Shepard interpolation, tonal optimisation in the anisotropic model also includes changes in the directional information. The optimisation automatically takes into account if it pays off to focus on improving grey and colour values or on refining the directional information. Thus, we expect that the gap between tonally optimised and original interpolations will be larger in the case of anisotropic Shepard interpolation.

9.3 INPAINTING EXPERIMENTS

Let us now evaluate the potential of anisotropic Shepard interpolation within several experiments. For now, we consider the pure inpainting case without any quantisation or compression.

To this end, we consider the greyscale images *cameraman*, *house* [342], and *trui* [402], as well as three colour images from the Kodak image database [115] *kodim19* (lighthouse), *kodim20* (airplane), and *kodim23* (parrots). To directly compare colour and greyscale setting, we additionally consider both a greyscale and a colour version of *peppers*.

For each image we employ a range of mask grid sizes $d = d_x = d_y$ from $d = 2$ to $d = 7$. This corresponds to a mask density $\frac{100}{d^2}\%$. On each combination of image and grid size, we compare the performance of isotropic and anisotropic Shepard interpolation, before and after tonal optimisation (TO). As an error measure, we consider the peak-signal-to-noise ratio (PSNR).

For all approaches, we optimise the parameters with a golden section search. In the case of isotropic Shepard interpolation, the only parameter is the standard deviation σ of the Gaussian weighting function. For anisotropic Shepard interpolation, the contrast parameter λ of the diffusivity is optimised in addition. In case of tonal optimisation, we interleave the parameter optimisation with the tonal optimisation iteratively, until the error does not improve significantly.

QUALITATIVE ANALYSIS. Figure 9.2 presents two reconstruction examples for the images *trui* and *cameraman* for different sampling rates. The isotropic Shepard interpolation leaves the images blurred, and strong block artefacts arise from the subsampled grid.

Our anisotropic Shepard model alleviates these problems: The image is generally sharper, and in particular edges are enhanced. This is especially apparent for diagonal edges, e.g. the hat and scarf contours of *trui*, and the camera tripod in *cameraman*. The PSNR values support this visual impression, as the anisotropic ones are significantly higher than those of the isotropic model.

The effect on the larger colour image *kodim23* is presented in Figure 9.3. A zoom onto the right macaw shows the benefits, but also the limitations of our anisotropic model. The coarse parts of the face texture are enhanced, and background structures appear less blocky. However, the pattern around the eye cannot be preserved by neither the isotropic nor the anisotropic model, as the sampling is too coarse.

QUANTITATIVE ANALYSIS. Table 9.1 lists the reconstruction errors on a selected subset of the test images.

Throughout the experiments, the improvements of the tonally optimised anisotropic model over the tonally optimised isotropic one are significant. On average, the improvement in all experiments is 0.86 dB,

original | isotropic + TO, PSNR 31.33 dB | anisotropic + TO, PSNR **32.50** dB

(a) *trui* with d=4

original | isotropic + TO, PSNR 26.57 dB | anisotropic + TO, PSNR **27.52** dB

(b) *cameraman* with d=5

Figure 9.2: Qualitative comparison between anisotropic and isotropic Shepard interpolation on *trui* (a) and *cameraman (b)*. Anisotropic images occur less blurred. In particular, diagonal edges are enhanced significantly.

where the average improvement on grey value images is significantly higher with 1.04 dB than that on the colour images with 0.69 dB.

Moreover, we observe that with decreasing mask density, the improvements stagnate slightly. On average, for $d = 2$, the anisotropic model is 0.87 dB better, while for $d = 7$, the improvement is only 0.76 dB. This is to be expected: With coarser sampling, the directional data used in the anisotropic model becomes more unreliable. Thus, with decreasing density the anisotropic model tends towards the isotropic one. On the individual image, this effect cannot always observed as coarse masks are not necessarily subsets of fine ones. This observation again motivates the use of an LP mode in a compression setting: We expect that the higher sampling rates in the luma channel mitigate this effect.

INFLUENCE OF COLOUR. As we have seen, the improvements of our anisotropic model compared to the isotropic one are not as large on colour images. To this end, Figure 9.4 visualises the reconstruction of *peppers* for both models with $d = 5$. We conjecture that, as isotropic

original zoom into original

isotropic + TO, PSNR 28.22 dB zoom into isotropic + TO

anisotropic + TO, PSNR **28.75** dB zoom into anisotropic + TO

Figure 9.3: Comparison between anisotropic and isotropic Shepard interpolation on the image *kodim23* with $d = 5$. The anisotropic model reconstructs outlines such as the beak and face pattern more faithfully. Moreover, background structures appear smoother.

Table 9.1: Comparison of PSNR values in decibels (dB) on selected images for isotropic (iso.) and anisotropic (aniso.) Shepard interpolation, and the corresponding tonally optimised counterparts (+TO). The mask density is defined by the sampling distance $d = d_x = d_y$. The best performance for each row is highlighted in bold.

image	d	density in %	iso.	aniso.	iso. + TO	aniso. + TO
cameraman (greyscale)	2	25.0%	34.19	35.91	37.12	**38.28**
	3	11.1%	29.67	30.22	31.82	**32.74**
	4	6.2%	26.80	27.05	28.49	**29.46**
	5	4.0%	25.05	25.21	26.53	**27.50**
	6	2.8%	23.91	24.08	25.35	**26.24**
	7	2.0%	22.93	23.06	24.53	**25.51**
peppers (greyscale)	2	25.0%	31.86	32.31	33.56	**34.12**
	3	11.1%	29.56	30.12	31.46	**32.49**
	4	6.2%	27.91	28.38	29.80	**31.03**
	5	4.0%	26.63	27.03	28.51	**29.75**
	6	2.8%	25.57	25.84	27.45	**28.56**
	7	2.0%	24.82	25.15	26.61	**27.80**
peppers (colour)	2	25.0%	29.81	30.15	31.43	**31.78**
	3	11.1%	28.08	28.59	29.88	**30.56**
	4	6.2%	26.80	27.26	28.64	**29.58**
	5	4.0%	25.73	26.14	27.58	**28.46**
	6	2.8%	24.85	25.17	26.69	**27.72**
	7	2.0%	24.21	24.56	25.97	**27.01**
kodim20 (colour)	2	25.0%	28.37	28.62	30.30	**31.79**
	3	11.1%	26.16	26.35	27.97	**29.18**
	4	6.2%	24.84	25.02	26.56	**27.47**
	5	4.0%	24.18	24.38	25.75	**26.44**
	6	2.8%	23.64	23.83	25.16	**25.78**
	7	2.0%	23.22	23.43	24.69	**25.16**

original, greyscale | isotropic + TO, PSNR 28.52 dB | anisotropic + TO, PSNR **29.75** dB

(a) greyscale version of *peppers* with d=5

original, colour | isotropic + TO, PSNR 27.61 dB | anisotropic + TO, PSNR **28.48** dB

(b) colour version of *peppers* with d=5

Figure 9.4: Comparison between anisotropic and isotropic Shepard interpolation on a grey and a colour version of *peppers* for $d = 5$.

Shepard interpolation already gives worse quality in the colour setting than in the greyscale setting, the margin to our anisotropic model becomes smaller in turn.

EFFECT OF TONAL OPTIMISATION. The tonal optimisation is the crucial part which massively increases the performance gap between anisotropic and isotropic Shepard interpolation. While the average improvement from isotropic to anisotropic with tonal optimisation is 0.86 dB, it is only 0.38 dB without it. This becomes even more apparent when splitting the average for grey and colour images: Without tonal optimisation, greyscale image reconstructions are improved by 0.57 dB, while colour image reconstructions are only improved by 0.19 dB.

On the contrary, the anisotropic model can leverage the freedom of tonal optimisation much more than the isotropic one: On average, tonal optimisation improves reconstructions by 1.80 dB in the isotropic case, but by 2.28 dB in the anisotropic case. This confirms our previous conjecture that the implicit optimisation of the interpolation directions in the anisotropic model helps drastically.

The image *kodim20* and its reconstructions in Figure 9.5 highlights this effect. The zoom into the aircraft propeller region shows that in

the isotropic model, the tonal optimisation partly corrects the block artefacts and the lacking colour information. However, the result still contains ragged edges. The anisotropic reconstruction without tonal optimisation suffers from a different problem: Due to the coarse sampling, the interpolation directions at the propeller edges alternate between only two possible choices, creating a zig-zag pattern. However, the tonal optimisation is able to cure this problem almost completely.

9.4 APPLICATION TO COMPRESSION

Let us now embed the anisotropic Shepard interpolation model into a compression context. Therein, several new questions arise, the main one being if the benefits of the anisotropic approach carry over. Additional factors such as a quantisation of the input data influence the quality of the reconstruction, and, in addition for the anisotropic model, also the accuracy of interpolation directions.

To this end, we build a simple codec based on ideas from [258, 280]. First, we start with a codec with scalar quantisation for grey and colour images which proceeds as follows:

Step 1: Mask optimisation. We optimise the mask density by stepping through the possible grid sizes d, starting with $d = 1$. We deliberately include the option of storing the full image, as quantisation alone is sometimes enough to reach low compression ratios.

Step 2: Parameter optimisation. For each grid size, we optimise the parameters λ and σ of the inpainting model with a golden section search.

Step 3: Quantisation. Afterwards, optimise the number of quantisation levels q for a desired compression ratio. To this end, we write the current mask data along with the image size, the parameters h, q, λ, and σ into a file and compress it with LPAQ [244].

We repeat steps 1–3 until we find the combination of grid size h and quantisation q which yields the lowest error for the given compression ratio.

Optional Step 4: Tonal optimisation. If desired, the mask data are tonally optimised while keeping the grid size and the quantisation range fixed. This step is repeated multiple times and interleaved with another optimisation of the inpainting parameters σ and λ, until the error does not improve significantly.

One of the crucial differences to the codecs of [258, 280] is that the tonal optimisation is excluded from the grid and quantisation optimisation due to runtime limitations. Moreover, no prediction is involved and the data are compressed with an off-the-shelf algorithm for simplicity.

original

isotropic,
PSNR 24.18 dB

isotropic + TO,
PSNR 25.78 dB

anisotropic,
PSNR 24.38 dB

anisotropic + TO,
PSNR **26.48** dB

(a) full size images

original isotropic anisotropic iso. + TO aniso. + TO

(b) zooms

Figure 9.5: Effect of tonal optimisation for the image *kodim20* for $d = 5$. A zoom into the anisotropic propeller region shows the benefit of anisotropic tonal optimisation: Not only colour information is adapted, but also the interpolation directions.

VARIANT: VECTOR QUANTISATION. Following [258], one variant that we implement is the option of vector quantisation for colour images. To this end, we simply replace the uniform quantisation in the codec above by vector quantisation [144]. Instead of quantising each colour channel independently, vector quantisation defines a sparse set of colours in the RGB space. While in the uniform quantisation case a quantisation parameter q allows for q^3 possible colours, it only yields q colours for the case of vector quantisation.

VARIANT: LP MODE. We also implement an LP mode for colour image compression. Therein, the interpolation directions are only computed from the luma channel of a YCbCr version of the input image. Moreover, it allows to choose different mask densities d_ℓ, d_c and quantisation levels q_ℓ, q_c for the luma (Y) and the chroma (Cb and Cr) channels. To steer the trade-off between the densities, we introduce a luma factor $f \in (0, 1)$ as in [258] which determines the ratio between the budget for the luma and the chroma channels.

With different luma and chroma densities, the need for two dedicated inpainting parameters σ_ℓ and σ_c arises. For the anisotropic model, we also introduce individual anisotropy parameters λ_ℓ and λ_c.

For the LP mode, the steps 1–3 described above are repeated for both luma and chroma channels and the corresponding parameters.

9.5 COMPRESSION EXPERIMENTS

We test the codecs described above on the full Kodak image database [115]. For each codec, we prescribe target compression ratios between 20 : 1 and 150 : 1 in steps of 10. We compare the average PSNR on the full database while averaging the achieved compression ratios in each step.

QUANTITATIVE ANALYSIS In a first experiment, we investigate the benefits of the anisotropic over the isotropic model. In Figure 9.6, we present four graphs comparing isotropic and anisotropic, each with and without tonal optimisation, for all codec variants: a greyscale version, RGB versions with scalar and vector quantisation, as well as the LP mode variant. For the LP mode, we evaluate luma factors from 0.5 to 0.9 in steps of 0.1 and choose the best performing one for each image and compression ratio.

Without tonal optimisation, the benefit of anisotropy is almost negligible. To some extent, we have already observed this in the pure inpainting setting. However, in the compression context, this effect is more severe since the set of possible grey values respectively colours is drastically reduced. This heavily impacts the possible values for interpolation directions which are computed from the mask points, which now reflect a small set of directional choices. As incorrect strongly an-

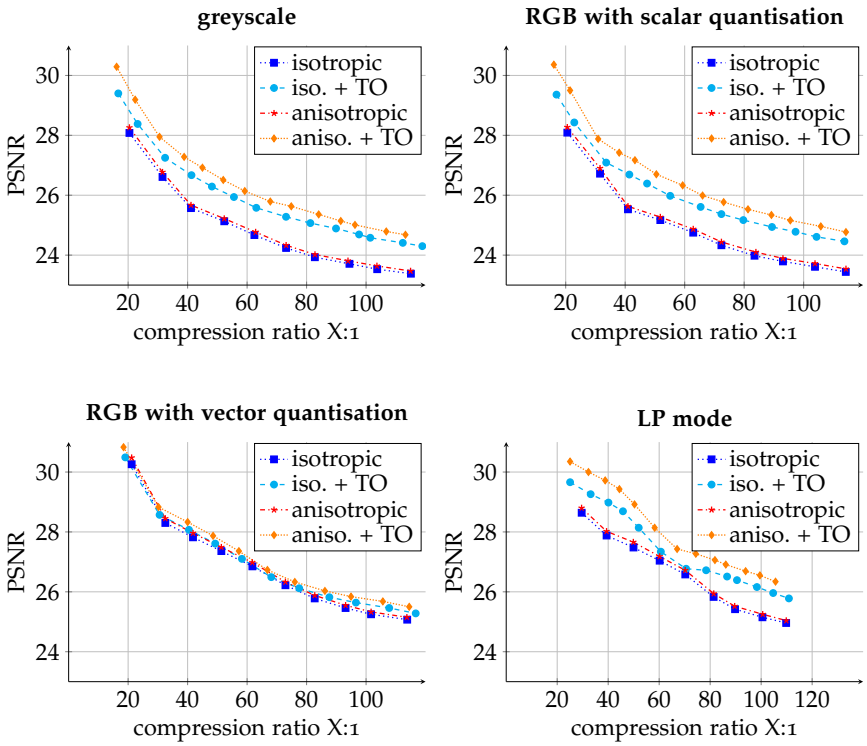


Figure 9.6: Average reconstruction error (PSNR) versus average compression ratio for all codec versions on the full Kodak database. Without tonal optimisation, the anisotropic model does not improve reconstructions significantly. With tonal optimisation, improvements are significant except for the vector quantisation codec.

isotropic kernels produce worse results than mediocre isotropic ones, no real benefit can be gained and the model resorts to an isotropic choice.

With tonal optimisation, however, the results change drastically. Note that the curves with tonal optimisation are shifted to the left, since the tonal optimisation is not accounted for in the selection of grid size and quantisation levels. Thus, tonal optimisation typically increases the entropy of the mask data, lowering the compression ratio.

In the greyscale case, the tonally optimised anisotropic codec shows improvements over the tonally optimised isotropic one of up to 0.9 dB. The higher the compression ratio, the smaller the improvement becomes. This is intuitive, since higher ratios imply stronger quantisation and thus less flexibility for anisotropic kernels.

The results for the RGB codec with scalar quantisation are almost identical, as it is the natural extension of the greyscale codec to colour images.

Interestingly, the RGB codec with vector quantisation paints a different picture. Therein, the benefits of both tonal optimisation and anisotropy are small compared to the other cases. This is a natural

original

isotropic, RGB scalar
ratio 72.7 : 1, PSNR 28.9 dB

anisotropic, RGB scalar,
ratio 73.4 : 1, PSNR 29.6 dB

isotropic, RGB vector
ratio 75.9 : 1, PSNR 29.5 dB

anisotropic, RGB vector
ratio 72.9 : 1, PSNR 30.0 dB

isotropic, LP mode, $f = 0.7$
ratio 76.9 : 1, PSNR 30.3 dB

anisotropic, LP mode, $f = 0.7$
ratio 73.0 : 1, PSNR **31.1** dB

Figure 9.7: Visual comparison of colour codec variants on the image *kodim23* from the Kodak database. All images are tonally optimised. The LP mode offers the best balance between colour and structure reconstructions.

effect of vector quantisation: A uniform quantisation with q levels allows for q^3 possible colours for an RGB image, while a vector quantisation directly specifies q possible colours. Thus, the options for tonal optimisation are limited even more, both concerning the mask data as well as the interpolation directions.

The behaviour of the LP codecs is again closer to the RGB scalar case. However, the shift in compression ratio through tonal optimisation is stronger than in the other versions, indicating that more tonal optimisation is performed.

QUALITATIVE ANALYSIS The different codec strategies are highlighted on the example image *kodim23* in Figure 9.7. The RGB scalar codec selects relatively coarse grid sizes to allow for a large colour set. The coarse grid size blurs the image, and the uniform quantisation introduces dithering-like artefacts. The anisotropic extension is able to obtain sharper outlines in some parts of the image.

The RGB vector codec can select much finer grid sizes of in exchange for a smaller number of colours. The result of the coarse colour selection is clearly visible in the form of dithered colour transitions.

The LP mode codec can focus on storing more mask points in the luma channel with with less grey values, while coarsely sampling the chroma channels, but allowing a larger variety of colours. From the finer luma sampling, the anisotropic model can benefit and significantly improve the result.

COMPARISON OF CODEC VARIANTS We can also interpret the results of the colour experiments from a different perspective: Figure 9.8 compares the codec variants. For each setting of isotropic and anisotropic, with and without tonal optimisation, we compare the performance of RGB scalar, RGB vector, and LP mode codecs.

Without tonal optimisation, the isotropic and the anisotropic setting are almost identical: The LP mode codec and the vector quantisation codec perform on par, with the RGB scalar codec performing significantly worse. Tonal optimisation improves both the LP mode codec and the one with scalar quantisation. As already highlighted, the RGB vector codec only improves slightly.

Thus, a central insight is the following: The more colours are available, the better the anisotropic model becomes. As a consequence, we expect even more improvements when the tonal optimisation is included within the grid size and quantisation optimisation: A larger set of colours might be beneficial in a tonally optimised setting such that it pays off to trade storage cost for better quality.

A comparison between our anisotropic codec and that of Mohideen et al. [258] highlights the potential of anisotropic Shepard interpolation. Figure 9.9 compares the two codecs with tonal optimisation for each variant. As a transform-based competitor, we consider JPEG [278]. We

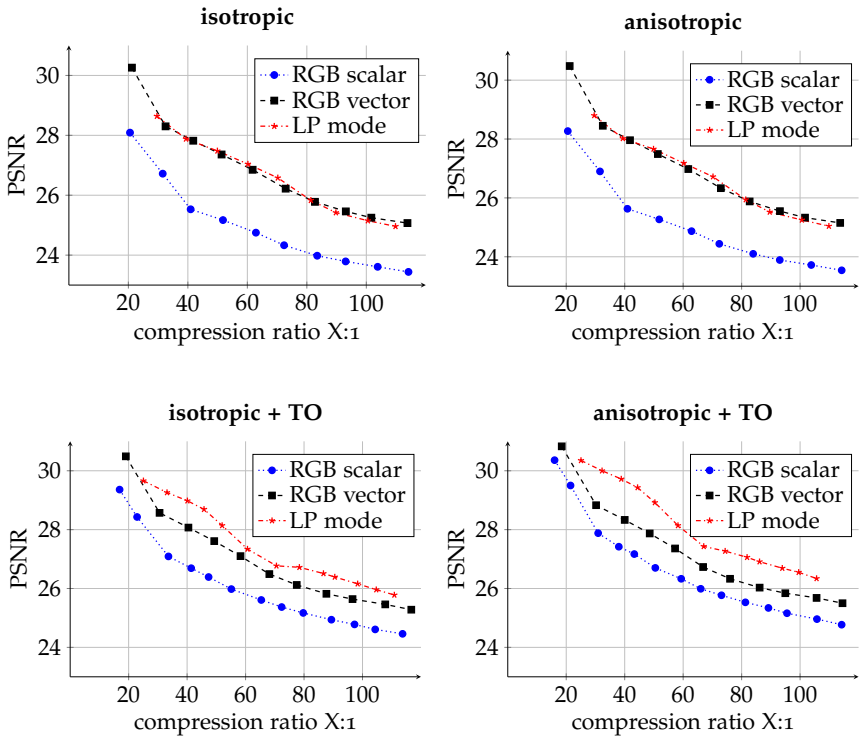


Figure 9.8: Comparison of codec variants for the colour version of the full Kodak database. The anisotropic RGB scalar and LP codecs improve significantly, with LP performing the best.

switch to MSE as a quality measure, as Mohideen et al. [258] provide only MSE values. Lower MSE values denote a better quality.

Even with a more simplistic coding framework and optimisation pipeline, our anisotropic models can outperform the isotropic ones of Mohideen et al. [258] for all the RGB scalar and LP codec variants. As the vector quantisation codec was relatively unaffected by the anisotropic extension, it is not surprising that also in this comparison, there is no clear favourite. The clear boost of the LP mode is what we had hoped for: Higher mask densities in the luma channel allow for a high adaptation of the anisotropic kernels.

What does this imply w.r.t. the goal of beating JPEG with Shepard interpolation? In the best performing LP mode, we see that the simple idea of anisotropic Shepard interpolation already halves the distance in MSE to the JPEG results. Thus, anisotropic Shepard interpolation is an important contribution towards simple yet efficient codecs that can beat JPEG. Thus, it pays off to invest additional efforts into a more sophisticated codec.

9.6 CONCLUSIONS

A simple modification of the isotropic Shepard interpolation model yields an anisotropic variant with better performance and only minor

overhead. In the pure inpainting setting, remarkable improvements can be achieved, especially for high mask densities.

Our experiments yield two central insights into the model: Structure-adaptive weighting functions allow for overall better reconstructions, and they can benefit more from tonal optimisation as interpolation directions are implicitly optimised as well.

The anisotropic Shepard interpolation is especially appealing for a compression context, as it does not need any additional information besides the regular inpainting mask and the associated mask data. Thus, it can create better reconstructions for the same storage cost. As we have shown, the anisotropic model can help to decrease the gap between Shepard codecs such as [258, 280] and its transform-based competitor of JPEG.

This is not the end of the road: We think that improving the coding aspect of our codec can improve the reconstructions even more, and combining it with other ideas for improving Shepard interpolation can lead to a codec which can outperform JPEG.

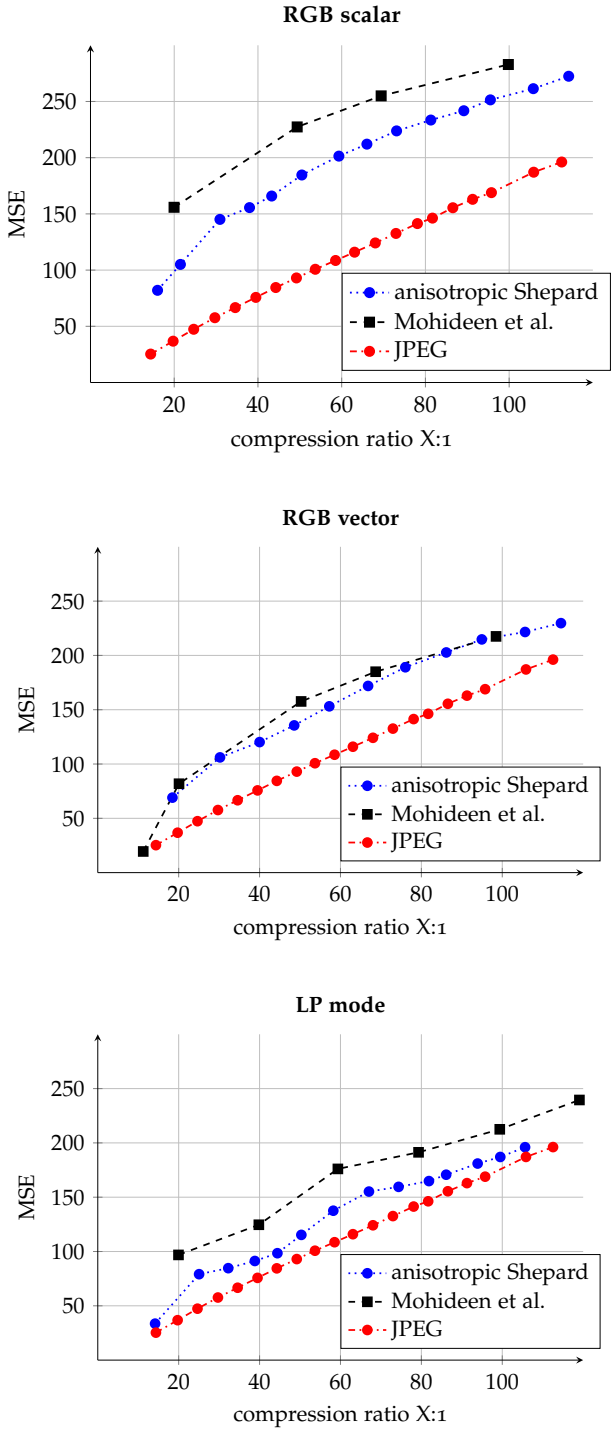


Figure 9.9: Comparison of our codec variants against the results of Mohideen et al. [258]. Our anisotropic models outperform the competitors in the RGB scalar and LP cases. Our anisotropic LP mode performs best.

10

LEARNING SPARSE MASKS FOR DIFFUSION INPAINTING

The final technical contribution of this thesis combines several important aspects from the previous chapters. At the example of diffusion-based inpainting, it unifies deep learning and classical methods in a practical approach.

In the previous chapter, we worked with regular inpainting masks. Freely optimised masks can boost the inpainting quality by a large amount, but optimising these masks is a challenging combinatorial problem. While there are theoretical results on optimal masks [39], practical implementations are often not convincing albeit highly efficient. On the other hand, stochastic mask optimisation strategies [246] produce high quality masks, but are computationally expensive.

We combine efficiency and quality of mask optimisation for PDE-based inpainting with the help of deep learning. To this end, we design a hybrid architecture which, to the best of our knowledge, constitutes the first instance of learned sparse masks for PDE-based inpainting.

We present a model for learning sparse inpainting masks for homogeneous, biharmonic, and edge-enhancing diffusion inpainting. We employ two networks: one which generates a sparse inpainting mask, and one which acts as a surrogate solver for the diffusion inpainting PDE. By using different loss functions for the two networks, we optimise both inpainting quality and fidelity to the inpainting equation.

The use of a surrogate solver is a crucial novelty. It reproduces results of a diffusion-based inpainting process without having to perform backpropagation through iterations of a numerical solver. This replicates the full inpainting pipeline to efficiently train a mask optimisation model.

We then evaluate the quality of the learned masks in a learning-free inpainting setting. Our model combines the speed of instantaneous mask generation approaches [39] with the quality of stochastic optimisation [246]. Thus, we reach a new level in sparse mask optimisation for diffusion-based inpainting.

RELATED WORK Diffusion-based inpainting plays a vital role in image and video compression [17, 135, 327], denoising [5], and many more. A good inpainting mask is crucial for successful image inpainting. Current approaches for the spatial optimisation of sparse inpainting data in images can be classified in four categories.

1. *Analytic Approaches.* Belhachmi et al. [39] have shown that in the continuous setting, optimal masks for homogeneous diffusion inpainting can be obtained from the Laplacian magnitude of the image. In practice this strategy is very fast, allowing real-time inpainting mask generation by dithering the Laplacian magnitude. However, the reconstruction quality is lacking, mainly due to limitations in the quality of the dithering operators [183, 246]. Moreover, analytic approaches for biharmonic and EED inpainting are not available.

2. *Nonsmooth Optimisation Strategies.* Several works [50, 76, 183, 271] consider sophisticated nonsmooth optimisation approaches that offer high quality, but do not allow to specify the desired mask density in advance. Instead one influences it by varying a regularisation parameter, which requires multiple program runs, resulting in a slow runtime. Moreover, adapting the model to different inpainting approaches is not trivial.

3. *Sparsification Methods.* They successively remove pixel data from the image to create an adaptive inpainting mask. The *probabilistic sparsification (PS)* of Mainberger et al. [246] is an example thereof; see Section 3.6.2. Sparsification strategies are generic as they work with various inpainting operators such as diffusion-based ones [183, 246] or interpolation on triangulations [97, 250]. Moreover, they allow to specify the desired mask density in advance. However, they are also computationally expensive as they require many inpaintings to judge the importance of individual data points to the reconstruction. This also makes their computation time dependent on the complexity of the inpainting operator. Due to their simplicity and their broad applicability, sparsification approaches are the most widely used mask optimisation strategies.

4. *Densification.* Densification strategies [77, 93, 208] start with empty or very sparse masks and successively populate them. This makes them reasonably efficient, while also yielding good quality. They are fairly easy to implement and work well for PDE-based [77, 93] and exemplar-based [208] inpainting operators. Still, they require multiple inpainting steps in the range of 10 to 100 to obtain a sufficiently good inpainting mask.

In order to escape from suboptimal local minima, the Categories 3 and 4 have been improved by NLPE [246] (see Section 3.6.2), at the expense of additional inpaintings and runtime. Moreover, it is well-known that optimising the grey or colour values of the mask pixels – so-called tonal optimisation – can boost the quality even further [183, 246]. Also the approaches of Category 2 may involve tonal optimisation implicitly or explicitly.

Qualitatively, carefully tuned approaches of Categories 2–4 play in a similar league, and are clearly ahead of Category 1. However, their runtime is also substantially larger than Category 1, mainly due to the many inpaintings that they require. Last but not least, all aforementioned approaches are fully model-based, in contrast to most recent approaches in image analysis that benefit from deep learning ideas.

The goal of the present paper is to show that the incorporation of deep learning can give us the best of two worlds: a real-time capability similar to Category 1, and a quality similar to Categories 2–4. In order to focus on the main ideas and to keep things simple, we do not consider tonal optimisation, but is equally possible for our novel approach.

Learning-based inpainting has also been successful in recent years. Following the popular work of Xie et al. [397], several architectures and strategies for inpainting have been proposed; see e.g. [176, 197, 237, 277, 321, 400, 403] However, inpainting from sparse data is rarely considered. Vašata et al. [364] present sparse inpainting based on Wasserstein generative adversarial networks. Similarly, Ulyanov et al. [360] consider inpainting from sparse data without mask generation. Dai et al. [91] present a trainable mask generation model from an adaptive sampling viewpoint. Our approach is the first to combine deep learning for mask optimisation for PDE-based inpainting in a transparent and efficient way.

PUBLICATION INFORMATION The presented work is set to appear as a conference publication of Alt et al. [9] at the 2022 Iberian Conference on Pattern Recognition and Image Analysis. This chapter is supplemented with additional experiments for biharmonic and EED inpainting.

ORGANISATION OF THE CHAPTER In Section 10.1, we briefly review data optimisation strategies for diffusion-based inpainting. Afterwards in Section 10.2, we introduce our model for learning inpainting masks. We evaluate the quality of the learned masks in Section 10.3 before presenting our conclusions in Section 10.4.

10.1 REVIEW: DATA OPTIMISATION FOR INPAINTING

As a reminder, diffusion inpainting restores missing information in a continuous greyscale image $f : \Omega \to \mathbb{R}$ on some rectangular domain Ω, where image data is only available on an inpainting mask $K \subset \Omega$ by computing the reconstruction u as the solution to the PDE

$$(1 - c)\, Lu - c\,(u - f) = 0. \tag{10.1}$$

with reflecting boundary conditions. Most diffusion-based inpainting models consider binary values for the confidence function c, however, it is also possible to use non-binary confidence functions [185], which we use to our advantage.

The differential operator L determines the type of inpainting: The choice $L = \Delta$ leads to homogeneous diffusion inpainting [195], biharmonic inpainting is achieved with $L = -\Delta^2$, and $L = \boldsymbol{\nabla}^\top(\boldsymbol{D}(\boldsymbol{\nabla}_\sigma u)\,\boldsymbol{\nabla} u)$ is EED inpainting [368, 390]; see also Sections 3.1.1 and 3.6.2. In the latter, one typically uses a Charbonnier diffusivity with a contrast parameter λ and a smoothing scale σ.

In practice, optimising the discrete binary mask $\boldsymbol{c}$ of the discrete inpainting equation is crucial. This problem is constrained by a desired mask density d which measures the percentage of mask pixels w.r.t. the number of image pixels.

One strategy for mask optimisation in the homogeneous diffusion case has been proposed by Belhachmi et al. [39]. They show that an optimal mask in the continuous setting can be obtained from the rescaled Laplacian magnitude of the image. However, transferring these results to the discrete setting often suffers from suboptimal dithering strategies. While being highly efficient, reconstruction quality is not fully satisfying. Moreover, such an approach is not available for the other inpainting operators.

Better quality can be achieved with the popular stochastic strategies of Mainberger et al. [246]: Probabilistic sparsification (PS) and nonlocal pixel exchange (NLPE) (see Section 3.6.2).

The use of PS and NLPE requires to solve the inpainting problem numerous times, leading to slow mask optimisation. To avoid this computational bottleneck, we want to reach the quality of stochastic mask optimisation with a more efficient model based on deep learning.

10.2 SPARSE MASKS WITH SURROGATE INPAINTING

Our model consists of two equally shaped U-nets [306] with different loss functions. By optimising both inpainting quality and fidelity to the inpainting equation, we obtain masks with good reconstruction quality for the inpainting problem at hand.

THE MASK NETWORK The *mask network* takes an original image $\boldsymbol{f}$ and transforms it into a mask $\boldsymbol{c}$. We denote the forward pass through the mask network by $\mathcal{M}(\cdot)$, i.e. the mask is computed as $\boldsymbol{c} = \mathcal{M}(\boldsymbol{f})$.

The mask entries lie in the interval $[0, 1]$. Permitting non-binary values allows for a differentiable network model. To obtain mask points in the desired range, we apply a sigmoid function to the output of the network. Moreover, the mask network is trained for a specific mask density d. To this end, we rescale the outputs of the network

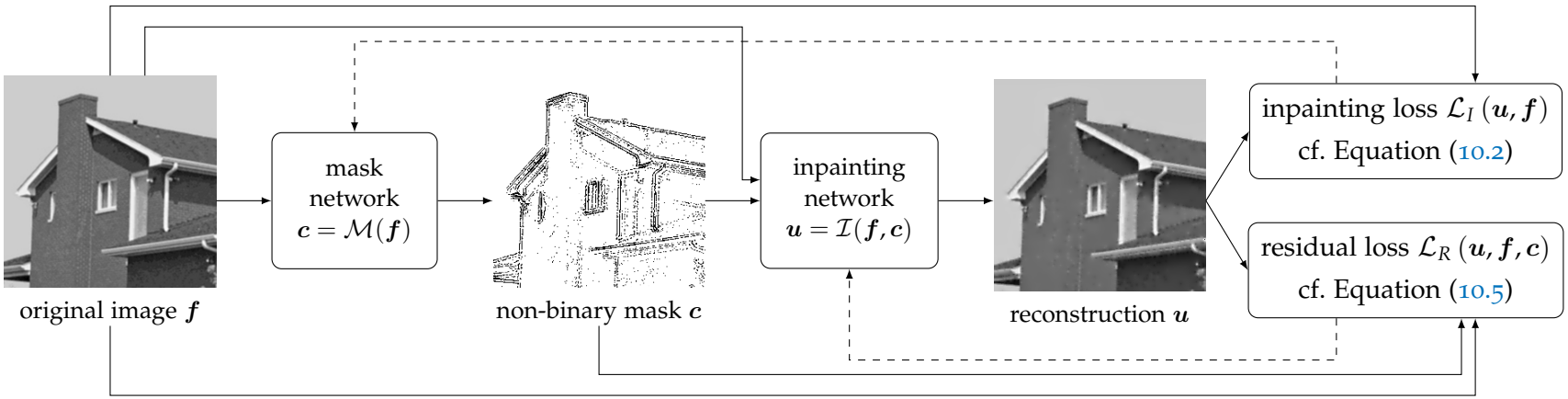


Figure 10.1: Overview of our model structure. Solid lines denote forward passes, dashed lines denote backpropagation.

if they exceed the desired density. We do not require a lower bound, since the loss function incites a sufficiently dense mask.

The mask network places the known data such that the inpainting error between the reconstruction $\boldsymbol{u}$ and the original image $\boldsymbol{f}$ is minimised. This yields the *inpainting loss*

$$\mathcal{L}_I(\boldsymbol{u}, \boldsymbol{f}) = \frac{1}{n_x n_y} \|\boldsymbol{u} - \boldsymbol{f}\|_2^2 \tag{10.2}$$

as its objective function. Its implicit dependency on the inpainting mask links the learned masks to the reconstructions.

We found that the mask network tends to get stuck in local minima with flat masks which are constant at every position, yielding a random sampling. To avoid this, we add a regularisation term $\mathcal{R}(\boldsymbol{c})$ to the inpainting loss $\mathcal{L}_I(\boldsymbol{u}, \boldsymbol{f})$. It penalises the inverse variance of the mask via

$$\mathcal{R}(\boldsymbol{c}) = \frac{\rho_1}{1 + \frac{\sigma_{\boldsymbol{c}}^2}{\rho_2^2}}, \tag{10.3}$$

The variance of a mask describes how strongly the confidence measures of the individual pixels differ from the mean probability. Thus, the regulariser serves two purposes: First, it lifts the bad local minima for flat masks by adding a strong penalty to the energy landscape. Second, it promotes probabilities closer to zero and one, as this maximises the variance. The impact of the regularisation term is steered by two positive regularisation parameters ρ_1, ρ_2.

THE INPAINTING NETWORK The second network is called the *inpainting network*. Its task is to create a reconstruction $\boldsymbol{u}$ which follows a classical inpainting process. In Chapter 7 we have shown that U-nets realise an efficient multigrid strategy at their core. Thus, we use a U-net as a surrogate solver which reproduces the results of the PDE-based inpainting. The inpainting network takes the original image $\boldsymbol{f}$ and the mask $\boldsymbol{c}$ and creates a reconstruction $\boldsymbol{u} = \mathcal{I}(\boldsymbol{f}, \boldsymbol{c})$. This result should solve the discrete version of the inpainting equation (10.1) which reads

$$(\boldsymbol{I} - \boldsymbol{C})\, \boldsymbol{A}(\boldsymbol{u})\boldsymbol{u} - \boldsymbol{C}\,(\boldsymbol{u} - \boldsymbol{f}) = 0. \tag{10.4}$$

Here, $\boldsymbol{A}(\boldsymbol{u})$ is a discrete implementation of the inpainting operator L with reflecting boundary conditions, and $\boldsymbol{C} = \operatorname{diag}(\boldsymbol{c})$ is a matrix representation of the mask. To ensure that the reconstruction $\boldsymbol{u}$ approximates a solution to this equation, we minimise its residual, yielding the *residual loss*

$$\mathcal{L}_R(\boldsymbol{u}, \boldsymbol{f}, \boldsymbol{c}) = \frac{1}{n_x n_y} \| \,(\boldsymbol{I} - \boldsymbol{C})\, \boldsymbol{A}(\boldsymbol{u})\boldsymbol{u} - \boldsymbol{C}\,(\boldsymbol{u} - \boldsymbol{f})\, \|_2^2 \,. \tag{10.5}$$

As the residual loss measures fidelity to the PDE-based process, an optimal network approximates the PDE solution in an efficient way

that allows fast backpropagation. This is another instance of the deep residual idea which we have established in Chapter 7 as an analogy to deep energies [150].

Figure 10.1 presents an overview of the full model structure. Note that the inpainting network receives both the mask and the original image as an input. Thus, this network is not designed for standalone inpainting. However, this allows the network to easily minimise the residual loss by transforming the original into an accurate inpainting result, given the mask as side information.

PRACTICAL APPLICATION After training the full pipeline in a joint fashion, the mask network can be used to generate masks for diffusion inpainting. To this end, we apply the mask network to an original image and obtain a non-binary mask. This mask is then binarised: The probability of a pixel belonging to a mask is given by its non-binary value. At each position, we perform a weighted coin flip with that probability. Afterwards, the binarised masks are fed into a numerical solver of choice for the inpainting problem.

While binarising the mask is not necessary in this pure inpainting framework, it is important for compression applications since storing binary masks with arbitrary point distributions is already highly non-trivial [259].

10.3 EXPERIMENTS

EXPERIMENTAL SETUP We train both U-nets jointly with their respective loss function on the BSDS500 dataset [20]. As a training set, we use 200 cropped grey value images of size 256×256 with values in the range $[0, 255]$. We do not use a validation set as the training process is fully fixed.

Both U-nets employ 5 scales, with 3 layers per scale. On the finest scale, they use 10 channels, and this number is doubled on each scale. Thus, each U-net possesses around 9×10^5 parameters. We use the Adam optimiser [212] with standard settings, a learning rate of $5 \cdot 10^{-4}$, and 4000 epochs. We train multiple instances of the model for densities between 10% and 1% with several random initialisations.

The regularisation parameter ρ_1 scales linearly with the density, ranging from 10 to 25 for densities between 1% and 10%. This ensures that the balance between regularisation term and loss is similar for all densities. The parameter ρ_2 is fixed to 0.1.

EED inpainting requires to provide the model parameters λ and σ. However, we found that learning the parameters during the inpainting is not trivial and suffers from easy minima in the sense that the model chooses parameter combinations which yield e.g. homogeneous diffusion inpainting. Thus, we settle for surrogate parameters by fixing $\lambda = 1$ and choosing $\sigma = \frac{0.2}{d}$ adaptively with the mask density. We

also set these parameters for the probabilistic strategies for a fair comparison. While this does not produce optimal inpainting quality, it allows to show that it is indeed possible to learn masks for EED inpainting.

After training, we binarise the masks and use them with a conjugate gradient solver for the inpainting problem to obtain a reconstruction. Since we aim at the highest quality, we take the best result out of 30 samplings.

Analogously, we generate masks once with PS only, and once with PS and additional NLPE. In the following, we denote the latter combination by PS+NLPE. In our sparsification we use candidate fractions $p = 0.1$ and $q = 0.05$, and we take the best result out of 5 runs. For NLPE, we use 30 candidates of which 10 are exchanged. We run NLPE for 10 cycles. In a single cycle, each mask point is exchanged once on average.

Moreover, we compare our homogeneous diffusion results against the strategy of Belhachmi et al. [39]. This approach is realised by taking the Laplacian magnitude of the image, rescaling it to obtain a desired density, and dithering the result with a binary Floyd–Steinberg algorithm [126].

We compare our results on the five popular test images *boat*, *house*, *peppers* [342], *cameraman*, and *trui* [402] since performing PS and NLPE on a large database is infeasible. We measure the quality in terms of peak signal-to-noise ratio (PSNR).

RECONSTRUCTION QUALITY Figure 10.2 shows a visual comparison of optimised masks and the corresponding inpainting results for the case of homogeneous diffusion inpainting. For all test cases, we observe that our learned masks are structurally similar to those obtained by PS with NLPE. This helps to create sharper contours, whereas the inpainting results of Belhachmi et al. suffer from fuzzy edges. The visual quality of the inpainting results for our model and PS+NLPE is indeed competitive.

Figure 10.3 presents the results for biharmonic diffusion inpainting. As no analytic method is available, we substitute the results by those obtained by probabilistic sparsification only. The PS+NLPE masks distribute points more evenly than those for homogeneous inpainting. Interestingly, the difference between PS only and PS+NLPE can become quite drastic, in particular for very sparse masks. As biharmonic inpainting can produce over- and undershoots, large areas without mask points can lead to heavy degradations.

We observed that learning masks for biharmonic diffusion is far more difficult than for homogeneous diffusion inpainting. We conjecture that the higher sensitivity of the operator w.r.t. individual mask points complicates the network optimisation. The resulting masks for high densities cluster more than the competing results, and the

inpainting consequently suffers from the over- and undershoots. Interestingly, the extremely sparse mask in Figure 10.3a is quasi-random. We presume that this is a local minimum which was easier to attain than that of PS+NLPE.

For EED inpainting, we made similar observations as for biharmonic diffusion. The model is harder to train and gets stuck in bad local minima more often. Figure 10.4 shows that the learned masks are competitive in the extremely sparse case, but cannot compete with neither PS nor PS+NLPE in the cases with higher density. Nevertheless, the coarse structure of the mask is similar. These results show that our methodology is applicable to more complex inpainting operators, however, some optimisations in the specific architecture and its training are warranted.

Figure 10.5 presents a comparison of the reconstruction quality averaged over the test images for all inpainting operators. For homogeneous diffusion, our learned masks consequently outperform the strategy of Belhachmi et al.. Moreover, our model is on par with PS for densities smaller than 6%. For some extremely small densities it even outperforms PS and is on par with PS+NLPE.

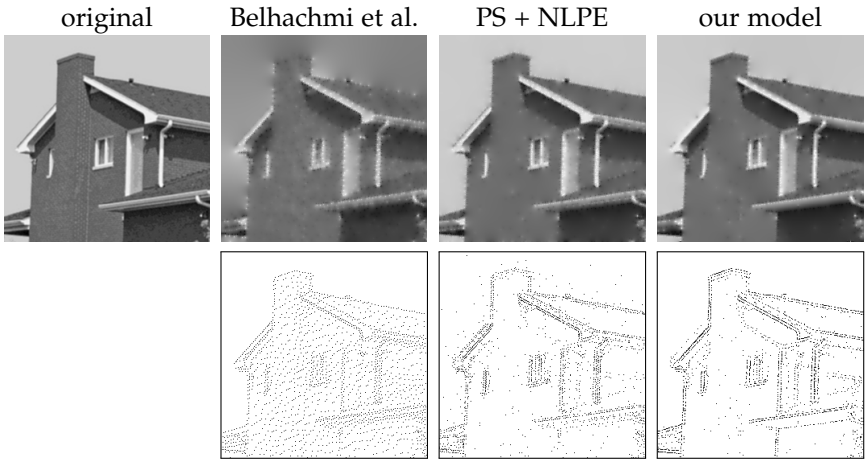
In the biharmonic diffusion case, our learned model is competitive with PS. The nonmonotone graph results from failure cases on individual images. Interestingly the trend for very sparse masks is the same as for homogeneous diffusion, albeit not that pronounced. The quality of PS+NLPE is out of reach.

This is even more the case for EED inpainting. Here, we are only competitive with PS for very sparse masks, while our model quickly falls of in comparison to PS and PS+NLPE for higher densities. It seems that our model does not account for the specific scaling required for EED.

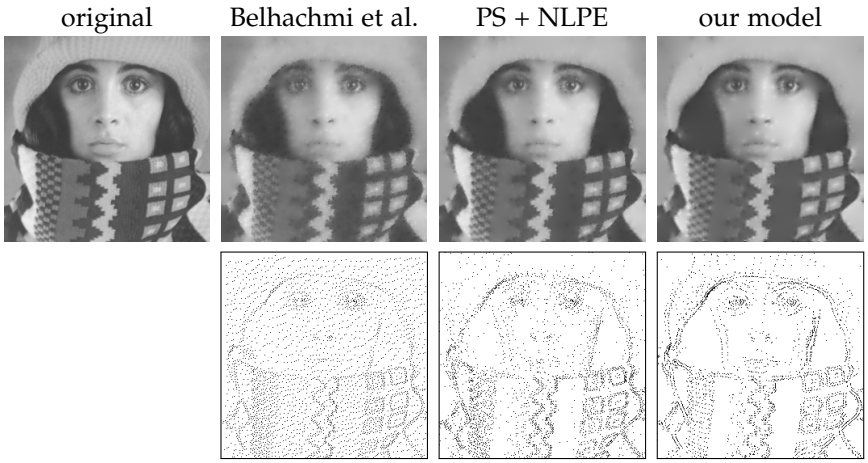
These results are still promising: For applications such as inpainting-based image compression, very sparse masks are more important and more challenging [246, 327]. Therefore, our mask generation model performs well for the practically relevant mask densities.

COMPUTATIONAL EFFICIENCY The big advantage of the learned mask generation is its speed. As inpainting operations are the dominant factor for computation time, we use the number of inpaintings as a measure for efficiency. In comparison, the forward pass of the mask network is negligible.

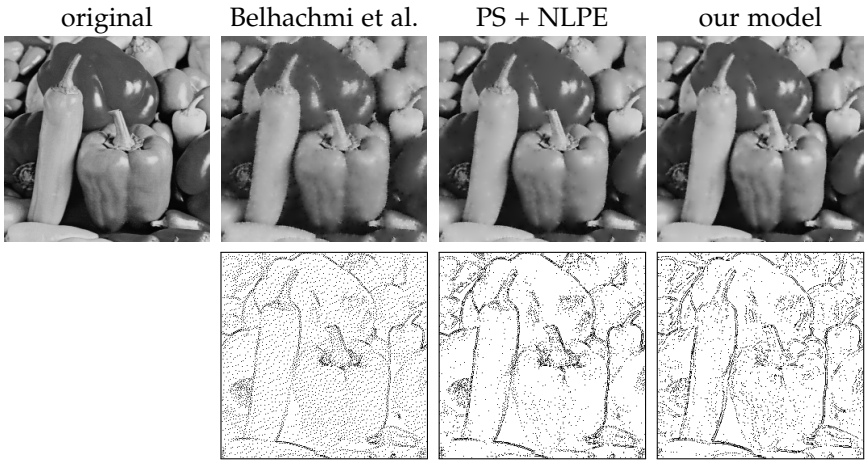
Figure 10.6 visualises the average number of inpaintings required to obtain masks of a specific density for the test set. For this comparison we choose homogeneous diffusion inpainting, however the number of inpaintings is independent of the operator at hand. To generate a mask, both our model and that of Belhachmi et al. do not require any inpainting operations. Thus, the efficiency of these mask generation strategies does not depend on the density.



(a) homogeneous diffusion on *house* with 3% density

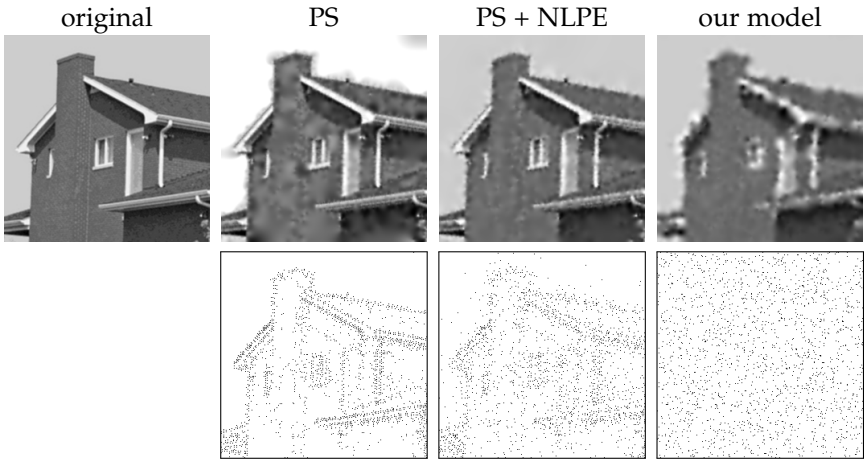


(b) homogeneous diffusion on *trui* with 5% density

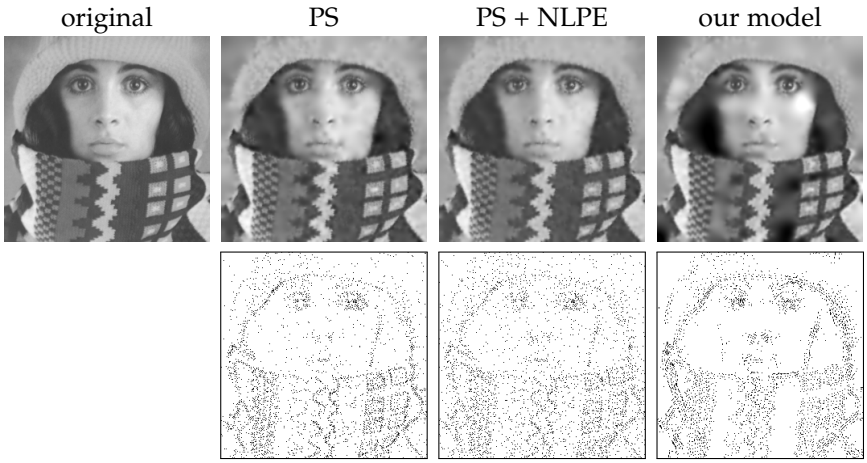


(c) homogeneous diffusion on *peppers* with 8% density

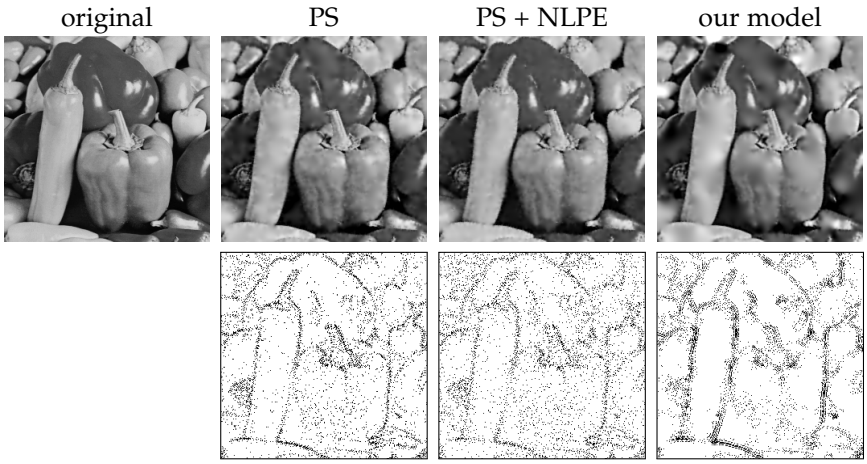
Figure 10.2: Homogeneous inpainting results for different mask densities. Mask points are shown in black, and mask images are framed for better visibility.



(a) biharmonic diffusion on *house* with 3% density

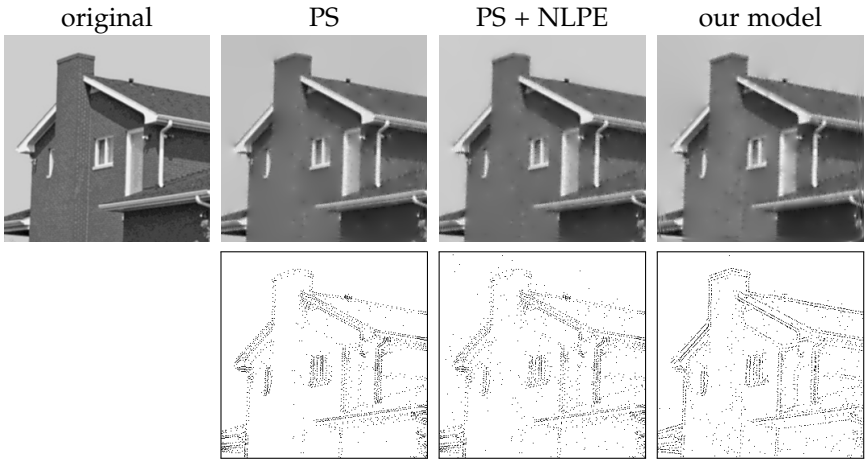


(b) biharmonic diffusion on *trui* with 5% density

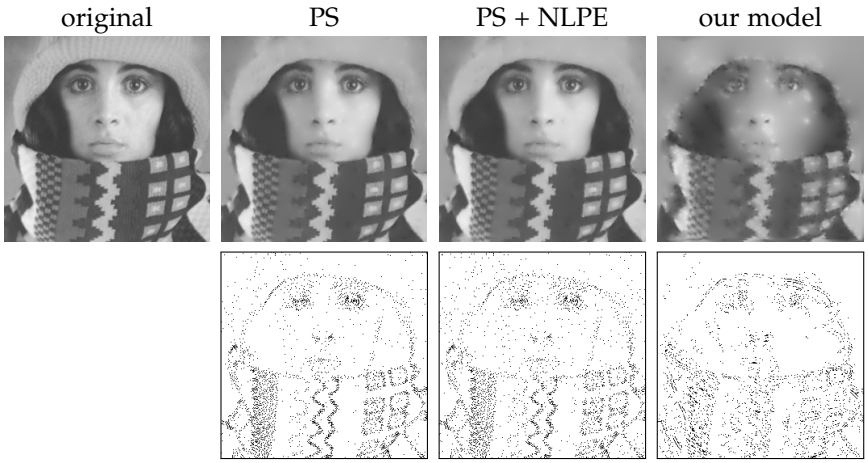


(c) biharmonic diffusion on *peppers* with 8% density

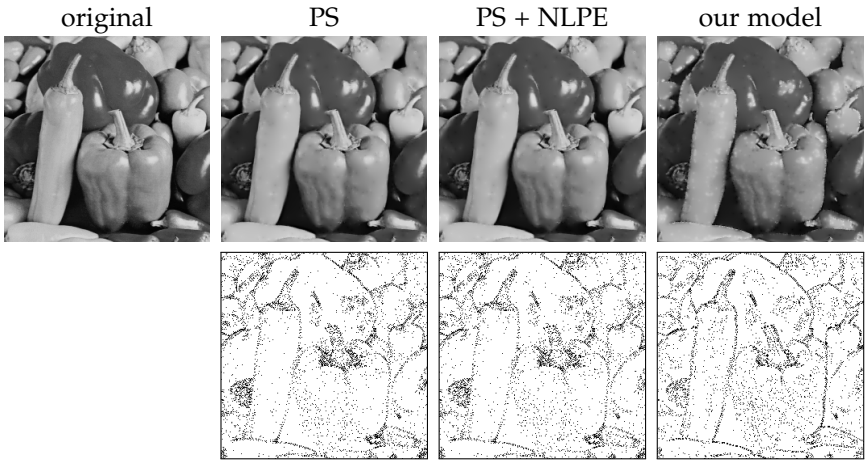
Figure 10.3: Biharmonic inpainting results for different mask densities. Mask points are shown in black, and mask images are framed for better visibility.



(a) EED on *house* with 3% density



(b) EED on *trui* with 5% density



(c) EED on *peppers* with 8% density

Figure 10.4: EED inpainting results for different mask densities. Mask points are shown in black, and mask images are framed for better visibility.

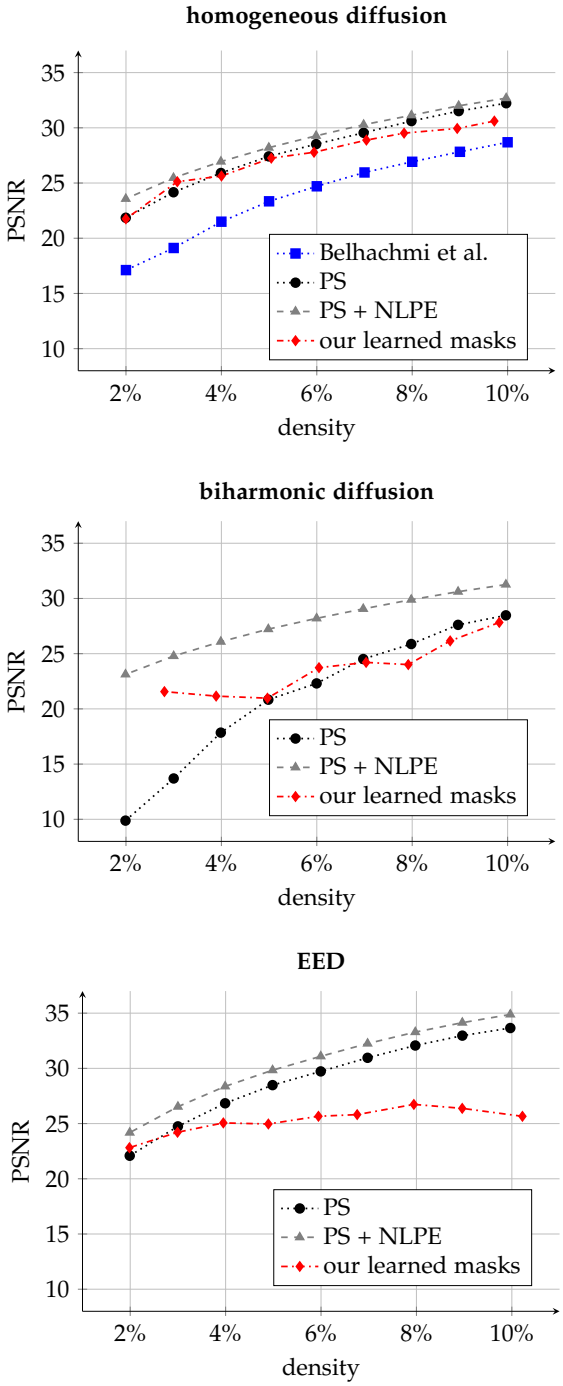


Figure 10.5: Comparison of average inpainting quality on the test images for three inpainting operators. For homogeneous diffusion, he learned masks consistently outperform those of Belhachmi et al. and can compete with masks generated by PS. For very sparse masks, the learned masks for homogeneous diffusion can compete with PS+NLPE. For biharmonic inpainting, this trend continues, however the margin between probabilistic strategies and our learned model increases. For EED, only very sparse learned masks are competitive.

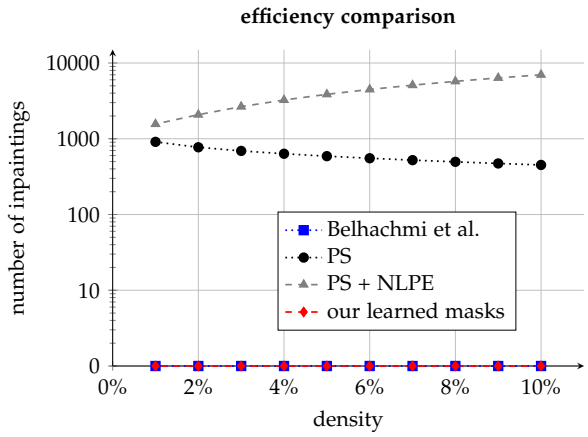


Figure 10.6: Comparison of efficiency in terms of the number of inpaintings for each density. Both our method and that of Belhachmi et al. generate masks without computing an inpainting. The stochastic optimisation strategies compute up to thousands of inpaintings.

For PS, lower densities require more inpainting operations. Adding NLPE requires even more inpainting operations depending on the number of cycles and the mask density. Both strategies trade computational efficiency for inpainting quality.

For example, a single sparsification run with homogeneous diffusion for a 3% mask on the *cameraman* image with realistic parameter settings requires 700 steps. On an *Intel Core i7-7700K CPU @ 4.20GHz*, this amounts to 58 seconds of runtime. The subsequent NLPE optimisation requires another 2000 steps, resulting in more than 3 minutes of additional runtime. For the more complex operators of biharmonic diffusion and EED this process can be orders of magnitude slower. In contrast, the strategy of Belhachmi et al. does not require any inpainting, and a mask can be generated in only 24 milliseconds.

Our model requires only 85 milliseconds for passing a single image through the mask network on the CPU. Thus, it plays in the same league as the strategy of Belhachmi et al., while being on par with the stochastic optimisation in terms of quality. This allows instantaneous high quality mask generation. As a consequence, our learned model can serve as a highly efficient replacement of stochastic mask optimisation in the case of homogeneous diffusion.

Moreover, our model can produce masks instantaneously for all inpainting operators, while the analytic strategy of Belhachmi et al. is only available for homogeneous diffusion inpainting. For biharmonic diffusion and EED inpainting, a combination of instantaneous mask generation with additional NLPE may be a good compromise to accelerate the mask optimisation process.

10.4 CONCLUSIONS

We have proposed the first approach of sparse mask learning for diffusion-based inpainting. It fuses ideas from deep learning with classical homogeneous diffusion inpainting. The key of this strategy is a combination of an inpainting loss for the mask generator and a residual loss for the surrogate inpainting network. Its results are competitive with stochastic mask optimisation for homogeneous diffusion inpainting, while being up to four orders of magnitude faster. This constitutes a new milestone in mask optimisation for diffusion-based inpainting.

Moreover, our model relies on our perspective to treat neural networks as numerical solvers. The use of PDE residuals for solving the inpainting equation with a surrogate model is the key to circumventing backpropagation through a classical numerical solver, yielding an efficient learning strategy. Thus, we see that our theoretical considerations from Part II are also of practical relevance.

Improving the model architecture and its hyperparameters is essential for stabilising the optimisation process, and it may lead to better results for the more complex inpainting operators of biharmonic diffusion and EED. Moreover, our methodology has the potential to optimise every part of the inpainting equation while simultaneously solving it. This includes learning the EED parameters as well as providing a tonally optimised version of the input image, given that the optimisation process is still well-behaved.

11 CONCLUSIONS AND OUTLOOK

11.1 CONCLUSIONS

We have shown that connecting and combining mathematically well-founded models and concepts from deep learning can yield both valuable theoretical insights as well as practical improvements. We were able to identify design criteria for neural network components that fulfil various mathematical guarantees, and with their help create advanced models for practical applications such as denoising and inpainting.

In the first part of the thesis, we have improved models relying on wavelet shrinkage and anisotropic diffusion by means of learning with additional model reduction. Chapter 4 presented an approach to design scale-adaptive wavelet shrinkage functions with the help of their connection to diffusion filters. Therein, the amount of shrinkage on each scale was defined by a trainable parameter set. We have identified a smooth relation between the parameters and were able to drastically reduce the parameter set to a practically feasible minimal set, without fundamentally sacrificing denoising performance.

In the same way, Chapter 5 extended this approach to anisotropic diffusion filters. Relying on integrodifferential equations, we were able to come up with integrodifferential anisotropic diffusion as an extension of edge-enhancing diffusion. By accumulating diffusion processes and structural information over multiple image scales, the proposed model can denoise images more effectively. The methodology remained unchanged: A small trainable set of parameters helps to adapt the model to the new configuration, while a manually inferred reduction leaves us with a minimal parameter set. This model reduction strategy is of general nature and can be applied in a straightforward way.

The second part is the heart of this thesis. Therein, we have extensively investigated the connections between diffusion, wavelets, variational methods, and specific neural network architectures. While Chapter 6 started with simple one-dimensional models with no learning involved, Chapter 7 extended our investigations to more general, trainable diffusion filters. Finally, Chapter 8 considered the two-dimensional setting and the newly arising concepts such as rotation invariance. Our core finding was that by regarding these architectures as numerical solvers, one can define criteria for network building blocks which are stable, well-posed, and rotationally invariant by design. These criteria include a symmetric filter structure and activation functions which coincide with diffusion flux functions that

can couple multiple network channels. Moreover, we have identified several popular numerical algorithms for PDEs with their network counterparts.

Our decision of translating concepts from the traditional mathematical models towards the world of neural networks was deliberate. It is motivated from the fact that the opposite way of translation is much more challenging, if not infeasible. Thus, our findings are predominantly to be understood as one-way connections. Nevertheless, they allow some conjectures in the other direction. While it is obvious that fully fledged neural networks perform very complex computations, our findings suggest that at their core, they are founded on the same basic concepts as their numerical counterparts. The many parallels that can be observed in the histories of both neural network design and numerical algorithms for PDEs suggests that our interpretations are plausible.

The final part of the thesis dealt with improvements for image inpainting and inpainting-based image compression by means of hybrid models. The first hybrid model in Chapter 9 combined Shepard interpolation, an efficient image inpainting strategy, with concepts from edge-enhancing diffusion inpainting. This allowed us to come up with anisotropic Shepard interpolation which offers a good compromise between computational efficiency and quality of inpainting results. We have shown that this model is able to significantly advance the compression performance of previous codecs relying on Shepard interpolation on the way towards beating JPEG with simple codecs.

Finally, Chapter 10 returned to our primary focus by combining diffusion-based inpainting and deep learning. By using PDE residuals within neural loss functions, we solved an inpainting problem with a surrogate network, while another one learned the corresponding inpainting masks. By attributing different losses to the networks, it was possible to efficiently learn a mask generating network for diffusion-based inpainting. This is a stark improvement over existing data optimisation strategies as our strategy does not require to perform any inpainting once the mask generator is fully trained. The conclusion in this chapter is of more general nature: When using neural networks in their full form and as a black box, then model-driven loss functions are an elegant way to control and predict the behaviour of the neural network. Prescribing a mathematical model as a loss is fully model-driven and reduces the neural network to a surrogate solver, and thus optimally decouples it from the model.

We have provided three strategies for bridging the gap between model- and data-driven approaches from different viewpoints: mathematical modelling, numerical analysis, and practical applications. We hope that our findings pave the way towards a better understanding of neural networks, and to a unification of mathematical models and deep learning.

11.2 OUTLOOK

The model reduction strategies that we have presented in Part I are of general nature, but also remain hand-crafted: We have inspected the trainable parameters manually and came up with suitable relations that compromise approximation quality and physical plausibility. Automating this process would provide a general framework that can be applied to various similar models. All that is required is to decide on one or multiple modelling dimensions such as scale or time, equip them with a discrete set of parameters and a smoothness condition, and finally train and automatically reduce the resulting parameter set.

Moreover, this would ease the application of e.g. the integrodifferential diffusion model for inpainting. As we have seen in our extensions of Chapter 5, without further adaptations this model is not yet outperforming edge-enhancing diffusion for inpainting.

Also our findings in Part II are not the end of the road. In the same systematic fashion one can analyse further numerical algorithms and their neural counterparts. This can only be beneficial: If a corresponding architecture exists one can gain insights from the newly found links, and if no such architecture is available it opens the door for coming up with new designs with potentially advantageous properties. In any case, enriching the dictionary between PDEs and CNNs will strengthen the theoretical understanding of the latter and help to find compromises between mathematical guarantees and performance.

In addition, our experimental evaluations in this part are simplistic for the sake of gaining fundamental insights. Nevertheless, they provide a solid foundation for extending our findings to more complex networks and applications that are more typical for neural networks. We think that an essential question in this context is how one should choose the balance between stability and performance. It is likely that this choice strongly depends on the task at hand and the modelling goals, yet it should be thoroughly considered.

Finally, the empirical success of using PDE residuals in loss functions warrants a closer look. In combination with deep energies, they allow to train networks to efficiently yield approximative solutions to mathematical models. It alleviates the need for data, and allows to use the network as a black box while leaving the mathematical model untouched. Strengthening the understanding of this concept and evaluating it in further practical settings promises to yield valuable insights into the synergy between neural networks and mathematical models.

This ties into our visions for the future: At some point, fully understanding the power of neural networks will be inevitable for further scientific progress, as one cannot and should not only rely on more data and faster hardware. This thesis provides one of many pieces in the big puzzle of this task from the viewpoint of mathematical models for image processing.

A

ROTATIONALLY INVARIANT WAVELET SHRINKAGE

In Chapter 4 we have used the rotationally invariant shrinkage rule of Mrázek and Weickert [263] (cf. Equation (4.13)) which reads

$$\boldsymbol{S}\left(\begin{pmatrix} w_x \\ w_y \\ w_{xy} \end{pmatrix}\right) = \left(1 - 4\tau g\left(w_x^2 + w_y^2 + 2\,w_{xy}^2\right)\right) \begin{pmatrix} w_x \\ w_y \\ w_{xy} \end{pmatrix}. \tag{A.1}$$

It connects the shrinkage to a diffusion process with time step size τ and diffusivity g. The three directional wavelet coefficients w_x, w_y, w_{xy} are coupled within the diffusivity argument to ensure a rotationally invariant process.

In the original paper [263], this rule contains two free parameters c and q and can be expressed as

$$\boldsymbol{S}\left(\begin{pmatrix} w_x \\ w_y \\ w_{xy} \end{pmatrix}\right) = \begin{pmatrix} w_x \\ w_y \\ w_{xy} \end{pmatrix} - 4\tau g\left(w_x^2 + w_y^2 + c\,w_{xy}^2\right) \begin{pmatrix} w_x \\ w_y \\ q\,w_{xy} \end{pmatrix}. \tag{A.2}$$

The parameter $c \in [0, 2]$ steers the contribution of the diagonal wavelet shrinkage coefficients w_{xy} to the diffusivity argument, and the choice of $q \in [0, 1]$ steers the contribution of axial and diagonal discretisations of the diffusion process. As a best practice choice, Mrázek and Weickert recommend to choose $q = \frac{1}{2}$ and $c = 2$.

However, the existing connections between wavelet shrinkage and diffusion [263], our considerations on flux functions, and the numerical advances of Weickert et al. [379] raise the question whether above shrinkage rule could employ only a single free parameter.

We claim that a shrinkage rule with a single free parameter γ

$$\boldsymbol{S}\left(\begin{pmatrix} w_x \\ w_y \\ w_{xy} \end{pmatrix}\right) = \begin{pmatrix} w_x \\ w_y \\ w_{xy} \end{pmatrix} - 4\tau g\left(w_x^2 + w_y^2 + \gamma\,w_{xy}^2\right) \begin{pmatrix} w_x \\ w_y \\ \sqrt{\gamma}\,w_{xy} \end{pmatrix} \tag{A.3}$$

together with a shift-invariant Haar wavelet shrinkage on the finest scale is equivalent to a nonlinear isotropic diffusion process which is discretised with the approach of Weickert et al. [379] with a free parameter $\alpha = \frac{2-\gamma}{4}$.

In our proof we closely follow the argumentation of Didas et al. [99]. We consider the shift-invariant Haar wavelet shrinkage as outlined in Chapter 4. A discrete image $\boldsymbol{f}$ can be perfectly reconstructed by

simply transforming it into the wavelet basis and back again without performing any shrinkage:

$$\boldsymbol{f} = \frac{1}{4^L}\boldsymbol{Q}^{(L)\top}\boldsymbol{Q}^{(L)}\boldsymbol{f} + \sum_{\ell=1}^{L}\frac{1}{4^\ell}\boldsymbol{Q}_H^{(\ell)\top}\boldsymbol{Q}_H^{(\ell)}\boldsymbol{f}. \tag{A.4}$$

Here, we have abbreviated the highpass filters $\boldsymbol{Q}_x^{(\ell)}, \boldsymbol{Q}_y^{(\ell)}$, and $\boldsymbol{Q}_{xy}^{(\ell)}$ by stacking them into a single highpass matrix $\boldsymbol{Q}_H^{(\ell)}$. The matrices are defined as discussed in Chapter 4. The rescaling of $\frac{1}{4^\ell}$ arises from the shift-invariant wavelet transformation.

If we apply scale-dependent shrinkage functions $\boldsymbol{S}^{(\ell)}$ to the wavelet coefficients, we obtain a modified image $\boldsymbol{u}$:

$$\boldsymbol{u} = \frac{1}{4^L}\boldsymbol{Q}^{(L)\top}\boldsymbol{Q}^{(L)}\boldsymbol{f} + \sum_{\ell=1}^{L}\frac{1}{4^\ell}\boldsymbol{Q}_H^{(\ell)\top}\boldsymbol{S}^{(\ell)}\left(\boldsymbol{Q}_H^{(\ell)}\boldsymbol{f}\right). \tag{A.5}$$

Taking the difference of the modified image (A.5) and the perfect reconstruction (A.4) yields

$$\boldsymbol{u} - \boldsymbol{f} = \sum_{\ell=1}^{L}\frac{1}{4^\ell}\boldsymbol{Q}_H^{(\ell)\top}\left(\boldsymbol{S}^{(\ell)}\left(\boldsymbol{Q}_H^{(\ell)}\boldsymbol{f}\right) - \boldsymbol{Q}_H^{(\ell)}\boldsymbol{f}\right) \tag{A.6}$$

Interpreting the input image $\boldsymbol{f}$ as an evolving image at time step k and the modified image $\boldsymbol{u}$ as an evolving image at time step $k+1$ gives

$$\boldsymbol{u}^{k+1} - \boldsymbol{u}^k = \sum_{\ell=1}^{L}\frac{1}{4^\ell}\boldsymbol{Q}_H^{(\ell)\top}\left(\boldsymbol{S}^{(\ell)}\left(\boldsymbol{Q}_H^{(\ell)}\boldsymbol{u}^k\right) - \boldsymbol{Q}_H^{(\ell)}\boldsymbol{u}^k\right). \tag{A.7}$$

If we now plug in our proposed shrinkage rule (A.3) into this expression, we obtain an explicit scheme of a nonlinear isotropic diffusion process

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} = -\sum_{\ell=1}^{L}\frac{1}{4^{\ell-1}}\boldsymbol{Q}_H^{(\ell)\top}\left(g\left(\left|\sqrt{\boldsymbol{\Gamma}}\,\boldsymbol{Q}_H^{(\ell)}\boldsymbol{u}^k\right|^2\right)\sqrt{\boldsymbol{\Gamma}}\,\boldsymbol{Q}_H^{(\ell)}\boldsymbol{u}^k\right), \tag{A.8}$$

where $\sqrt{\boldsymbol{\Gamma}}$ denotes a rescaling of only the diagonal wavelet coefficients with the root of the free parameter $\sqrt{\gamma}$. The squared magnitude within the diffusivity argument is to be interpreted position-wise, i.e.

$$\left(\left|\sqrt{\boldsymbol{\Gamma}}\,\boldsymbol{Q}_H^{(\ell)}\boldsymbol{u}^k\right|^2\right)_{i,j} = \left(w_x^{(\ell)}\right)_{i,j}^2 + \left(w_y^{(\ell)}\right)_{i,j}^2 + \gamma\left(w_{xy}^{(\ell)}\right)_{i,j}^2. \tag{A.9}$$

In light of our findings in Part II of this thesis, expression (A.8) also fits the bigger picture better than the original shrinkage rule of Mrázek and Weickert [263]. Rewriting the bracketed term as a flux

$$\boldsymbol{\Phi}\left(\sqrt{\boldsymbol{\Gamma}}\,\boldsymbol{Q}_H^{(\ell)}\boldsymbol{u}^k\right) = g\left(\left|\sqrt{\boldsymbol{\Gamma}}\,\boldsymbol{Q}_H^{(\ell)}\boldsymbol{u}^k\right|^2\right)\sqrt{\boldsymbol{\Gamma}}\,\boldsymbol{Q}_H^{(\ell)}\boldsymbol{u}^k \tag{A.10}$$

shows that from the viewpoint of a continuous flux function, the argument of the diffusivity and the term that it is multiplied with

should be exactly the same. While this is not mandatory, it helps to easily reduce the amount of free parameters.

Let us now analyse the diffusion process only on the finest scale $\ell = 1$. We need to show two things: First, the argument of the diffusivity function must be the same approximation as obtained with the approximation of Weickert et al. [379]. Secondly, the full discretisation of the right hand side of (A.8) must yield the isotropic stencil for the divergence term of the diffusion process according to [379].

Let us first inspect the diffusivity argument on the finest scale $\ell = 1$. It is easy to check that at a fixed off-grid position $(i + \frac{1}{2}, j + \frac{1}{2})$ we have

$$(w_x)^2_{i+\frac{1}{2},j+\frac{1}{2}} = \left(\frac{1}{2}\left(\boxed{}\!\!- + \boxed{}\!\!- \right)\right)^2, \tag{A.11}$$

$$(w_y)^2_{i+\frac{1}{2},j+\frac{1}{2}} = \left(\frac{1}{2}\left(\boxed{}\!\!| + \boxed{}\!\!| \right)\right)^2, \tag{A.12}$$

$$\begin{aligned}(w_{xy})^2_{i+\frac{1}{2},j+\frac{1}{2}} &= \left(\frac{1}{2}\left(\boxed{}\!\!- - \boxed{}\!\!- \right)\right)^2 \\ &= \left(\frac{1}{2}\left(\boxed{}\!\!| - \boxed{}\!\!| \right)\right)^2 \\ &= \frac{1}{2}\left(\left(\frac{1}{2}\left(\boxed{}\!\!- - \boxed{}\!\!- \right)\right)^2 + \left(\frac{1}{2}\left(\boxed{}\!\!| - \boxed{}\!\!| \right)\right)^2\right).\end{aligned} \tag{A.13}$$

Here we make use of the abbreviations defined in Section 3.1.2. Moreover, we omit the superscript denoting the scale for the sake of readability.

After tedious but straightforward calculations, one can show that the diffusivity argument equals

$$\begin{aligned}&\left((w_x)^2 + (w_y)^2 + \gamma\ (w_{xy})^2 \right)_{i+\frac{1}{2},j+\frac{1}{2}} \\ &\quad= \frac{2+\gamma}{8}\left(\boxed{}\!\!-^2 + \boxed{}\!\!-^2 \right) + \frac{2-\gamma}{4}\ \boxed{}\!\!- \cdot \boxed{}\!\!- \\ &\quad\quad+ \frac{2+\gamma}{8}\left(\boxed{}\!\!|^2 + \boxed{}\!\!|^2 \right) + \frac{2-\gamma}{4}\ \boxed{}\!\!| \cdot \boxed{}\!\!| .\end{aligned} \tag{A.14}$$

With our proposed choice $\alpha = \frac{2-\gamma}{4}$, this is the exact same gradient approximation as in the diffusion case (cf. Equations (3.48) and (3.49), as well as [379]).

Moreover, in a similar manner, the right hand side of (A.8) for a pixel position i, j can be condensed into the stencil

$$\frac{1}{2h^2}\begin{array}{|c|c|c|}\hline (1-\frac{\gamma}{2})\,g_{-+} & \frac{\gamma}{2}\,(g_{-+}+g_{++}) & (1-\frac{\gamma}{2})\,g_{++} \\ \hline \frac{\gamma}{2}\,(g_{--}+g_{-+}) & -(1+\frac{\gamma}{2})\left(g_{--}+g_{+-}+g_{-+}+g_{++}\right) & \frac{\gamma}{2}\,(g_{+-}+g_{++}) \\ \hline (1-\frac{\gamma}{2})\,g_{--} & \frac{\gamma}{2}\,(g_{--}+g_{+-}) & (1-\frac{\gamma}{2})\,g_{+-} \\ \hline\end{array}. \tag{A.15}$$

Here, we have used shorthand notations such as $g_{++} = g_{i+\frac{1}{2},j+\frac{1}{2}}$ for the sake of readability.

With the relation $\alpha = \frac{2-\gamma}{4}$, this is indeed the same stencil as in the isotropic diffusion case (cf. (3.51) and [379]).

Thus, we conclude that wavelet shrinkage on a single scale and nonlinear isotropic diffusion are equivalent, if shrinkage function and diffusivity are chosen according to (A.3).

This provides a one-to-one mapping between the rotationally invariant wavelet shrinkage discretisation of Mrázek and Weickert [263] and that of Weickert et al. [379] for diffusion. Nevertheless, we do not claim that the shrinkage rule with two independent parameters cannot provide better rotation invariance properties in practical examples. In that case, it may be worthwhile to introduce an equivalent, second parameter in the diffusion discretisation of Weickert et al. [379]. However, this comes at the cost of losing the energy-based derivation as the diffusivity argument and the divergence term discretisation employ different discretisations in that case.

B

STABILITY OF DU FORT–FRANKEL SCHEMES

In this section we extend the stability proof for linear Du Fort–Frankel schemes of [156] to the nonlinear setting. The following proof can be found in [11] and has been contributed by Dr. Matthias Augustin. We present it for the sake of completeness.

First we rewrite the scheme (7.17) as a multi-step method, obtaining

$$\begin{pmatrix} \boldsymbol{u}^{k+1} \\ \boldsymbol{u}^{k} \end{pmatrix} = \begin{pmatrix} \frac{4\tau\alpha}{1+2\tau\alpha}\boldsymbol{I} - \frac{2\tau}{1+2\tau\alpha}\boldsymbol{A}\left(\boldsymbol{u}^{k}\right) & \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{k} \\ \boldsymbol{u}^{k-1} \end{pmatrix}. \tag{B.1}$$

Here, we have abbreviated $\boldsymbol{A}(\boldsymbol{u}^k) = \boldsymbol{K}^\top \boldsymbol{G}(\boldsymbol{u}^k)\boldsymbol{K}$.

To analyse the stability of the Du Fort–Frankel scheme, we have to show that all eigenvalues of the matrix of the multistep method (B.1) have an absolute value less than or equal to one. Note that this matrix is not symmetric such that it might have complex eigenvalues.

Let us first define as short-hand notations

$$\begin{aligned} \boldsymbol{Q} &:= \frac{4\tau\alpha}{1+2\tau\alpha}\boldsymbol{I} - \frac{2\tau}{1+2\tau\alpha}\boldsymbol{A}\left(\boldsymbol{u}^{k}\right), \\ \boldsymbol{B} &:= \begin{pmatrix} \boldsymbol{Q} & \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0} \end{pmatrix}. \end{aligned} \tag{B.2}$$

We start with the naive approach to compute eigenvalues of $\boldsymbol{B}$: $\nu \in \mathbb{C}$ is an eigenvalue of $\boldsymbol{B}$ if

$$\det(\boldsymbol{B} - \nu\boldsymbol{I}) = \det\left(\begin{pmatrix} \boldsymbol{Q} - \nu\boldsymbol{I} & \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{I} \\ \boldsymbol{I} & -\nu\boldsymbol{I} \end{pmatrix}\right) = 0. \tag{B.3}$$

As $\boldsymbol{B} - \nu\boldsymbol{I}$ is a block matrix containing square blocks of the same shape where the lower two blocks commute, we have

$$\begin{aligned} \det(\boldsymbol{B} - \nu\boldsymbol{I}) &= \det\left((\boldsymbol{Q} - \nu\boldsymbol{I})(-\nu\boldsymbol{I}) - \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{I}\right) \\ &= \det\left(\nu^2\boldsymbol{I} - \nu\boldsymbol{Q} - \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{I}\right). \end{aligned} \tag{B.4}$$

To proceed from here, it is reasonable to involve the eigenvalues of the matrix $\boldsymbol{Q}$. As $\boldsymbol{Q}$ is real-valued and symmetric, there exist an orthogonal matrix $\boldsymbol{V}$ of eigenvectors of $\boldsymbol{Q}$ and a diagonal matrix $\boldsymbol{\Gamma}$ with the eigenvalues γ of $\boldsymbol{Q}$ on its diagonal such that

$$\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{\Gamma}\boldsymbol{V}^T. \tag{B.5}$$

As $\boldsymbol{V}$ is an orthogonal matrix, it holds that

$$\boldsymbol{I} = \boldsymbol{V}\boldsymbol{V}^T. \tag{B.6}$$

Plugging both of these relations into (B.3) yields

$$\begin{aligned}
\det(\boldsymbol{B} - \nu\boldsymbol{I}) &= \det\left(\nu^2\boldsymbol{V}\boldsymbol{V}^T - \nu\boldsymbol{V}\boldsymbol{\Gamma}\boldsymbol{V}^T - \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{V}\boldsymbol{V}^T\right) \\
&= \det\left(\boldsymbol{V}\left(\nu^2\boldsymbol{I} - \nu\boldsymbol{\Gamma} - \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{I}\right)\boldsymbol{V}^T\right) \\
&= \det(\boldsymbol{V})\det\left(\nu^2\boldsymbol{I} - \nu\boldsymbol{\Gamma} - \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{I}\right)\det\left(\boldsymbol{V}^T\right) \\
&= \det\left(\nu^2\boldsymbol{I} - \nu\boldsymbol{\Gamma} - \frac{1-2\tau\alpha}{1+2\tau\alpha}\boldsymbol{I}\right),
\end{aligned} \tag{B.7}$$

since the determinants of the orthogonal matrices $\boldsymbol{V}$ and $\boldsymbol{V}^T$ are both 1. The remaining determinant is concerned with a diagonal matrix. Thus, it is equal to the product of the diagonal elements, i.e.

$$\det(\boldsymbol{B} - \nu\boldsymbol{I}) = \prod_{j=1}^{N}\left(\nu^2 - \nu\,\gamma_j - \frac{1-2\tau\alpha}{1+2\tau\alpha}\right). \tag{B.8}$$

For ν to be an eigenvalue of $\boldsymbol{B}$, we need that this product vanishes. This is exactly the case if one or more factors in the product vanish. Hence, we get that for a fixed eigenvalue γ of $\boldsymbol{Q}$, an eigenvalue ν of $\boldsymbol{B}$ satisfies

$$\nu = \frac{\gamma}{2} \pm \sqrt{\frac{\gamma^2}{4} + \frac{1-2\tau\alpha}{1+2\tau\alpha}}. \tag{B.9}$$

For stability of (B.1), we need that $|\nu| \leq 1$ for all solutions ν for all eigenvalues γ of $\boldsymbol{Q}$. To proceed further, we consider the discriminant $\frac{\gamma^2}{4} + \frac{1-2\tau\alpha}{1+2\tau\alpha}$. Since we know that $\boldsymbol{Q}$ is real-valued and symmetric, it follows that γ is a real number. Thus, γ^2 is positive. We also know that τ and α are positive, such that

$$-1 < \frac{1-2\tau\alpha}{1+2\tau\alpha} < 1. \tag{B.10}$$

Therefore, it is possible for the discriminant to have negative values, which results in complex eigenvalues ν. Let us therefore distinguish the three cases of negative discriminant, vanishing discriminant, and positive discriminant. We will see that the last one is the only case which introduces a lower bound on α for unconditional stability.

VANISHING DISCRIMINANT. First, we consider a vanishing discriminant, which can only happen if $2\tau\alpha \geq 1$. This yields

$$\gamma = \pm 2\sqrt{\frac{2\tau\alpha - 1}{2\tau\alpha + 1}}. \tag{B.11}$$

Thus, the eigenvalues of $\boldsymbol{B}$ are given by

$$\nu = \frac{\gamma}{2} = \pm\sqrt{\frac{2\tau\alpha - 1}{2\tau\alpha + 1}}\,. \tag{B.12}$$

Since the fraction takes values between zero and one, the same is true for the square root. Therefore, we have $|\nu| < 1$.

NEGATIVE DISCRIMINANT. If the discriminant is negative, the corresponding values of ν are complex and we can write

$$\nu = \frac{\gamma}{2} \pm \mathrm{i}\sqrt{-\frac{\gamma^2}{4} - \frac{1 - 2\tau\alpha}{1 + 2\tau\alpha}}\,. \tag{B.13}$$

Then we get for the squared absolute value of ν

$$|\nu|^2 = \frac{\gamma^2}{4} + \left(-\frac{\gamma^2}{4} - \frac{1 - 2\tau\alpha}{1 + 2\tau\alpha}\right) = \frac{2\tau\alpha - 1}{2\tau\alpha + 1}\,. \tag{B.14}$$

Surprisingly, this does not depend on γ and we recover once more the condition

$$-1 < \frac{2\tau\alpha - 1}{2\tau\alpha + 1} < 1\,. \tag{B.15}$$

Thus, we again have $|\nu| < 1$.

POSITIVE DISCRIMINANT. In this case, the eigenvalues ν are real-valued. Thus, we get the condition

$$-1 < \frac{\gamma}{2} \pm \sqrt{\frac{\gamma^2}{4} + \frac{1 - 2\tau\alpha}{1 + 2\tau\alpha}} < 1\,. \tag{B.16}$$

If $\tau > 0$ and $\alpha > 0$, this system has the following solutions for γ:

$$\begin{aligned} |\gamma| &< \frac{4\tau\alpha}{1 + 2\tau\alpha}, \qquad \text{if} \quad 0 \le \frac{1 - 2\tau\alpha}{1 + 2\tau\alpha} < 1\,, \\ \sqrt{\frac{8\tau\alpha - 4}{2\tau\alpha + 1}} \le |\gamma| &< \frac{4\tau\alpha}{1 + 2\tau\alpha}, \qquad \text{if} \quad -1 < \frac{1 - 2\tau\alpha}{1 + 2\tau\alpha} < 0\,. \end{aligned} \tag{B.17}$$

If we treat γ as a complex number for the moment, the first condition means that γ has to be inside a disc of radius $\frac{4\tau\alpha}{1+2\tau\alpha}$ around the origin and the second condition means that γ has to be inside that disc, but outside or on the boundary of a second disk of radius $\sqrt{\frac{8\tau\alpha-4}{2\tau\alpha+1}}$. Hence, the second condition is more restrictive.

It remains to determine conditions on τ and α such that (B.17) is always satisfied for all eigenvalues γ of the matrix $\boldsymbol{Q}$. As the matrix $\boldsymbol{A}(\boldsymbol{u}^k)$ is real-valued and symmetric, we can diagonalise it in the same fashion as $\boldsymbol{Q}$: There exist an orthogonal matrix $\boldsymbol{W}$ and a diagonal matrix $\boldsymbol{\Lambda}$ with the eigenvalues λ on its diagonal such that

$$\boldsymbol{A}(\boldsymbol{u}^k) = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^\top. \tag{B.18}$$

With the help of this representation, we can write

$$\begin{aligned} \boldsymbol{Q} &= \frac{4\tau\alpha}{1+2\tau\alpha}\boldsymbol{I} - \frac{2\tau}{1+2\tau\alpha}\boldsymbol{A}\left(\boldsymbol{u}^k\right) \\ &= \frac{4\tau\alpha}{1+2\tau\alpha}\boldsymbol{W}\boldsymbol{W}^T - \frac{2\tau}{1+2\tau\alpha}\boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T \\ &= \boldsymbol{W}\left(\frac{4\tau\alpha}{1+2\tau\alpha}\boldsymbol{I} - \frac{2\tau}{1+2\tau\alpha}\boldsymbol{\Lambda}\right)\boldsymbol{W}^T. \end{aligned} \tag{B.19}$$

Hence, the eigenvalues of $\boldsymbol{Q}$ are given by

$$\gamma = \frac{4\tau\alpha}{1+2\tau\alpha} - \frac{2\tau}{1+2\tau\alpha}\lambda\,, \tag{B.20}$$

where λ is an eigenvalue of $\boldsymbol{A}(\boldsymbol{u}^k)$.

With this formula, the first case of (B.17) reduces to

$$0 < \lambda < 4\alpha\,. \tag{B.21}$$

This condition is similar to the stability condition of the explicit scheme, however now for α instead of $\frac{1}{\tau}$. The estimate on the left-hand side is always fulfilled if $\boldsymbol{A}(\boldsymbol{u}^k)$ is positive definite. For now, we assume that $\boldsymbol{A}(\boldsymbol{u}^k)$ is positive definite and consider the case of a zero eigenvalue afterwards.

The inequality (B.21) has to hold for every eigenvalue of $\boldsymbol{A}(\boldsymbol{u}^k)$. If $\boldsymbol{A}(\boldsymbol{u}^k)$ is positive definite, we can replace λ by the spectral radius $\rho(\boldsymbol{A}(\boldsymbol{u}^k))$ as an upper bound. Hence, we arrive at

$$\alpha > \frac{\rho\left(\boldsymbol{A}(\boldsymbol{u}^k)\right)}{4}\,. \tag{B.22}$$

This is in line with the result that has been derived for the linear case [156].

For the second case in (B.17), plugging in (B.20) and simplifying yields

$$\begin{aligned} &0 < \lambda \leq 2\alpha - \sqrt{4\alpha^2 - \frac{1}{\tau^2}} \qquad \text{or} \\ &2\alpha + \sqrt{4\alpha^2 - \frac{1}{\tau^2}} \leq \lambda < 4\alpha\,. \end{aligned} \tag{B.23}$$

As we are in the case in which $-1 < \frac{1-2\tau\alpha}{1+2\tau\alpha} < 0$, the square root is a nonnegative real number.

Moreover, we have to investigate what happens if

$$\begin{aligned} &-1 \leq \frac{1-2\tau\alpha}{1+2\tau\alpha} < 0 \qquad \text{and} \\ &2\alpha - \frac{\sqrt{4\alpha^2\tau^2-1}}{\tau^2} < \lambda < 2\alpha + \frac{\sqrt{4\alpha^2\tau^2-1}}{\tau^2}\,. \end{aligned} \tag{B.24}$$

With some tedious but straight-forward computations, one can show that this case corresponds exactly to the case with a negative discriminant, which did not introduce any new stability conditions.

Lastly, we consider the case where $\boldsymbol{A}(\boldsymbol{u}^k)$ is only positive semidefinite. If $2\tau\alpha < 1$, then $\lambda = 0$ lies in the range given for λ in (B.24). For this case, we have already shown that $|\nu| < 1$ and the scheme is stable.

If $2\tau\alpha > 1$, we obtain from (B.20):

$$\gamma = \frac{4\tau\alpha}{1 + 2\tau\alpha}\,. \tag{B.25}$$

We can plug this value of γ into (B.11) to see that $\nu = 1$ is always a solution in this case. We have to ensure that all other solutions for ν fulfil $|\nu| < 1$. The other solution of (B.11) for $\lambda = 0$ is

$$\nu = \frac{2\tau\alpha - 1}{2\tau\alpha + 1}\,, \tag{B.26}$$

which yields $|\nu| < 1$ since $\tau > 0$ and $\alpha > 0$. All other eigenvalues of $\boldsymbol{B}$ have an absolute value of less than one by the considerations above. Thus, we can conclude that also in this case, the Du Fort–Frankel scheme is stable.

Consequently, the only stability condition on α is the one in (B.22). In a similar manner as for the symmetric ResNet, we can transform this condition into an a priori constraint

$$\alpha \geq \frac{L}{4}\,. \tag{B.27}$$

This constraint helps us to stabilise the Du Fort–Frankel networks in the same way as the symmetric ResNet.

C

STABILITY OF MULTISCALE ARCHITECTURES

In Chapter 7, we have explicitly proven the Euclidean stability of generalised one-dimensional diffusion blocks and resulting network architectures. In the following, we generalise this proof to the ResNeXt extensions proposed in Chapter 8.

Theorem 5 (Euclidean Stability of Symmetric ResNeXts)**.** *Consider a ResNeXt chaining generalised multiscale diffusion blocks of the unifying form* (8.34)

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \tau \sum_{\ell=1}^{L} \omega_\ell \, \boldsymbol{K}_\ell^\top \boldsymbol{\Phi}\left(\boldsymbol{u}^k, \boldsymbol{K}_\ell \boldsymbol{u}^k\right). \tag{C.1}$$

with a coupling activation function $\tau\boldsymbol{\Phi}$ and convolutions $\boldsymbol{K}_\ell$ for $\ell = 1, \dots, L$. Furthermore, we assume that the activation function has Lipschitz constant P. Then the ResNeXt is well-posed and stable in the Euclidean norm, if the time step size τ is bounded by

$$\tau \leq \frac{2}{P \sum_{\ell=1}^{L} \omega_\ell \|\boldsymbol{K}_\ell\|_2^2}. \tag{C.2}$$

Here, $\|\cdot\|_2$ denotes the spectral norm induced by the Euclidean norm.

Proof. As in the one-dimensional setting, the application of the activation function in each channel can be rewritten by a multiplication with diagonal matrices $\boldsymbol{G}_\ell$, containing diffusivity values on their diagonal. As the diffusivities are smooth, well-posedness follows from the continuity of the operator $(\boldsymbol{I} - \tau \sum_{\ell=1}^{L} \omega_\ell \boldsymbol{K}_\ell^\top \boldsymbol{G}_\ell \boldsymbol{K}_\ell)$ [369].

In the following, we show that the stability requirement (C.2) guarantees that this scheme constitutes a contractive mapping which in turn yields Euclidean stability. The explicit scheme (C.1) is a contraction mapping if the eigenvalues of the operator lie within the interval $[-1, 1]$. To this end, we now bound the eigenvalues of the operator.

The nonnegative diagonal matrices $\boldsymbol{G}_\ell$ are positive semi-definite, since the diffusivity is nonnegative. The symmetric multiplication with $\boldsymbol{K}_\ell$ does not change this. Thus, each matrix $\boldsymbol{K}_\ell^\top \boldsymbol{G}_\ell \boldsymbol{K}_\ell$ is positive semi-definite. As a consequence, we can simply estimate the eigenvalues of each summand to obtain the eigenvalues of the full summation.

The maximal eigenvalue of each diffusivity matrix $\boldsymbol{G}_\ell$ is the supremum of the diffusivity g, which in turn is bounded by the Lipschitz constant P of Φ:

$$P = \sup_s \left|\Phi'(s)\right| = \sup_s \left|g(s^2) + 2s^2 g'(s^2)\right| \geq \sup_s \left|g(s^2)\right|. \tag{C.3}$$

As a consequence the eigenvalues of $\boldsymbol{K}_\ell^\top \boldsymbol{G}_\ell \boldsymbol{K}_\ell$ lie within the interval $\left[0, P \|\boldsymbol{K}_\ell\|_2^2\right]$.

Then the eigenvalues of the full operator $(\boldsymbol{I} - \tau \sum_{\ell=1}^{L} \omega_\ell \boldsymbol{K}_\ell^\top \boldsymbol{G}_\ell \boldsymbol{K}_\ell)$ lie within $\left[1 - \tau P \sum_{\ell=1}^{L} \omega_\ell \|\boldsymbol{K}_\ell\|_2^2, 1\right]$ and the condition

$$1 - \tau P \sum_{\ell=1}^{L} \omega_\ell \|\boldsymbol{K}_\ell\|_2^2 \geq -1 \tag{C.4}$$

yields the bound (C.2). □

As for the stability result of Theorem 4 in the one-dimensional setting, this proof holds independently of the form of the matrices $\boldsymbol{K}_\ell$. The central structural requirement is the symmetric convolution structure $\boldsymbol{W}_{2,\ell} = -\boldsymbol{W}_{1,\ell}^\top$ on each scale.

Moreover, the only requirement for the activation function is that it constitutes a nonnegative rescaling of its input. This is the case for all three models considered, since they use nonnegative diffusivities. However, this also includes e.g. the ReLU function, which rescales positive arguments with one and negative ones with zero.

This proof can also be extended further to activations which are applied independently for each channel. In that case, the Lipschitz constant P is the upper bound of all Lipschitz constants P_ℓ of the activations $\boldsymbol{\Phi}_\ell$. This includes our integrodifferential models of Chapter 5.

The time step size bound is governed by two effects. A finer sampling of scales for a fixed largest scale decreases ω_ℓ and increases the number of summation terms. However, we find that ω_ℓ decreases more quickly than the sum increases. Thus, a finer sampling of scales drives the time step size limit towards an upper bound. This is a numerical effect.

On the other hand, adding additional scales beyond the current largest one decreases the time step size as expected, since both the summation and the spectral radius of the corresponding filters increases. This is an effect of the model itself.

It brings forth an interesting question: Is it worthwhile introducing more coarse scales at the cost of a smaller time step size, or are multiple iterations of few scales with larger time steps more beneficial? With the help of our interpretations, one can relate this question to the debate of breadth versus width in the neural network literature; see e.g. [405]. Detailed practical evaluations of this trade-off in future work may be insightful not only for our diffusion models, but also for the resulting network architectures.

D

CONTRIBUTIONS AND PUBLICATIONS

D.1 FURTHER CONTRIBUTIONS

The contributions that have been discussed in this thesis only cover my first-authored publications. This section briefly discusses several other projects that I was involved in as a co-author.

- We have presented the concept of using PDE residuals in neural surrogate solvers in Chapters 7 and 10. In combination with deep energies [150], this allows to solve any well-behaved PDE- or energy-based model with a neural surrogate. We evaluate the potential of this idea in the work of Schrader et al. [333]. Therein, we solve challenging limit cases for inpainting with Euler's elastica [251] with the help of neural methods. The resulting surrogates solve the inpainting problems with high accuracy and efficiency, without having to rely on complex discretisations and hand-tuning of numerical solvers.

- For inpainting-based image compression, EED is still the state-of-the-art inpainting operator. However, it is not well-suited for inpainting textured image areas. Transform-based codecs such as JPEG [278] excel in exactly these cases. This was the motivation to come up with an hybrid block-based codec called B-EED by Andris et al. [19] which generalised probabilistic sparsification and nonlocal pixel exchange to blocks of pixels. Remaining blocks are compressed with JPEG, and missing blocks are inpainted with EED. The B-EED codec can compete with the state-of-the-art R-EED-LP codec [327], while being much more efficient due to the block optimisation strategies.

- The regular grid codec with joint inpainting and prediction (RJIP) of Peter [280] uses highly efficient Shepard interpolation for inpainting-based compression. This was the basis for several extensions. In the work of Mohideen et al. [258], we extended the RJIP codec to colour images and introduced more efficient optimisation techniques. Moreover, our anisotropic Shepard interpolation model introduced in Chapter 9 is one of the two pillars of the tree-based anisotropic Shepard codec of Mohideen et al., for which a preprint is currently being prepared. Therein, anisotropy and tree-based mask optimisation are combined with the efficiency of Shepard interpolation to design a codec that balances quality and efficiency.

- Chapter 10 presents a model for learning masks that are suitable for diffusion inpainting. Therein, we supplement our conference publication [9] which only focused on homogeneous diffusion inpainting with an extension on biharmonic and EED inpainting. An orthogonal extension considers not only the automatic selection of mask pixel positions, but also an optimised colour value at that position. This additional tonal optimisation can improve the inpainting quality drastically [183, 246]. As for the spatial positions, tonal optimisation is a complex problem. We present a deep learning based solution in the work of Peter et al. [283] as an extension of our conference publication [9]. Therein, we combine spatial and tonal optimisation for homogeneous diffusion inpainting in a deep learning framework. This yields results which are competitive to the state of the art, while also being several orders of magnitude faster.

D.2 LIST OF PUBLICATIONS

JOURNAL PUBLICATIONS

- **T. Alt**, K. Schrader, J. Weickert, P. Peter and M. Augustin: Designing rotationally invariant neural networks from PDEs and variational methods, *Research in the Mathematical Sciences*, Vol. 9, No. 3, Article no. 52, Sept. 2022
- **T. Alt**, K. Schrader, M. Augustin, P. Peter and J. Weickert: Connections between numerical algorithms for PDEs and neural networks, *Journal of Mathematical Imaging and Vision*, Online first, June 2022

CONFERENCE PUBLICATIONS

- K. Schrader, **T. Alt**, J. Weickert and M. Ertel: CNN-based Euler's elastica inpainting with deep energy and deep image prior, *Proc. 10th European Workshop on Visual Information Processing*, Lisbon, Portugal, July 2022, IEEE Computer Society Press
- **T. Alt**, P. Peter and J. Weickert: Learning sparse masks for diffusion-based image inpainting, In A. J. Pinho, P. Georgieva, L. F. Teixeira and J. A. Sánchez (Eds.): *Pattern Recognition and Image Analysis*, Lecture Notes in Computer Science, Vol. 13256, 528–539, Springer, Cham, May 2022
- **T. Alt** and J. Weickert: Learning integrodifferential models for image denoising, *Proc. 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2045–2049, Toronto, Canada, June 2021, IEEE Computer Society Press
- S. Andris, J. Weickert, **T. Alt** and P. Peter: JPEG meets PDE-based image compression, *Proc. 35th Picture Coding Symposium*, Bristol, UK, June 2021, IEEE Computer Society Press

- **T. Alt**, P. Peter, J. Weickert and K. Schrader: Translating numerical concepts for PDEs into neural architectures, In A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin and L. Simon (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 12679, 294–306, Springer, Cham, 2021
- **T. Alt** and J. Weickert: Learning a generic adaptive wavelet shrinkage function for denoising, *Proc. 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018–2022, Barcelona, Spain, May 2020, IEEE Computer Society Press

TECHNICAL REPORTS

- P. Peter, K. Schrader, **T. Alt** and J. Weickert: Deep spatial and tonal optimisation for homogeneous diffusion inpainting, *arXiv:2208.14371v2 [cs.LG]*, Sept. 2022
- R. M. K. Mohideen, P. Peter, **T. Alt**, J. Weickert and A. Scheer: Compressing colour images with joint inpainting and prediction, *arXiv:2010.09866 [eess.IV]*, Oct. 2020
- **T. Alt**, J. Weickert and P. Peter: Translating diffusion, wavelets, and regularisation into residual networks, *arXiv:2002.02753v3 [cs.LG]*, June 2020

THESES

- **T. Alt**: Coupled optical flow, *Bachelor's Thesis in Computer Science, Saarland University, Saarbrücken, Germany*, Aug. 2016

E

BIBLIOGRAPHY

[1] M. Abadi et al.: TensorFlow: large-scale machine learning on heterogeneous systems, 2015, Available at https://www.tensorflow.org/, last visited March 30, 2022 (cited on pp. 54, 60).

[2] R. Acar and C. R. Vogel: Analysis of bounded variation penalty methods for ill–posed problems, *Inverse Problems*, Vol. 10, 1217–1229, 1994 (cited on p. 19).

[3] T. Acar and M. Gökmen: Image coding using weak membrane model of images, In A. K. Katsaggelos (Ed.): *Visual Communications and Image Processing '94*, Proceedings of SPIE, Vol. 2308, 1221–1230, SPIE Press, Bellingham, 1994 (cited on p. 68).

[4] R. Achanta, N. Arvanitopoulos and S. Süsstrunk: Extreme image completion, *Proc. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, 1333–1337, New Orleans, LA, Mar. 2017, IEEE Computer Society Press (cited on pp. 67, 187).

[5] R. D. Adam, P. Peter and J. Weickert: Denoising by inpainting, In F. Lauze, Y. Dong and A. B. Dahl (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 10302, 121–132, Springer, Cham, 2017 (cited on pp. 67, 207).

[6] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt and J. M. Ogden: Pyramid methods in image processing, *RCA Engineer*, Vol. 29, No. 6, 33–41, 1984 (cited on p. 41).

[7] A. Adler, Y. Hel-Or and M. Elad: A weighted discriminative approach for image denoising with overcomplete representations, *Proc. 2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 782–785, Dallas, TX, Mar. 2010, IEEE Computer Society Press (cited on p. 74).

[8] M. Aharon, M. Elad and A. Bruckstein: K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation, *IEEE Transactions on Signal Processing*, Vol. 54, No. 11, 4311–4322, Nov. 2006 (cited on p. 65).

[9] T. Alt, P. Peter and J. Weickert: Learning sparse masks for diffusion-based image inpainting, In A. J. Pinho, P. Georgieva, L. F. Teixeira and J. A. Sánchez (Eds.): *Pattern Recognition and Image Analysis*, Lecture Notes in Computer Science, Vol. 13256, 528–539, Springer, Cham, 2022 (cited on pp. 49, 209, 240).

[10] T. Alt, P. Peter, J. Weickert and K. Schrader: Translating numerical concepts for PDEs into neural architectures, In A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin and L. Simon (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 12679, 294–306, Springer, Cham, 2021 (cited on pp. 49, 131, 144, 146, 166).

[11] T. Alt, K. Schrader, M. Augustin, P. Peter and J. Weickert: Connections between numerical algorithms for PDEs and neural networks, *Journal of Mathematical Imaging and Vision*, Online first, June 2022 (cited on pp. 49, 131, 231).

[12] T. Alt, K. Schrader, J. Weickert, P. Peter and M. Augustin: Designing rotationally invariant neural networks from PDEs and variational methods, *Russian Mathematical Surveys*, Vol. 9, No. 3, Article no. 52, Sept. 2022 (cited on p. 163).

[13] T. Alt and J. Weickert: Learning a generic adaptive wavelet shrinkage function for denoising, *Proc. 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018–2022, Barcelona, Spain, May 2020, IEEE Computer Society Press (cited on pp. 74, 92).

[14] T. Alt and J. Weickert: Learning integrodifferential models for image denoising, *Proc. 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2045–2049, Toronto, Canada, June 2021, IEEE Computer Society Press (cited on pp. 92, 96, 104, 172, 177).

[15] T. Alt, J. Weickert and P. Peter: Translating diffusion, wavelets, and regularisation into residual networks, *arXiv:2002.02753v3 [cs.LG]*, June 2020 (cited on pp. 49, 114).

[16] F. Andreu, C. Ballester, V. Caselles and J. M. Mazón: Minimizing total variation flow, *Differential and Integral Equations*, Vol. 14, No. 3, 321–360, Mar. 2001 (cited on pp. 19, 65, 123).

[17] S. Andris, P. Peter, R. M. K. Mohideen, J. Weickert and S. Hoffmann: Inpainting-based video compression in FullHD, In A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin and L. Simon (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 12679, 425–436, Springer, Cham, 2021 (cited on pp. 67, 207).

[18] S. Andris, P. Peter and J. Weickert: A proof-of-concept framework for PDE-based video compression, *Proc. 32nd Picture Coding Symposium*, Nuremberg, Germany, Dec. 2016 (cited on p. 67).

[19] S. Andris, J. Weickert, T. Alt and P. Peter: JPEG meets PDE-based image compression, *Proc. 35th Picture Coding Symposium*, Bristol, UK, June 2021, IEEE Computer Society Press (cited on pp. 68, 239).

[20] P. Arbelaez, M. Maire, C. Fowlkes and J. Malik: Contour detection and hierarchical image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 5, 898–916, Aug. 2011 (cited on pp. 82, 96, 213).

[21] P. Arias, G. Facciolo, V. Caselles and G. Sapiro: A variational framework for exemplar-based image inpainting, *International Journal of Computer Vision*, Vol. 93, No. 3, 319–347, July 2011 (cited on p. 37).

[22] M. Arjovsky, S. Chintala and L. Bottou: Wasserstein generative adversarial networks, *Proc. 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh (Ed.), Vol. 70, Proceedings of Machine Learning Research, 214–223, Sydney, Australia, Aug. 2017 (cited on pp. 60, 63).

[23] S. Arridge, P. Maas, O. Öktem and C.-B. Schönlieb: Solving inverse problems using data-driven models, *Acta Numerica*, Vol. 28, 1–174, May 2019 (cited on p. 63).

[24] Association for Computing Machinery: Fathers of the deep learning revolution receive ACM A. M. Turing award, Mar. 2019, Available at https://www.acm.org/media-center/2019/march/turing-award-2018, last visited March 30, 2022 (cited on pp. 58, 60).

[25] P. Athavale and E. Tadmor: Integro-differential equations based on (BV, L^1) image decomposition, *SIAM Journal on Imaging Sciences*, Vol. 4, No. 1, 300–312, Mar. 2011 (cited on p. 92).

[26] G. Aubert, R. Deriche and P. Kornprobst: Computing optical flow via variational techniques, *SIAM Journal on Applied Mathematics*, Vol. 60, No. 1, 156–182, 1999 (cited on p. 37).

[27] G. Aubert and P. Kornprobst: Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations, Applied Mathematical Sciences, Vol. 147, Springer, New York, 2002 (cited on pp. 2, 6, 15, 36, 37).

[28] T. Avant and K. A. Morgansen: Analytical bounds on the local Lipschitz constants of ReLU networks, *arXiv:2104.14672v1 [cs.LG]*, Apr. 2021 (cited on p. 64).

[29] M. Bäker: Another look at neural multigrid, *International Journal of Modern Physics C*, Vol. 8, No. 2, 191–205, Apr. 1997 (cited on p. 131).

[30] M. Bäker, G. Mack and M. Speh: Multigrid meets neural nets, *Nuclear Physics B - Proceedings Supplements*, Vol. 30, 269–272, Mar. 1993 (cited on p. 131).

[31] C. Ballester, V. Caselles, J. Verdera, M. Bertalmío and G. Sapiro: A variational model for filling-in gray level and color images, *Proc. 18th International Conference on Computer Vision*, Vol. 1, 10–16, Vancouver, Canada, July 2001, IEEE Computer Society Press (cited on p. 37).

[32] S. Barbeiro and D. Lobo: Learning stable nonlinear cross-diffusion models for image restoration, *Journal of Mathematical Imaging and Vision*, Vol. 62, 223–237, Apr. 2020 (cited on p. 92).

[33] T. Barbu: Feature keypoint-based image compression technique using a well-posed nonlinear fourth-order PDE-based model, *Mathematics*, Vol. 8, No. 6, Article no. 930, June 2020 (cited on p. 22).

[34] S. Battiato, G. Gallo and F. Stanco: Smart interpolation by anisotropic diffusion, *Proc. Twelvth International Conference on Image Analysis and Processing*, 572–577, Montova, Italy, Sept. 2003, IEEE Computer Society Press (cited on p. 67).

[35] A. G. Baydin, B. A. Pearlmutter, A. A. Radul and J. M. Siskind: Automatic differentiation in machine learning: a survey, *Journal of Machine Learning Research*, Vol. 18, No. 153, 1–43, 2017 (cited on p. 54).

[36] A. Beck and M. Teboulle: Fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM Journal on Imaging Sciences*, Vol. 2, 183–202, 2009 (cited on p. 19).

[37] J. Behrmann, S. Dittmer, P. Fernsel and P. Maass: Analysis of invariance and robustness via invertibility of ReLU-networks, *arXiv:1806.09730v2 [cs.LG]*, June 2018 (cited on p. 114).

[38] A. Belahmidi and F. Guichard: A partial differential equation approach to image zoom, *Proc. 2004 IEEE International Conference on Image Processing*, Vol. 1, 649–652, Singapore, Oct. 2004, IEEE Computer Society Press (cited on p. 67).

[39] Z. Belhachmi, D. Bucur, B. Burgeth and J. Weickert: How to choose interpolation data in images, *SIAM Journal on Applied Mathematics*, Vol. 70, No. 1, 333–352, 2009 (cited on pp. 68, 207, 208, 210, 214).

[40] M. Belkin, D. Hsu, S. Ma and S. Mandal: Reconciling modern machine-learning practice and the classical bias–variance trade-off, *Proceedings of the National Academy of Sciences*, Vol. 116, No. 32, 15849–15854, Aug. 2019 (cited on pp. 2, 61).

[41] I. Ben-Yair, G. B. Shalom, M. Eliasof and E. Treister: Quantized convolutional neural networks through the lens of partial differential equations, *Russian Mathematical Surveys*, Vol. 9, No. 4, Article no. 58, Sept. 2022 (cited on p. 63).

[42] Y. Bengio, P. Simard and P. Frasconi: Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, 157–166, Mar. 1994 (cited on pp. 49, 53).

[43] M. Benning, E. Celledoni, M. J. Erhardt, B. Owren and C.-B. Schönlieb: Deep learning as optimal control problems: models and numerical methods, *IFAC-PapersOnline*, Vol. 54, No. 9, 620–623, 2021 (cited on p. 130).

[44] L. Bergerhoff, M. Cardénas, J. Weickert and M. Welk: Stable backward diffusion models that minimise convex energies, *Journal of Mathematical Imaging and Vision*, Vol. 62, No. 6-7, 941–960, July 2020 (cited on pp. 21, 74).

[45] M. Bertalmío, V. Caselles, S. Masnou and G. Sapiro: Inpainting, In K. Ikeuchi (Ed.): *Computer Vision: A Reference Guide*, 401–416, Springer, New York, 2014 (cited on p. 67).

[46] M. Bertalmío, G. Sapiro, V. Caselles and C. Ballester: Image inpainting, *Proc. SIGGRAPH 2000*, 417–424, New Orleans, LI, July 2000 (cited on pp. 15, 67).

[47] D. Bertoin, J. Bolte, S. Gerchinovitz and E. Pauwels: Numerical influence of $\mathrm{ReLU}'(0)$ on backpropagation, *Proc. 35th International Conference on Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang and J. W. Vaughan (Ed.), Vol. 34, Advances in Neural Information Processing Systems, 468–479, Virtual, Dec. 2021 (cited on p. 54).

[48] K. Bodduna, J. Weickert and M. Cárdenas: Multi-frame super-resolution from noisy data, In A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin and L. Simon (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 12679, 565–576, Springer, Cham, 2021 (cited on p. 163).

[49] B. Bohn, M. Griebel and D. Kannan: Deep neural networks and PIDE discretizations, *SIAM Journal on Mathematics of Data Science*, Vol. 4, No. 3, 1145–1170, Aug. 2022 (cited on p. 62).

[50] S. Bonettini, I. Loris, F. Porta, M. Prato and S. Rebegoldi: On the convergence of a linesearch based proximal-gradient method for nonconvex optimization, *Inverse Problems*, Vol. 33, No. 5, Article No. 055005, Mar. 2017 (cited on pp. 68, 208).

[51] Y.-L. Boureau, J. Ponce and Y. LeCun: A theoretical analysis of feature pooling in visual recognition, *Proc. 27th International Conference on Machine Learning*, J. Fürnkranz and T. Joachims (Ed.), 111–118, Haifa, Israel, June 2010 (cited on p. 46).

[52] A. Bourquard and M. Unser: Anisotropic interpolation of sparse generalized image samples, *IEEE Transactions on Image Processing*, Vol. 22, No. 2, 459–472, 2013 (cited on p. 67).

[53] A. Brandt: Multi-level adaptive solutions to boundary-value problems, *Mathematics of Computation*, Vol. 31, No. 138, 333–390, Apr. 1977 (cited on pp. 35, 129, 130, 142).

[54] W. L. Briggs, V. E. Henson and S. F. McCormick: A Multigrid Tutorial, Second, SIAM, Philadelphia, 2000 (cited on pp. 35, 129, 130, 142).

[55] E.-M. Brinkmann, M. Burger and I. Grah: Regularization with sparse vector fields: from image compression to TV-type reconstruction, In J.-F. Aujol, M. Nikolova and N. Papadakis (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 9087, 191–202, Springer, Berlin, 2015 (cited on p. 68).

[56] C. Brito-Loeza and K. Chen: Fast numerical algorithms for Euler's Elastica inpainting model, *International Journal of Modern Mathematics*, Vol. 5, No. 2, 157–182, 2010 (cited on p. 67).

[57] C. Brito-Loeza and K. Chen: Multigrid algorithm for high order denoising, *SIAM Journal on Imaging Sciences*, Vol. 3, No. 3, 363–389, 2010 (cited on p. 142).

[58] T. Brox, A. Bruhn, N. Papenberg and J. Weickert: High accuracy optical flow estimation based on a theory for warping, In T. Pajdla and J. Matas (Eds.): *Computer Vision – ECCV 2004, Part IV*, Lecture Notes in Computer Science, Vol. 3024, 25–36, Springer, Berlin, 2004 (cited on p. 37).

[59] A. Bruhn, J. Weickert, T. Kohlberger and C. Schnörr: A multigrid platform for real-time motion computation with discontinuity-preserving variational methods, *International Journal of Computer Vision*, Vol. 70, No. 3, 257–277, Dec. 2006 (cited on p. 142).

[60] J. Bruna and S. Mallat: Invariant scattering convolution networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 35, No. 8, 1872–1886, Aug. 2013 (cited on p. 63).

[61] A. Buades, B. Coll and J.-M. Morel: A non-local algorithm for image denoising, *Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, 60–65, San Diego, CA, June 2005, IEEE Computer Society Press (cited on p. 65).

[62] L. Bungert, R. Raab, T. Roith, L. Schwinn and D. Tenbrinck: CLIP: Cheap Lipschitz training of neural networks, In A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin and L. Simon (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 12679, 307–319, Springer, Cham, 2021 (cited on pp. 64, 137).

[63] M. Cárdenas, J. Weickert and S. Schäffer: A linear scale-space theory for continuous nonlocal evolutions, In J.-F. Aujol, M. Nikolova and N. Papadakis (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 9087, 103–114, Springer, Berlin, 2015 (cited on p. 92).

[64] S. Carlsson: Sketch based coding of grey level images, *Signal Processing*, Vol. 15, 57–83, 1988 (cited on pp. 67, 68).

[65] F. Catté, P.-L. Lions, J.-M. Morel and T. Coll: Image selective smoothing and edge detection by nonlinear diffusion, *SIAM Journal on Numerical Analysis*, Vol. 32, 1895–1909, 1992 (cited on pp. 21, 23, 92–94, 101).

[66] A. Chambolle: Total variation minimization and a class of binary MRF models, In A. Rangarajan, B. C. Vemuri and A. L. Yuille (Eds.): *Energy Minimization Methods in Computer Vision and Pattern Recognition – EMMCVPR 2005*, Lecture Notes in Computer Science, Vol. 3757, 136–152, Springer, Berlin, 2005 (cited on p. 19).

[67] T. F. Chan, S. Esedoglu and F. E. Park: A fourth order dual method for staircase reduction in texture extraction and image restoration problems, *Proc. 17th IEEE International Conference on Image Processing*, 4137–4140, Hong Kong, Sept. 2010, IEEE Computer Society Press (cited on p. 22).

[68] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert and E. Holtham: Reversible architectures for arbitrarily deep residual neural networks, *Proc. 32nd AAAI Conference on Artificial Intelligence*, 2811–2818, New Orleans, LA, Feb. 2018 (cited on p. 114).

[69] S. G. Chang, B. Yu and M. Vetterli: Adaptive wavelet thresholding for image denoising and compression, *IEEE Transactions on Image Processing*, Vol. 9, No. 9, 1532–1546, Sept. 2002 (cited on p. 74).

[70] P. Charbonnier, L. Blanc-Féraud, G. Aubert and M. Barlaud: Two deterministic half-quadratic regularization algorithms for computed imaging, *Proc. 1994 IEEE International Conference on Image Processing*, Vol. 2, 168–172, Austin, TX, Nov. 1994, IEEE Computer Society Press (cited on pp. 19, 37, 104, 123, 148, 156).

[71] K. Chellapilla, S. Puri and P. Simard: High performance convolutional neural networks for document processing, *Proc. 10th International Workshop on Frontiers in Handwriting Recognition*, Rennes, France, Oct. 2006 (cited on p. 59).

[72] A. Chen and G. Lin: Robust data-driven discovery of partial differential equations with time-dependent coefficients, *arXiv:2102.01432v1 [stat.ML]*, Feb. 2021 (cited on p. 62).

[73] R. T. Q. Chen, Y. Rubanova, J. Bettencourt and D. K. Duvenaud: Neural ordinary differential equations, *Proc. 32nd International Conference on Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett (Ed.), Vol. 31, Advances in Neural Information Processing Systems, 6571–6583, Montréal, Canada, Dec. 2018 (cited on pp. 2, 49, 62).

[74] Y. Chen, Z. X. Lyu, X. Kang and Z. J. Wang: A rotation-invariant convolutional neural network for image enhancement forensics, *Proc. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2111–2115, Calgary, Canada, Apr. 2018, IEEE Computer Society Press (cited on pp. 64, 162).

[75] Y. Chen and T. Pock: Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 6, 1256–1272, Aug. 2016 (cited on pp. 2, 48, 62, 63, 65, 74, 92, 118, 130, 141, 176).

[76] Y. Chen, R. Ranftl and T. Pock: A bi-level view of inpainting-based image compression, *Proc. 19th Computer Vision Winter Workshop*, Z. Kúkelová and J. Heller (Ed.), Křtiny, Czech Republic, Feb. 2014 (cited on pp. 67, 68, 208).

[77] V. Chizhov and J. Weickert: Efficient data optimisation for harmonic inpainting with finite elements, In N. Tsapatsoulis, A. Panayides, T. Theocharides, A. Lanitis, C.S. Pattichis and M. Vento (Eds.): *Computer Analysis of Images and Patterns. Part 2*. Lecture Notes in Computer Science, Vol. 13053, 432–441, Springer, Cham, 2021 (cited on pp. 67, 68, 208).

[78] D. Ciresan, A. Giusti, L. M. Gambardella and J. Schmidhuber: Deep neural networks segment neuronal membranes in electron microsopy images, *Proc. 26th International Conference on Neural Information Processing Systems*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger (Ed.), Vol. 25, Advances in Neural Information Processing Systems, 2852–2860, Lake Tahoe, NV, Dec. 2012 (cited on p. 48).

[79] D. Ciresan, U. Meier, L. M. Gambardella and J. Schmidhuber: Deep big simple neural nets for handwritten digit recognition, *12*, Vol. 22, No. 12, 3207–3220, Nov. 2010 (cited on pp. 58, 59).

[80] D. Ciresan, U. Meier, J. Masci, L. M. Gambardella and J. Schmidhuber: Flexible, high performance convolutional neural networks for image classification, *Proc. 22nd International Joint Conference on Artificial Intelligence*, T. Walsh (Ed.), 1237–1242, Barcelona, Spain, July 2011 (cited on pp. 58, 60).

[81] D. Ciresan, U. Meier, J. Masci and J. Schmidhuber: A committee of neural networks for traffic sign classification, *Proc. 2011 International Joint Conference on Neural Networks*, T. Walsh (Ed.), 1918–1921, San Jose, CA, July 2011, IEEE Computer Society Press (cited on p. 60).

[82] A.-S. Cohen, R. Cont, A. Rossier and R. Xu: Scaling properties of deep residual networks, *Proc. 38th International Conference on Machine Learning*, Vol. 139, Proceedings of Machine Learning Research, 2039–2048, Virtual, July 2021 (cited on pp. 49, 62).

[83] T. Cohen, M. Geiger, J. Koehler and M. Welling: Spherical CNNs, *Proc. 6th International Conference on Learning Representations*, Vancouver, Canada, Apr. 2018 (cited on p. 162).

[84] T. Cohen and M. Welling: Group equivariant convolutional networks, *Proc. 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger (Ed.), Vol. 48, Proceedings of Machine Learning Research, 2990–2999, New York City, NY, June 2016 (cited on pp. 64, 162).

[85] R. R. Coifman and D. Donoho: Translation invariant denoising, In A. Antoine and G. Oppenheim (Eds.): *Wavelets in Statistics*, 125–150, Springer, New York, 1995 (cited on pp. 40, 65, 76).

[86] P. L. Combettes and J.-C. Pesquet: Deep neural network structures solving variational inequalities, *Set-Valued and Variational Analysis*, Vol. 28, No. 3, 491–518, Sept. 2020 (cited on p. 62).

[87] A. Criminisi, P. Pérez and K. Toyama: Region filling and object removal by exemplar-based image inpainting, *IEEE Transactions on Image Processing*, Vol. 13, No. 9, 1200–1212, Sept. 2004 (cited on p. 67).

[88] E. Cuesta-Montero and J. Finat: Image processing by means of a linear integro-differential equation, *Proc. Third IASTED International Conference on Visualization, Imaging and Image Processing*, 438–442, Benalmadena, Spain, Sept. 2003, ACTA Press (cited on p. 92).

[89] G. Cybenko: Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, Vol. 2, 303–314, 1989 (cited on p. 64).

[90] K. Dabov, A. Foi, V. Katkovnik and K. Egiazarian: Image denoising by sparse 3D transform-domain collaborative filtering, *IEEE Transactions on Image Processing*, Vol. 16, No. 8, 2080–2095, Aug. 2007 (cited on p. 65).

[91] Q. Dai, H. Chopp, E. Pouyet, O. Cossairt, M. Walton and A. K. Katsaggelos: Adaptive image sampling using deep learning and its application on X-Ray fluorescence image reconstruction, *IEEE Transactions on Multimedia*, Vol. 22, No. 10, 2564–2578, Dec. 2019 (cited on pp. 49, 209).

[92] E. D'Angelo, L. Jacques, A. Alahi and P. Vandergheynst: From bits to images: inversion of local binary descriptors, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 36, No. 5, 874–887, 2014 (cited on p. 68).

[93] V. Daropoulos, M. Augustin and J. Weickert: Sparse inpainting with smoothed particle hydrodynamics, *SIAM Journal on Applied Mathematics*, Vol. 14, No. 4, 1669–1704, Nov. 2021 (cited on pp. 68, 208).

[94] I. Daubechies: Ten Lectures on Wavelets, SIAM, Philadelphia, 1992 (cited on p. 39).

[95] I. Daubechies, R. DeVore, S. Foucart, B. Hanin and G. Petrova: Nonlinear approximation and (deep) ReLU networks, *Constructive Approximation*, Vol. 55, 127–172, Apr. 2022 (cited on p. 64).

[96] P. De Felice, C. Marangi, G. Nardulli, G. Pasquariello and L. Tedesco: Dynamics of neural networks with non-monotone activation function, *Network: Computation in Neural Systems*, Vol. 4, No. 1, 1–9, 1993 (cited on pp. 114, 123, 131).

[97] L. Demaret, N. Dyn and A. Iske: Image compression by linear splines over adaptive triangulations, *Signal Processing*, Vol. 86, No. 7, 1604–1616, 2006 (cited on pp. 68, 208).

[98] S. Di Zenzo: A note on the gradient of a multi-image, *Computer Vision, Graphics and Image Processing*, Vol. 33, 116–125, 1986 (cited on pp. 95, 169, 190).

[99] S. Didas, G. Steidl and J. Weickert: Discrete multiscale wavelet shrinkage and integrodifferential equations, In P. Schelkens, T. Ebrahimi, G. Christobal and F. Truchetet (Eds.): *Optical and Digital Image Processing – Photonics Europe*, Proceedings of SPIE, Vol. 7000, SPIE Press, Bellingham, 2008 (cited on p. 227).

[100] S. Didas and J. Weickert: Integrodifferential equations for continuous multiscale wavelet shrinkage, *Inverse Problems and Imaging*, Vol. 1, No. 1, 47–62, Feb. 2007 (cited on pp. 92, 172).

[101] S. Didas, J. Weickert and B. Burgeth: Properties of higher order nonlinear diffusion filtering, *Journal of Mathematical Imaging and Vision*, Vol. 35, 208–226, Nov. 2009 (cited on pp. 134, 165).

[102] S. Dieleman, J. De Fauw and K. Kavukcuoglu: Exploiting cyclic symmetry in convolutional neural networks, *Proc. 33rd International Conference on Machine Learning*, Vol. 48, Proceedings of Machine Learning Research, 1889–1898, New York, NY, June 2016 (cited on pp. 64, 162).

[103] R. Distasi, M. Nappi and S. Vitulano: Image compression by B-tree triangular coding, *IEEE Transactions on Communications*, Vol. 45, No. 9, 1095–1100, Sept. 1997 (cited on p. 67).

[104] S. Dittmer, T. Kluth, P. Maass and D. O. Baguer: Regularization by architecture: a deep prior approach for inverse problems, *Journal of Mathematical Imaging and Vision*, Vol. 62, 456–470, Apr. 2020 (cited on p. 63).

[105] B. Dong, Q. Jiang and Z. Shen: Image restoration: wavelet frame shrinkage, nonlinear evolution PDEs, and beyond, *Multiscale Modeling and Simulation*, Vol. 15, No. 1, 606–660, 2017 (cited on p. 119).

[106] H. Dong, G. Yang, F. Liu, Y. Mo and Y. Guo: Automatic brain tumor detection and segmentation using U-Net based fully convolutional networks, In M. Valdés Hernández and V. González-Castro (Eds.): *Medical Image Understanding and Analysis – MIUA 2017*, Communications in Computer and Information Science, Vol. 723, 506–517, Springer, Cham, 2017 (cited on pp. 48, 49).

[107] D. L. Donoho: De-noising by soft thresholding, *IEEE Transactions on Information Theory*, Vol. 41, 613–627, May 1995 (cited on pp. 40, 65, 74, 77, 116, 123).

[108] D. L. Donoho and I. M. Johnstone: Ideal spatial adaptation by wavelet shrinkage, *Biometrica*, Vol. 81, No. 3, 425–455, 1994 (cited on pp. 2, 15, 38, 73).

[109] Y. Du and I. Mordatch: Implicit generation and modeling with energy based models, *Proc. 33rd International Conference on Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox and R. Garnett (Ed.), Vol. 32, Advances in Neural Information Processing Systems, 3603–3613, Vancouver, Canada, Dec. 2019 (cited on p. 63).

[110] E. C. Du Fort and S. P. Frankel: Stability conditions in the numerical treatment of parabolic differential equations, *Mathematical Tables and Other Aids to Computation*, Vol. 7, 135–152, 1953 (cited on pp. 129, 130, 137, 138).

[111] J. Duchi, E. Hazan and Y. Singer: Adaptive subgradient methods for online learning and stochastic optimization, *Journal of Machine Learning Research*, Vol. 12, 2121–2159, July 2011 (cited on p. 56).

[112] R. Duits, B. Smets, E. Bekkers and J. Portegies: Equivariant deep learning via morphological and linear scale space PDEs on the space of positions and orientations, In A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin and L. Simon (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 12679, 27–39, Springer, Cham, 2021 (cited on pp. 64, 114, 162).

[113] W. E: A proposal on machine learning via dynamical systems, *Communications in Mathematics and Statistics*, Vol. 5, 1–11, Mar. 2017 (cited on p. 114).

[114] W. E, J. Han and A. Jentzen: Algorithms for solving high dimensional PDEs: from nonlinear Monte Carlo to machine learning, *Nonlinearity*, Vol. 35, No. 1, Article No. 278, Dec. 2021 (cited on p. 62).

[115] Eastman Kodak Company: Kodak true color image suite, 1999, Available at `http://r0k.us/graphics/kodak/`, last visited March 30, 2022 (cited on pp. 193, 200).

[116] R. Eldan and O. Shamir: The power of depth for feedforward neural networks, *Proc. 29th Conference on Learning Theory*, V. Feldman, A. Rakhlin and O. Shamir (Ed.), 907–940, New York, NY, June 2016 (cited on p. 64).

[117] M. Eliasof, J. Ephrath, R. Ruthotto and E. Treister: Mgic: multigrid-in-channels neural network architectures, *arXiv:2011.09128v2 [cs.CV]*, Nov. 2020 (cited on p. 131).

[118] M. Eliasof, E. Haber and E. Treister: PDE-GCN novel architectures for graph neural networks motivated by partial differential equations, *Proc. 35th International Conference on Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang and J. W. Vaughan (Ed.), Vol. 34, Advances in Neural Information Processing Systems, 3836–3849, Virtual, Dec. 2021 (cited on p. 63).

[119] S. Esedoglu: An analysis of the Perona–Malik scheme, *Communications on Pure and Applied Mathematics*, Vol. 54, No. 12, 1442–1487, May 2001 (cited on p. 21).

[120] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno and D. Song: Robust physical-world attacks on deep learning visual classification, *Proc. 2018 IEEE Conference on Computer Vision and Pattern Recognition*, 1625–1634, Salt Lake City, UT, June 2018, IEEE Computer Society Press (cited on p. 1).

[121] R. Fablet, L. Drumetz and F. Rousseau: End-to-end learning of variational models and solvers for the resolution of interpolation problems, *Proc. 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2360–2364, Toronto, Canada, June 2021, IEEE Computer Society Press (cited on p. 63).

[122] G. Facciolo, P. Arias, V. Caselles and G. Sapiro: Exemplar-based interpolation of sparsely sampled images, In D. Cremers, Y. Boykov, A. Blake and F. R. Schmidt (Eds.): *Energy Minimisation Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science, Vol. 5681, 331–344, Springer, Berlin, 2009 (cited on p. 67).

[123] B. Fasel and D. Gatica-Perez: Rotation-invariant neoperceptron, *Proc. 18th International Conference on Pattern Recognition*, Vol. 3, 336–339, Hong Kong, Aug. 2006, IEEE Computer Society Press (cited on pp. 64, 162).

[124] W. Feng, P. Qiao, X. Xi and Y. Chen: Image denoising via multiscale nonlinear diffusion models, *SIAM Journal on Imaging Sciences*, Vol. 10, No. 3, 1234–1257, 2017 (cited on p. 176).

[125] A. Fick: Ueber Diffusion, *Annalen der Physik*, Vol. 170, No. 1, 59–86, Apr. 1855 (cited on p. 16).

[126] R. W. Floyd and L. Steinberg: An adaptive algorithm for spatial grey scale, *Proceedings of the Society of Information Display*, Vol. 17, 75–77, 1976 (cited on p. 214).

[127] W. Förstner and E. Gülch: A fast operator for detection and precise location of distinct points, corners and centres of circular features, *Proc. ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, 281–305, Interlaken, Switzerland, June 1987 (cited on pp. 92, 93, 164).

[128] R. Franke and G. Nielson: Smooth interpolation of large sets of scattered data, *Numerical Methods in Engineering*, Vol. 15, No. 11, 1691–1704, Nov. 1980 (cited on p. 186).

[129] W. T. Freeman and E. H. Adelson: The design and use of steerable filters, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 9, 891–906, Sept. 1991 (cited on p. 162).

[130] D. S. Fritsch: A medial description of greyscale image structure by gradient-limited diffusion, In R. A. Robb (Ed.): *Visualization in Biomedical Computing '92*, Proceedings of SPIE, Vol. 1808, 105–117, SPIE Press, Bellingham, 1992 (cited on p. 21).

[131] S. Fujieda, K. Takayama and T. Hachisuka: Wavelet convolutional neural networks, *arXiv:1805.08620v1 [cs.CV]*, May 2018 (cited on p. 63).

[132] K. Fukushima: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biological Cybernetics*, Vol. 36, 193–202, 1980 (cited on pp. 1, 44, 58, 59).

[133] K. Fukushima: Artificial vision by multi-layered neural networks: neocognitron and its advances, *Neural Networks*, Vol. 37, 103–119, Jan. 2013 (cited on p. 59).

[134] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev and H.-P. Seidel: Towards PDE-based image compression, In N. Paragios, O. Faugeras, T. Chan and C. Schnörr (Eds.): *Variational, Geometric and Level-Set Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 3752, 37–48, Springer, Berlin, 2005 (cited on p. 67).

[135] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev and H.-P. Seidel: Image compression with anisotropic diffusion, *Journal of Mathematical Imaging and Vision*, Vol. 31, No. 2–3, 255–269, July 2008 (cited on pp. 15, 23, 67, 156, 187, 207).

[136] H.-Y. Gao: Wavelet shrinkage denoising using the non-negative garrote, *Journal of Computational and Graphical Statistics*, Vol. 7, No. 4, 469–488, Dec. 1998 (cited on pp. 74, 77, 123).

[137] I. M. Gelfand and S. V. Fomin: Calculus of Variations, Dover, New York, 2000 (cited on pp. 6, 36, 37).

[138] S. Geman and D. Geman: Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, 721–741, 1984 (cited on p. 123).

[139] R. Gens and P. Domingos: Deep symmetry networks, *Proc. 28th International Conference on Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger (Ed.), Vol. 27, Advances in Neural Information Processing Systems, 2537–2545, Montréal, Canada, Dec. 2014 (cited on pp. 64, 162).

[140] M. Genzel, J. Macdonald and M. März: Solving inverse problems with deep neural networks – robustness included?, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 45, No. 1, 1119–1134, Feb. 2022 (cited on pp. 2, 64).

[141] G. Gerig, O. Kübler, R. Kikinis and F. A. Jolesz: Nonlinear anisotropic filtering of MRI data, *IEEE Transactions on Medical Imaging*, Vol. 11, 221–232, 1992 (cited on pp. 169, 177).

[142] F. A. Gers, J. Schmidhuber and F. Cummins: Learning to forget: continual prediction with LSTM, *Neural Computation*, Vol. 12, No. 10, 2451–2471, 2000 (cited on pp. 47, 59).

[143] S. Gerschgorin: Fehlerabschätzung für das Differenzenverfahren zur Lösung Partieller Differentialgleichungen, *Zeitschrift für Angewandte Mathematik und Mechanik*, Vol. 10, 373–382, 1930 (cited on pp. 10, 96, 136).

[144] A. Gersho and M. R. Gray: Vector Quantization and Signal Compression, The Springer International Series in Engineering and Computer Science, Vol. 159, Springer, New York, 1992 (cited on p. 200).

[145] P. Getreuer, P. Milanfar and X. Luo: Solving image PDEs with a shallow network, *arXiv:2110.08327v1 [cs.CV]*, Oct. 2021 (cited on p. 62).

[146] G. Gilboa and S. Osher: Nonlocal operators with applications to image processing, *Multiscale Modeling and Simulation*, Vol. 7, 1005–1028, Nov. 2008 (cited on p. 92).

[147] G. Gilboa, N. A. Sochen and Y. Y. Zeevi: Forward-and-backward diffusion processes for adaptive image enhancement and denoising, *IEEE Transactions on Image Processing*, Vol. 11, No. 7, 689–703, Nov. 2002 (cited on pp. 21, 74).

[148] G. Gilboa, Y. Zeevi and N. Sochen: Image enhancement segmentation and denoising by time dependent nonlinear diffusion processes, *Proc. 2001 IEEE International Conference on Image Processing*, Vol. 3, 134–137, Thessaloniki, Greece, Oct. 2001, IEEE Computer Society Press (cited on p. 151).

[149] X. Glorot and Y. Bengio: Understanding the difficulty of training deep feedforward neural networks, *Proc. Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256, Sardinia, Italy, May 2010 (cited on pp. 43, 60).

[150] A. Golts, D. Freedman and M. Elad: Deep energy: task driven training of deep neural networks, *IEEE Journal of Selected Topics in Signal Processing*, Vol. 15, No. 2, 324–338, Feb. 2021 (cited on pp. 63, 156, 213, 239).

[151] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville and Y. Bengio: Maxout networks, *Proc. 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester (Ed.), Vol. 28, Proceedings of Machine Learning Research, 1319–1327, Atlanta, GA, June 2013 (cited on pp. 130, 176).

[152] I. J. Goodfellow, Y. Bengio and A. Courville: Deep Learning, MIT Press, Cambridge, MA, 2016 (cited on pp. 1, 15, 41, 57, 59, 60).

[153] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville and Y. Bengio: Generative adversarial nets, *Proc. 28th International Conference on Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger (Ed.), Vol. 27, Advances in Neural Information Processing Systems, 2672–2680, Montréal, Canada, Dec. 2014 (cited on pp. 58, 60).

[154] I. J. Goodfellow, J. Shlens and C. Szegedy: Explaining and harnessing adversarial examples, *Proc. 3rd International Conference on Learning Representations*, Y. Bengio and Y. LeCun (Ed.), San Diego, CA, May 2015 (cited on pp. 1, 61).

[155] Google Corporation: Google scholar metrics: top cited publications over the last five years, Available at `https://scholar.google.com/citations?view_op=top_venues`, Last visited November 2, 2021 (cited on p. 60).

[156] D. Gottlieb and B. Gustafsson: Generalized Du Fort–Frankel methods for parabolic initial-boundary value problems, *SIAM Journal on Numerical Analysis*, Vol. 13, No. 1, 129–144, Mar. 1976 (cited on pp. 138, 231, 234).

[157] H. Gouk, E. Frank, B. Pfahringer and M. J. Cree: Regularisation of neural networks by enforcing Lipschitz continuity, *Machine Learning*, Vol. 110, 393–416, Feb. 2021 (cited on pp. 64, 137).

[158] T. Grandits and T. Pock: Optimizing wavelet bases for sparser representations, In M. Pelillo and E. Hancock (Eds.): *Energy Minimisation Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science, Vol. 10746, 249–262, Springer, Cham, 2018 (cited on p. 74).

[159] D. Greenfeld, M. Galun, R. Kimmel, I. Yavneh and R. Basri: Learning to optimize multigrid PDE solvers, *Proc. 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov (Ed.), Vol. 97, Proceedings of Machine Learning Research, 2415–2423, Long Beach, CA, June 2019 (cited on pp. 131, 143).

[160] S. Grewenig, J. Weickert and A. Bruhn: From box filtering to fast explicit diffusion, In M. Goesele, S. Roth, A. Kuijper, B. Schiele and K. Schindler (Eds.): *Pattern Recognition*, Lecture Notes in Computer Science, Vol. 6376, 533–542, Springer, Berlin, 2010 (cited on p. 33).

[161] S. Gu and R. Timofte: A brief review of image denoising algorithms and beyond, In S. Escalera, S. Ayache, J. Wan, M. Madadi, U. Güçlü and X. Baró (Eds.): *Inpainting and Denoising Challenges*, The Springer Series on Challenges in Machine Learning, 1–21, Springer, Cham, 2019 (cited on p. 65).

[162] C. Guillemot and O. Le Meur: Image inpainting: overview and recent advances, *IEEE Signal Processing Magazine*, Vol. 31, No. 1, 127–144, 2014 (cited on p. 67).

[163] S. Günther, L. Ruthotto, J. B. Schroder, E. C. Cyr and N. R. Gauger: Layer-parallel training of deep residual neural networks, *SIAM Journal on Mathematics of Data Science*, Vol. 2, No. 1, 1–23, Feb. 2020 (cited on p. 131).

[164] J. Gusak, A. Katrutsa, T. Daulbaev, A. Cichocki and I. Oseledets: Meta-solver for neural ordinary differential equations, *arXiv:2103.08561v1 [cs.LG]*, Mar. 2021 (cited on pp. 49, 62).

[165] A. Haar: Zur Theorie der orthogonalen Funktionensysteme, *Mathematische Annalen*, Vol. 69, 331–371, 1910 (cited on pp. 2, 15, 39).

[166] E. Haber, K. Lensink, E. Treister and L. Ruthotto: IMEXnet — a forward stable deep neural network, *Proc. 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov (Ed.), Vol. 97, Proceedings of Machine Learning Research, 2525–2534, Long Beach, CA, June 2019 (cited on pp. 62, 130).

[167] E. Haber and L. Ruthotto: Stable architectures for deep neural networks, *Inverse Problems*, Vol. 34, No. 1, Article no. 014004, Dec. 2017 (cited on pp. 1, 2, 49, 62, 114, 130).

[168] W. Hackbusch: Multigrid Methods and Applications, Springer, New York, 1985 (cited on pp. 35, 142).

[169] J. Hadamard: Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées, *Mémoires presentés par divers savants étrangers à l'Académie des Sciences de l'Institut de France*, Vol. 33, No. 4, 1908 (cited on p. 11).

[170] D. Hafner, P. Ochs, J. Weickert, M. Reißel and S. Grewenig: FSI schemes: fast semi-iterative solvers for PDEs and optimisation methods, In B. Rosenhahn and B. Andres (Eds.): *Pattern Recognition*, Lecture Notes in Computer Science, Vol. 9796, 91–102, Springer, Cham, 2016 (cited on pp. 33, 34, 105, 129, 130, 138, 139).

[171] D. Hartmann, C. Lessig, N. Margenberg and T. Richter: A neural network multigrid solver for the Navier-Stokes equations, *Journal of Computational Physics*, Vol. 460, Article No. 110983, July 2022 (cited on p. 131).

[172] J. He and J. Xu: MgNet: a unified framework of multigrid and convolutional neural network, *Science China Mathematics*, Vol. 62, 1331–1354, May 2019 (cited on p. 131).

[173] K. He, X. Zhang, S. Ren and J. Sun: Deep residual learning for image recognition, *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 770–778, Las Vegas, NV, June 2016, IEEE Computer Society Press (cited on pp. 48, 58, 60, 116, 136, 137, 148).

[174] D. O. Hebb: The Organization of Behavior, Wiley, New York, 1949 (cited on pp. 57, 58).

[175] Y. Hel-Or and D. Shaked: A discriminative approach for wavelet denoising, *IEEE Transactions on Image Processing*, Vol. 17, No. 4, 443–457, Mar. 2008 (cited on pp. 40, 74, 86).

[176] L. Helminger, M. Bernasconi, A. Djelouah, M. Gross and C. Schroers: Generic image restoration with flow based priors, *Proc. 2021 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 334–343, Virtual, June 2021, IEEE Computer Society Press (cited on pp. 67, 209).

[177] G. Hinton: Neural networks for machine learning: lecture 6, Available at `https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf`, last visited March 30, 2022 (cited on p. 56).

[178] G. E. Hinton: Training products of experts by minimizing contrastive divergence, *Neural Computation*, Vol. 14, No. 8, 1771–1800, Aug. 2002 (cited on p. 63).

[179] G. E. Hinton, S. Osindero and Y.-W. Teh: A fast learning algorithm for deep belief nets, *Science*, Vol. 18, No. 7, 1527–1554, July 2006 (cited on p. 59).

[180] G. E. Hinton and R. Salakhutdinov: Reducing the dimensionality of data with neural networks, *Science*, Vol. 313, No. 5786, 504–507, July 2006 (cited on pp. 58, 59).

[181] S. Hochreiter: Untersuchungen zu dynamischen neuronalen Netzen, *Diploma Thesis, Institut für Informatik, Technische Universität München, Germany*, 1991 (cited on pp. 49, 53, 58, 59).

[182] S. Hochreiter and J. Schmidhuber: Long short-term memory, *Neural Computation*, Vol. 9, No. 8, 1735–1780, Nov. 1997 (cited on pp. 47, 58, 59).

[183] L. Hoeltgen, M. Mainberger, S. Hoffmann, J. Weickert, C. H. Tang, S. Setzer, D. Johannsen, F. Neumann and B. Doerr: Optimising spatial and tonal data for PDE-based inpainting, In M. Bergounioux, G. Peyré, C. Schnörr, J.-P. Caillau and T. Haberkorn (Eds.): *Variational Methods in Imaging and Geometric Control*, Radon Series on Computational and Applied Mathematics, Vol. 18, 35–83, De Gruyter, Berlin, 2017 (cited on pp. 68, 191, 208, 240).

[184] L. Hoeltgen, S. Setzer and J. Weickert: An optimal control approach to find sparse data for Laplace interpolation, In A. Heyden, F. Kahl, C. Olsson, M. Oskarsson and X.-C. Tai (Eds.): *Energy Minimisation Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science, Vol. 8081, 151–164, Springer, Berlin, 2013 (cited on p. 68).

[185] L. Hoeltgen and J. Weickert: Why does non-binary mask optimisation work for diffusion-based image compression?, In X.-C. Tai, E. Bae, T. F. Chan, S. Y. Leung and M. Lysaker (Eds.): *Energy Minimisation Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science, Vol. 8932, 85–98, Springer, Berlin, 2015 (cited on p. 210).

[186] S. Hoffmann, M. Mainberger, J. Weickert and M. Puhl: Compression of depth maps with segment-based homogeneous diffusion, In A. Kuijper, K. Bredies, T. Pock and H. Bischof (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 7893, 319–330, Springer, Berlin, 2013 (cited on p. 67).

[187] M. Holschneider, R. Kronland-Martinet, J. Morlet and P. Tchamitchian: A real-time algorithm for signal analysis with the help of the wavelet transform, In J. M. Combes, A. Grossman and P. Tchamitchian (Eds.): *Wavelets: Time-Frequency Methods and Phase Space*, 286–297, Springer, Berlin, 1989 (cited on pp. 41, 76).

[188] J. J. Hopfield: Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, Vol. 79, No. 8, 2554–2558, Apr. 1982 (cited on pp. 47, 58, 59, 141).

[189] B. Horn and B. Schunck: Determining optical flow, *Artificial Intelligence*, Vol. 17, 185–203, 1981 (cited on p. 37).

[190] K. Hornik, M. Stinchcombe and H. White: Multilayer feedforward networks are universal approximators, *Neural Networks*, Vol. 2, No. 5, 359–366, 1989 (cited on pp. 64, 123).

[191] G. Huang, Z. Liu, L. van der Maaten and K. Q. Weinberger: Densely connected convolutional networks, *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708, Honolulu, HI, July 2017, IEEE Computer Society Press (cited on pp. 58, 60, 137, 141).

[192] D. H. Hubel and T. N. Wiesel: Receptive fields of single neurones in the cat's striate cortex, *Journal of Physiology*, Vol. 148, No. 3, 574–591, Oct. 1959 (cited on pp. 1, 59).

[193] P. J. Huber: Robust regression: asymptotics, conjectures and Monte Carlo, *The Annals of Statistics*, Vol. 1, No. 5, 799–821, Sept. 1973 (cited on p. 123).

[194] T. Iijima: Basic theory of pattern observation, In*Papers of Technical Group on Automata and Automatic Control*, In Japanese: IECE, Japan, 1959 (cited on p. 18).

[195] T. Iijima: Basic theory on normalization of pattern (in case of typical one-dimensional pattern), *Bulletin of the Electrotechnical Laboratory*, Vol. 26, In Japanese, 368–388, 1962 (cited on pp. 2, 15, 18, 115, 123, 150, 188, 210).

[196] T. Iijima: Basic theory on normalization of two-dimensional visual pattern, *Studies on Information and Control (IECE, Japan)*, No. 1, Pattern Recognition Issue. In Japanese, 15–22, 1963 (cited on p. 18).

[197] S. Iizuka, E. Simo-Serra and H. Ishikawa: Globally and locally consistent image completion, *ACM Transactions on Graphics*, Vol. 36, No. 4, Article No. 107, July 2017 (cited on pp. 67, 209).

[198] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy and T. Brox: FlowNet 2.0: evolution of optical flow estimation with deep networks, *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2462–2470, Honolulu, HI, July 2017, IEEE Computer Society Press (cited on p. 48).

[199] S. Ioffe and C. Szegedy: Batch normalization: accelerating deep network training by reducing internal covariate shift, *Proc. 32nd International Conference on Machine Learning*, F. Bach and D. Blei (Ed.), Vol. 37, Proceedings of Machine Learning Research, 448–456, Lille, France, July 2015 (cited on p. 54).

[200] A. G. Ivakhnenko: The group method of data handling — a rival of the method of stochastic approximation, *Soviet Automatic Control*, Vol. 13, No. 3, 43–55, 1968 (cited on pp. 57, 58).

[201] E. Jenner and M. Weiler: Steerable partial differential operators for equivariant neural networks, *Proc. 10th International Conference on Learning Representations*, Virtual, Apr. 2022 (cited on p. 176).

[202] Y. Jia, C. Huang and T. Darrell: Beyond spatial pyramids: receptive field learning for pooled image features, *Proc. 2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3370–3377, Providence, RI, July 2012, IEEE Computer Society Press (cited on p. 49).

[203] F. Jost, P. Peter and J. Weickert: Compressing flow fields with edge-aware homogeneous diffusion inpainting, *Proc. 2020 International Conference on Acoustics, Speech, and Signal Processing*, 2198–2202, Barcelona, Spain, May 2020, IEEE Computer Society Press (cited on p. 67).

[204] F. Jost, P. Peter and J. Weickert: Compressing piecewise smooth images with the Mumford–Shah cartoon model, *Proc. 28th European Signal Processing Conference*, 511–515, Amsterdam, Netherlands, Jan. 2021 (cited on p. 67).

[205] I. Jumakulyyev and T. Schultz: Fourth-order anisotropic diffusion for inpainting and image compression, In E. Özarslan, T. Schultz, E. Zhang and A. Fuster (Eds.): *Anisotropy Across Fields and Scales*, Mathematics and Visualization, 99–124, Springer, Cham, 2021 (cited on pp. 22, 24).

[206] J. Jumper et al.: Highly accurate protein structure prediction with AlphaFold, *Nature*, Vol. 596, 583–589, July 2021 (cited on pp. 58, 61).

[207] N. Kämper and J. Weickert: Domain decomposition algorithms for real-time homogeneous diffusion inpainting in 4K, *Proc. 2022 IEEE International Conference on Acoustics, Speech and Signal Processing*, to appear, Singapore, May 2022, IEEE Computer Society Press (cited on p. 67).

[208] L. Karos, P. Bheed, P. Peter and J. Weickert: Optimising data for exemplar-based inpainting, In J. Blanc-Talon, D. Helbert, W. Philips, D. Popescu and P. Scheunders (Eds.): *Advanced Concepts for Intelligent Vision Systems*, Lecture Notes in Computer Science, Vol. 11182, 547–558, Springer, Cham, 2018 (cited on pp. 67, 68, 208).

[209] A. Katrutsa, T. Daulbaev and I. Oseledets: Black-box learning of multigrid parameters, *Journal of Computational and Applied Mathematics*, Vol. 368, Article No. 112524, Apr. 2020 (cited on pp. 49, 131, 143).

[210] S. L. Keeling and R. Stollberger: Nonlinear anisotropic diffusion filters for wide range edge sharpening, *Inverse Problems*, Vol. 18, 175–190, Jan. 2002 (cited on p. 123).

[211] S. Kichenassamy: The Perona–Malik paradox, *SIAM Journal on Applied Mathematics*, Vol. 57, 1343–1372, 1997 (cited on p. 21).

[212] D. P. Kingma and J. Ba: Adam: A method for stochastic optimization, *Proc. 3rd International Conference on Learning Representations*, San Diego, CA, May 2015 (cited on pp. 56, 58, 60, 149, 157, 179, 213).

[213] D. P. Kingma and M. Welling: An introduction to variational autoencoders, *Foundations and Trends in Machine Learning*, No. 4, 307–392, Dec. 2018 (cited on p. 60).

[214] I. Kligvasser, T. R. Shaham and T. Michaeli: XUnit: learning a spatial activation function for efficient image restoration, *Proc. 2018 IEEE Conference on Computer Vision and Pattern Recognition*, 2433–2442, Salt Lake City, UT, June 2018, IEEE Computer Society Press (cited on p. 114).

[215] H. Knutsson and C.-F. Westin: Normalized and differential convolution, *Proc. 1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 515–523, New York, NY, June 1993, IEEE Computer Society Press (cited on p. 186).

[216] E. Kobler, A. Effland, K. Kunisch and T. Pock: Total deep variation for linear inverse problems, *Proc. 2020 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 7549–7558, Seattle, WA, June 2020, IEEE Computer Society Press (cited on p. 63).

[217] E. Kobler, T. Klatzer, K. Hammernik and T. Pock: Variational networks: connecting variational methods and deep learning, In V. Roth and T. Vetter (Eds.): *Pattern Recognition*, Lecture Notes in Computer Science, Vol. 10496, 281–293, Springer, Cham, 2017 (cited on pp. 48, 63, 120).

[218] M. J. Kochenderfer and T. A. Wheeler: Algorithms for Optimization, MIT Press, Cambridge, MA, 2019 (cited on pp. 54, 56).

[219] H. Köstler, M. Stürmer, C. Freundl and U. Rüde: PDE based video compression in real time, Technical Report No. 07-11, Lehrstuhl für Informatik 10, Univ. Erlangen–Nürnberg, Germany, 2007 (cited on pp. 67, 142).

[220] A. Krizhevsky, I. Sutskever and G. E. Hinton: Imagenet classification with deep convolutional neural networks, *Proc. 26th International Conference on Neural Information Processing Systems*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger (Ed.), Vol. 25, Advances in Neural Informa-

tion Processing Systems, 1106–1114, Lake Tahoe, NV, Dec. 2012 (cited on pp. 41, 48, 58, 60).

[221] D. Laptev, N. Savinov, J. M. Buhmann and M. Pollefeys: TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks, *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 289–297, Las Vegas, NV, June 2016, IEEE Computer Society Press (cited on pp. 64, 162).

[222] Y. LeCun: Une procédure d'apprentissage pour réseau à seuil asymétrique, *Proc. Cognitiva 85*, 599–604, Paris, France, 1985 (cited on p. 59).

[223] Y. LeCun, Y. Bengio and G. Hinton: Deep learning, *Nature*, Vol. 521, 436–444, May 2015 (cited on pp. 1, 15, 41).

[224] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Back-propagation applied to handwritten zip code recognition, *Neural Computation*, Vol. 1, No. 4, 541–551, 1989 (cited on pp. 58, 59).

[225] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, Vol. 86, No. 11, 2278–2324, Nov. 1998 (cited on pp. 1, 15, 41, 44, 59).

[226] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato and F. J. Huang: A tutorial on energy-based learning, *Predicting Structured Data*, G. Bakir, T. Hofman, B. Schölkopf, A. Smola and B. Taskar (Ed.), 191–246, Cambridge, MA, 2007, MIT Press (cited on p. 63).

[227] K. Leino, Z. Wang and M. Fredrikson: Globally-robust neural networks, *Proc. 38th International Conference on Machine Learning*, Vol. 139, Proceedings of Machine Learning Research, 6212–6222, Virtual, July 2021 (cited on p. 64).

[228] R. J. LeVeque: Finite Difference Methods for Ordinary and Partial Differential Equations, SIAM, Philadelphia, 2007 (cited on p. 25).

[229] M. Li, L. He and Z. Lin: Implicit Euler skip connections: Enhancing adversarial robustness via numerical stability, *Proc. 37th International Conference on Machine Learning*, H. Daumé III and A. Singh (Ed.), Vol. 119, Proceedings of Machine Learning Research, 5874–5883, Vienna, Austria, July 2020 (cited on pp. 1, 62, 130, 137).

[230] X. Li and M. T. Orchard: Spatially adaptive image denoising under overcomplete expansion, *Proc. 2000 International Conference on Image Processing*, Vol. 3, 300–303, Vancouver, Canada, Sept. 2000 (cited on p. 74).

[231] Z. Li and Z. Shi: Deep residual learning and PDEs on manifolds, *arXiv:1708.05115v3 [cs:IT]*, Jan. 2018 (cited on pp. 63, 114).

[232] Z. Li, X. Zhang, R. Zhu, Z. Zhang and Z. Weng: Integrating data-to-data correlation into inverse distance weighting, *Computational Geosciences*, Vol. 24, No. 1, 203–216, Nov. 2019 (cited on p. 186).

[233] H. W. Lin, M. Tegmark and D. Rolnick: Why does deep and cheap learning work so well?, *Journal of Statistical Physics*, Vol. 168, No. 6, 1223–1247, Sept. 2017 (cited on p. 64).

[234] S. Linnainmaa: The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors, *Master's Thesis, Department of Computer Science, University of Helsinki, Finland*, 1970 (cited on p. 59).

[235] S. Linnainmaa: Taylor expansion of the accumulated rounding error, *BIT Numerical Mathematics*, Vol. 16, No. 2, 146–160, June 1976 (cited on p. 59).

[236] D. C. Liu and J. Nocedal: On the limited memory BFGS method for large scale optimization, *Mathematical Programming*, Vol. 45, No. 1–3, 503–528, Aug. 1989 (cited on pp. 56, 81).

[237] H. Liu, B. Jiang, Y. Xiao and C. Yang: Coherent semantic attention for image inpainting, *Proc. 2019 IEEE International Conference on Computer Vision*, 4170–4179, Seoul, Korea, Oct. 2017, IEEE Computer Society Press (cited on pp. 67, 209).

[238] Z. Long, Y. Lu and B. Dong: PDE-net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network, *Journal of Computational Physics*, Vol. 399, No. 2197, Article no. 108925, Dec. 2019 (cited on pp. 2, 62, 92).

[239] J. M. Lorenzi, T. Stecher, K. Reuter and S. Matera: Local-metrics error-based Shepard interpolation as surrogate for highly nonlinear material models in high dimensions, *Journal of Chemical Physics*, Vol. 147, No. 16, Article no. 164106, Oct. 2017 (cited on p. 186).

[240] Y. Lu, A. Zhong, Q. Li and B. Dong: Beyond finite layer neural networks: bridging deep architectures and numerical differential equations, *Proc. 35th International Conference on Machine Learning*, J. Dy and A. Krause (Ed.), Vol. 80, Proceedings of Machine Learning Research, 3276–3285, Stockholm, Sweden, July 2018 (cited on pp. 62, 114, 130, 137, 139, 151).

[241] Z. Luo, Z. Sun, W. Zhou, Z. Wu and S. Kamata: Rethinking ResNets: improved stacking strategies with high-order schemes for image classification, *Complex & Intelligent Systems*, Vol. 8, 3395–3407, Feb. 2022 (cited on p. 62).

[242] M. Lysaker, A. Lundervold and X.-C. Tai: Noise removal using fourth-order partial differential equations with applications to medical magnetic resonance images in space and time, *IEEE Transactions on Image Processing*, Vol. 12, No. 12, 1579–1590, Dec. 2003 (cited on pp. 22, 165, 167, 177).

[243] J. Ma and G. Plonka: Combined curvelet shrinkage and nonlinear anisotropic diffusion, *IEEE Transactions on Image Processing*, Vol. 16, No. 9, 2198–2206, Aug. 2007 (cited on p. 119).

[244] M. Mahoney: Adaptive weighing of context models for lossless data compression, Technical Report No. CS-2005-16, Florida Institute of Technology, Melbourne, Florida, Dec. 2005 (cited on p. 198).

[245] M. Mainberger, A. Bruhn, J. Weickert and S. Forchhammer: Edge-based compression of cartoon-like images with homogeneous diffusion, *Pattern Recognition*, Vol. 44, No. 9, 1859–1873, Sept. 2011 (cited on p. 67).

[246] M. Mainberger, S. Hoffmann, J. Weickert, C. H. Tang, D. Johannsen, F. Neumann and B. Doerr: Optimising spatial and tonal data for homogeneous diffusion inpainting, In A. M. Bruckstein, B. ter Haar Romeny, A. M. Bronstein and M. M. Bronstein (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 6667, 26–37, Springer, Berlin, 2012 (cited on pp. 68, 107, 142, 207, 208, 210, 215, 240).

[247] F. Malgouyres and F. Guichard: Edge direction preserving image zooming: a mathematical and numerical analysis, *SIAM Journal on Numerical Analysis*, Vol. 39, No. 1, 1–37, 2001 (cited on p. 67).

[248] S. Mallat: A Wavelet Tour of Signal Processing, Second, Academic Press, San Diego, 1999 (cited on pp. 2, 15, 39, 65, 74, 77, 116, 123).

[249] D. Marcos, M. Volpi and D. Tuia: Learning rotation invariant convolutional filters for texture classification, *Proc. 23rd International Conference on Pattern Recognition*, Vol. 2, 2012–2017, Cancun, Mexico, Dec. 2016 (cited on pp. 64, 162).

[250] D. Marwood, P. Massimino, M. Covell and S. Baluja: Representing images in 200 bytes: compression via triangulation, *Proc 2018 IEEE International Conference on Image Processing*, 405–409, Athens, Greece, Oct. 2018, IEEE Computer Society Press (cited on pp. 68, 208).

[251] S. Masnou and J.-M. Morel: Level lines based disocclusion, *Proc. 1998 IEEE International Conference on Image Processing*, Vol. 3, 259–263, Chicago, IL, Oct. 1998, IEEE Computer Society Press (cited on pp. 15, 67, 239).

[252] M. T. McCann, K. H. Jin and M. Unser: Convolutional neural networks for inverse problems in imaging: a review, *IEEE Signal Processing Magazine*, Vol. 34, No. 6, 85–95, Nov. 2017 (cited on p. 63).

[253] W. S. McCulloch and W. Pitts: A logical calculus of the ideas immanent in nervous activity, *The Bulletin of Mathematical Biophysics*, Vol. 5, 115–133, 1943 (cited on pp. 1, 43, 57, 58).

[254] I. Meilijson and E. Ruppin: Optimal signalling in attractor neural networks, *Proc. 7th International Conference on Neural Information Processing Systems*, G. Tesauro, D.S. Touretzky and T.K. Leen (Ed.), Vol. 7, Advances in Neural Information Processing Systems, 485–492, Denver, CO, Dec. 1994 (cited on pp. 114, 123, 131).

[255] R. E. Mickens: Nonstandard Finite Difference Models of Differential Equations, World Scientific, Singapore, 1994 (cited on p. 29).

[256] M. L. Minsky and S. A. Papert: Perceptrons, MIT Press, Cambridge, MA, 1969 (cited on pp. 57, 58).

[257] D. Misra: Mish: A self regularized non-monotonic activation function, *Proc. 31st British Machine Vision Conference 2020*, Virtual, Sept. 2020 (cited on p. 130).

[258] R. M. K. Mohideen, P. Peter, T. Alt, J. Weickert and A. Scheer: Compressing colour images with joint inpainting and prediction, *arXiv:2010.09866 [eess.IV]*, Oct. 2020 (cited on pp. 67, 187, 190, 192, 198, 200, 203–206, 239).

[259] R. M. K. Mohideen, P. Peter and J. Weickert: A systematic evaluation of coding strategies for sparse binary images, *Signal Processing: Image Communication*, Vol. 99, Article no. 116424, Nov. 2021 (cited on p. 213).

[260] V. Monga, Y. Li and Y. C. Eldar: Algorithm unrolling: interpretable, efficient deep learning for signal and image processing, *IEEE Signal Processing Magazine*, Vol. 38, No. 2, 18–44, Feb. 2021 (cited on pp. 48, 63, 120).

[261] J.-M. Morel and S. Solimini: Segmentation of images by variational methods: a constructive approach, *Revista Matematica de la Universidad Complutense de Madrid*, Vol. 1, 169–182, 1988 (cited on p. 37).

[262] P. Mrázek and J. Weickert: Rotationally invariant wavelet shrinkage, In B. Michaelis and G. Krell (Eds.): *Pattern Recognition*, Lecture Notes in Computer Science, Vol. 2781, 156–163, Springer, Berlin, 2003 (cited on pp. 63, 162).

[263] P. Mrázek and J. Weickert: From two-dimensional nonlinear diffusion to coupled Haar wavelet shrinkage, *Journal of Visual Communication and Image Representation*, Vol. 18, No. 2, 162–175, Apr. 2007 (cited on pp. 63, 74, 79, 227, 228, 230).

[264] P. Mrázek, J. Weickert and G. Steidl: Diffusion-inspired shrinkage functions and stability results for wavelet denoising, *International Journal of Computer Vision*, Vol. 64, 171–186, Sept. 2005 (cited on pp. 41, 63, 114, 119).

[265] D. Mumford and J. Shah: Optimal approximation of piecewise smooth functions and associated variational problems, *Communications on Pure and Applied Mathematics*, Vol. 42, 577–685, 1989 (cited on p. 37).

[266] V. Nair and G. E. Hinton: Rectified linear units improve restricted Boltzmann machines, *Proc. 27th International Conference on Machine Learning*, J. Fürnkranz and T. Joachims (Ed.), 807–814, Haifa, Israel, June 2010 (cited on pp. 43, 58, 60, 122, 123, 148).

[267] Y. Nesterov: A method for solving the convex programming problem with convergence rate $O(1/k^2)$, *Soviet Mathematics Doklady*, Vol. 4, 1035–1038, 1963 (cited on pp. 55, 139).

[268] A. Newell, K. Yang and J. Deng: Stacked hourglass networks for human pose estimation, In B. Leibe, J. Matas, N. Sebe and M. Welling (Eds.): *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, Vol. 9912, 483–499, Springer, Cham, 2016 (cited on pp. 49, 51, 147, 158).

[269] W. J. Niessen, B. M. ter Haar Romeny, L. M. Florack and M. A. Viergever: A general framework for geometry-driven evolution equations, *International Journal of Computer Vision*, Vol. 21, No. 3, 187–205, Feb. 1997 (cited on pp. 21, 92).

[270] J. Nocedal and S. J. Wright: Numerical Optimization, Springer, New York, 2006 (cited on pp. 54, 56).

[271] P. Ochs, Y. Chen, T. Brox and T. Pock: IPiano: inertial proximal algorithm for nonconvex optimization, *SIAM Journal on Imaging Sciences*, Vol. 7, 1388–1419, 2014 (cited on pp. 68, 208).

[272] P. Ochs, T. Meinhardt, L. Leal-Taixe and M. Möller: Lifting layers: analysis and applications, In V. Ferrari, M. Herbert, C. Sminchisescu and Y. Weiss (Eds.): *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, Vol. 11205, 53–68, Springer, Cham, 2018 (cited on pp. 130, 176).

[273] K.-S. Oh and K. Jung: GPU implementation of neural networks, *Pattern Recognition*, Vol. 37, No. 6, 1311–1314, June 2004 (cited on p. 59).

[274] K. Ott, P. Katiyar, P. Hennig and M. Tiemann: ResNet after all? Neural ODEs and their numerical solution, *Proc. 9th International Conference on Learning Representations*, Vienna, Austria, May 2021 (cited on pp. 49, 62).

[275] S. Ouala, A. Pascual and R. Fablet: Residual integration neural network, *Proc. 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, 3622–3626, Brighton, UK, May 2019, IEEE Computer Society Press (cited on pp. 130, 137, 139).

[276] R. Parhi and R. D. Nowak: Banach space representer theorems for neural networks and ridge splines, *Journal of Machine Learning Research*, Vol. 22, 1–40, Feb. 2021 (cited on p. 64).

[277] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell and A. A. Efros: Context encoders: feature learning by inpainting, *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2536–2544, Las Vegas, NV, June 2016, IEEE Computer Society Press (cited on pp. 67, 209).

[278] W. B. Pennebaker and J. L. Mitchell: JPEG: Still Image Data Compression Standard, Springer, New York, 1992 (cited on pp. 203, 239).

[279] P. Perona and J. Malik: Scale space and edge detection using anisotropic diffusion, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, 629–639, July 1990 (cited on pp. 2, 15, 18, 19, 23, 65, 80, 92, 93, 104, 115, 123, 132, 148, 164, 177, 178, 188).

[280] P. Peter: Fast inpainting-based compression: combining shepard interpolation with joint inpainting and prediction, *Proc. 26th IEEE International Conference on Image Processing*, 3557–3561, Taipei, Taiwan, Sept. 2019, IEEE Computer Society Press (cited on pp. 67, 185–187, 191, 192, 198, 205, 239).

[281] P. Peter, S. Hoffmann, F. Nedwed, L. Hoeltgen and J. Weickert: Evaluating the true potential of diffusion-based inpainting in a compression context, *Signal Processing: Image Communication*, Vol. 46, 40–53, Aug. 2016 (cited on p. 190).

[282] P. Peter, L. Kaufhold and J. Weickert: Turning diffusion-based image colorization into efficient color compression, *IEEE Transactions on Image Processing*, Vol. 26, No. 2, 860–869, Feb. 2017 (cited on pp. 21, 22, 24, 67).

[283] P. Peter, K. Schrader, T. Alt and J. Weickert: Deep spatial and tonal optimisation for homogeneous diffusion inpainting, *arXiv:2208.14371v2 [cs.LG]*, Sept. 2022 (cited on p. 240).

[284] P. Peter and J. Weickert: Justifying tensor-driven diffusion from structure-adaptive statistics of natural images, In X.-C. Tai, E. Bae, T. F. Chan, S. Y. Leung and M. Lysaker (Eds.): *Energy Minimisation Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science, Vol. 8932, 263–277, Springer, Berlin, 2015 (cited on p. 95).

[285] T. Pinetz, E. Kobler, T. Pock and A. Effland: Shared prior learning of energy-based models for image reconstruction, *SIAM Journal on Imaging Sciences*, Vol. 14, No. 4, 1706–1748, Nov. 2021 (cited on p. 63).

[286] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda and Q. Liao: Why and when can deep - but not shallow - networks avoid the curse of dimensionality: a review, *International Journal of Automation and Computing*, Vol. 14, No. 5, 503–519, Oct. 2017 (cited on p. 64).

[287] B. T. Polyak: Some methods of speeding up the convergence of iteration methods, *USSR Computational Mathematics and Mathematical Physics*, Vol. 4, No. 5, 1–17, 1964 (cited on pp. 55, 139).

[288] J. Portilla, V. Strela, M. J. Wainwright and E. P. Simoncelli: Image denoising using scale mixtures of Gaussians in the wavelet domain, *IEEE Transactions on Image Processing*, Vol. 12, No. 11, 1338–1351, Nov. 2003 (cited on p. 74).

[289] A. Poulenard, M.-J. Rakotosaona, Y. Ponty and M. Ovsjanikov: Effective rotation-invariant point CNN with spherical harmonics kernels, *Proc. 2019 International Conference on 3D Vision*, 47–56, Quebec City, Canada, Sept. 2019, IEEE Computer Society Press (cited on p. 162).

[290] J. Prost, A. Houdard, A. Almansa and N. Papadakis: Learning local regularization for variational image restoration, In A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin and L. Simon (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 12679, 358–370, Springer, Cham, 2021 (cited on p. 63).

[291] C. Rackauckas, Y. Ma, J. Martensen, C. Warnter, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan and A. Edelman: Universal differential equations for scientific machine learning, *arXiv:2001.04385v3 [cs.LG]*, Aug. 2020 (cited on pp. 62, 92).

[292] E. Radmoser, O. Scherzer and J. Weickert: Scale-space properties of nonstationary iterative regularization methods, *Journal of Visual Communication and Image Representation*, Vol. 11, No. 2, 96–114, June 2000 (cited on p. 120).

[293] M. Raissi, P. Perdikaris and G. E. Karniadakis: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics*, Vol. 378, 686–707, Feb. 2019 (cited on p. 62).

[294] P. Ramachandran, B. Zoph and Q. V. Le: Searching for activation functions, *arXiv:1710.05941v2 [cs.NE]*, Oct. 2017 (cited on p. 130).

[295] Z. Ramzi, J.-L. Starck, T. Moreau and P. Ciuciu: Wavelets in the deep learning era, *Proc. 28th European Signal Processing Conference*, 1417–1421, Amsterdam, Netherlands, Jan. 2021 (cited on p. 63).

[296] M. Ranzato, C. S. Poultney, S. Chopra and Y. LeCun: Efficient learning of sparse representations with an energy-based model, *Proc. 20th International Conference on Neural Information Processing Systems*, B. Schölkopf, J. C. Platt and T. Hofmann (Ed.), Vol. 19, Advances in Neural Information Processing Systems, 1137–1144, Vancouver, Canada, Dec. 2006, MIT Press (cited on pp. 58, 59).

[297] V. Ratner and Y. Y. Zeevi: The dynamics of image processing viewed as damped elastic deformation, *Proc. 17th European Signal Processing Conference*, 45–49, Glasgow, UK, Aug. 2009, IEEE Computer Society Press (cited on p. 138).

[298] J. S. J. Ren, L. Xu and W. Sun: Shepard convolutional neural networks, *Proc. 29th International Conference on Neural Information Processing Systems*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett (Ed.), Vol. 28, Advances in Neural Information Processing Systems, 901–909, Montréal, Canada, Dec. 2015 (cited on p. 67).

[299] R. J. Renka: Multivariate interpolation of large sets of scattered data, *ACM Transactions on Mathematical Software*, Vol. 14, No. 2, 139–148, June 1988 (cited on p. 186).

[300] E. Ringaby, O. Friman and. P.-E. Forssén, T. O. Opsahl, T. V. Haavardsholm and I. Kasen: Anisotropic scattered data interpolation for pushbroom image rectification, *IEEE Transactions on Image Processing*, Vol. 23, No. 5, 2302–2314, Apr. 2014 (cited on p. 186).

[301] H. Robbins and S. Monro: A stochastic approximation method, *Annals of Mathematical Statistics*, Vol. 22, No. 3, 400–407, 1951 (cited on p. 55).

[302] M. X. B. Rodriguez, A. Gruson, L. F. Polanía, S. Fujieda, F. P. Ortiz, K. Takayama and T. Hachisuka: Deep adaptive wavelet network, *Proc. IEEE Winter Conference on Applications of Computer Vision*, Snowmass, CO, Mar. 2020 (cited on p. 63).

[303] D. Rolnick and M. Tegmark: The power of deeper networks for expressing natural functions, *Proc. 6th International Conference on Learning Representations*, Vancouver, Canada, Apr. 2018 (cited on p. 2).

[304] Y. Romano, A. Aberdam, J. Sulam and M. Elad: Adversarial noise attacks of deep learning architectures: stability analysis via sparse-modeled signals, *Journal of Mathematical Imaging and Vision*, Vol. 62, 313–327, Apr. 2020 (cited on pp. 1, 114).

[305] Y. Romano, M. Elad and P. Milanfar: The little engine that could: regularization by denoising (RED), *SIAM Journal on Imaging Sciences*, Vol. 10, No. 4, 1804–1844, Oct. 2017 (cited on p. 63).

[306] O. Ronneberger, P. Fischer and T. Brox: U-net: convolutional networks for biomedical image segmentation, In N. Navab, J. Hornegger, W. Wells and A. Frangi (Eds.): *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, Vol. 9351, 234–241, Springer, Cham, 2015 (cited on pp. 48, 49, 51, 58, 60, 130, 131, 142, 147, 210).

[307] F. Rosenblatt: The perceptron: a probabilistic model for information storage and organization in the brain, *Psychological Review*, Vol. 65, No. 6, 386–408, 1958 (cited on pp. 43, 57, 58).

[308] S. Roth and M. J. Black: Fields of experts: a framework for learning image priors, *Proc. 2005 IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, 860–867, San Diego, CA, June 2005, IEEE Computer Society Press (cited on pp. 65, 82).

[309] F. Rousseau, L. Drumetz and R. Fablet: Residual networks as flows of diffeomorphisms, *Journal of Mathematical Imaging and Vision*, Vol. 62, 365–375, Apr. 2020 (cited on pp. 114, 130, 135).

[310] A. Roussos and P. Maragos: Vector-valued image interpolation by an anisotropic diffusion-projection PDE, In F. Sgallari, F. Murli and N. Paragios (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 4485, 104–115, Springer, Berlin, 2007 (cited on p. 94).

[311] A. Roussos and P. Maragos: Tensor-based image diffusions derived from generalizations of the total variation and Beltrami functionals, *Proc. 2010 IEEE International Conference on Image Processing*, 4141–4144, Hong Kong, Sept. 2010, IEEE Computer Society Press (cited on p. 24).

[312] S. Ruder: An overview of gradient descent optimization algorithms, *arXiv:1609.04747v2 [cs.LG]*, June 2017 (cited on pp. 54, 56).

[313] L. I. Rudin, S. Osher and E. Fatemi: Nonlinear total variation based noise removal algorithms, *Physica D*, Vol. 60, No. 1–4, 259–268, Nov. 1992 (cited on pp. 19, 37, 65, 123).

[314] S. H. Rudy, S. L. Brunton, J. L. Proctor and J. N. Kutz: Data-driven discovery of partial differential equations, *Science Advances*, Vol. 3, No. 4, Article no. e1602614, 2017 (cited on p. 62).

[315] D. Ruiz-Balet and E. Zuazua: Neural ODE control for classification, approximation and transport, *arXiv:2104.05278v1 [math.OC]*, Apr. 2021 (cited on p. 62).

[316] D. E. Rumelhart, G. E. Hinton and R. J. Williams: Learning representations by back-propagating errors, *Nature*, Vol. 323, No. 9, 533–536, Oct. 1986 (cited on pp. 47, 51, 58, 59).

[317] D. E. Rumelhart and J. L. McClelland: Parallel distributed processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge, MA, 1986 (cited on p. 139).

[318] O. Russakovsky et al.: ImageNet large scale visual recognition challenge, *International Journal of Computer Vision*, Vol. 115, 211–252, Apr. 2015 (cited on pp. 60, 157).

[319] L. Ruthotto and E. Haber: Deep neural networks motivated by partial differential equations, *Journal of Mathematical Imaging and Vision*, Vol. 62, 352–364, Apr. 2020 (cited on pp. 2, 49, 62, 114, 120, 130, 134–136, 151, 166).

[320] Y. Saad: Iterative Methods for Sparse Linear Systems, Second, SIAM, Philadelphia, 2003 (cited on pp. 34, 142).

[321] M. Sagong, Y. Shin, S. Kim, S. Park and S. Ko: PEPSI: fast image inpainting with parallel decoding network, *Proc. 2019 IEEE Conference on Computer Vision and Pattern Recognition*, 11360–11368, Long Beach, CA, June 2019 (cited on pp. 67, 209).

[322] H. Sak, A. W. Senior and F. Beaufays: Long short-term memory recurrent neural network architectures for large scale acoustic modeling, *Proc. 15th Annual Conference of the International Speech Communication Association*, H. Li, H. M. Meng, B. Ma, E. Chng and L. Xie (Ed.), Singapore, Sept. 2014 (cited on p. 48).

[323] T. Salimans and D. P. Kingma: Weight normalization: a simple reparameterization to accelerate training of deep neural networks, *Proc. 30th International Conference on Neural Information Processing Systems*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon and R. Garnett (Ed.), Vol. 29, Advances in Neural Information Processing Systems, 901–909, Barcelona, Spain, Dec. 2016 (cited on p. 137).

[324] H. Schaeffer: Learning partial differential equations via data discovery and sparse optimization, *Proceedings of the Royal Society of London, Series A*, Vol. 473, No. 2197, Article no. 20160446, Jan. 2017 (cited on pp. 62, 92).

[325] H. Scharr, M. J. Black and H. W. Haussecker: Image statistics and anisotropic diffusion, *Proc. Ninth International Conference on Computer Vision*, Vol. 1, 840–847, Nice, France, Oct. 2003, IEEE Computer Society Press (cited on p. 24).

[326] O. Scherzer and J. Weickert: Relations between regularization and diffusion filtering, *Journal of Mathematical Imaging and Vision*, Vol. 12, No. 1, 43–63, Feb. 2000 (cited on pp. 11, 21, 37, 92, 114, 119, 132, 164).

[327] C. Schmaltz, P. Peter, M. Mainberger, F. Ebel, J. Weickert and A. Bruhn: Understanding, optimising, and extending data compression with anisotropic diffusion, *International Journal of Computer Vision*, Vol. 108, No. 3, 222–240, July 2014 (cited on pp. 15, 67, 187, 207, 215, 239).

[328] C. Schmaltz and J. Weickert: Video compression with 3-D pose tracking, PDE-based image coding, and electrostatic halftoning, In A. Pinz, T. Pock, H. Bischof and F. Leberl (Eds.): *Pattern Recognition*, Lecture Notes in Computer Science, Vol. 7476, 438–447, Springer, Berlin, 2012 (cited on p. 67).

[329] C. Schmaltz, J. Weickert and A. Bruhn: Beating the quality of JPEG 2000 with anisotropic diffusion, In J. Denzler, G. Notni and H. Süße (Eds.): *Pattern Recognition*, Lecture Notes in Computer Science, Vol. 5748, 452–461, Springer, Berlin, 2009 (cited on pp. 15, 67).

[330] J. Schmidhuber: Deep learning in neural networks: an overview, *Neural Networks*, Vol. 61, 85–117, Jan. 2015 (cited on pp. 1, 15, 41, 57, 60).

[331] U. Schmidt and S. Roth: Shrinkage fields for effective image restoration, *Proc. 2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2774–2781, Columbus, OH, June 2014, IEEE Computer Society Press (cited on pp. 65, 74, 92).

[332] I. J. Schoenberg: Über variationsvermindernde lineare Transformationen, *Mathematische Zeitschrift*, Vol. 32, 321–328, 1930 (cited on p. 119).

[333] K. Schrader, T. Alt, J. Weickert and M. Ertel: CNN-based Euler's elastica inpainting with deep energy and deep image prior, *Proc. 10th European Workshop on Visual Information Processing*, Lisbon, Portugal, July 2022, IEEE Computer Society Press (cited on p. 239).

[334] J. Shen and T. F. Chan: Mathematical models for local nontexture inpaintings, *SIAM Journal on Applied Mathematics*, Vol. 62, No. 3, 1019–1043, 2002 (cited on p. 67).

[335] J. Shen, S. H. Kang and T. F. Chan: Euler's elastica and curvature-based inpainting, *SIAM Journal on Applied Mathematics*, Vol. 63, No. 2, 564–592, 2002 (cited on p. 67).

[336] D. Shepard: A two-dimensional interpolation function for irregularly-spaced data, *Proc. 23rd ACM National Conference*, 517–524, Las Vegas, NV, Aug. 1968 (cited on pp. 67, 185–187).

[337] L. Sifre and S. Mallat: Rotation, scaling and deformation invariant scattering for texture discrimination, *Proc. 2013 IEEE Conference on Computer Vision and Pattern Recognition*, 1233–1240, Portland, OR, June 2013, IEEE Computer Society Press (cited on pp. 64, 162).

[338] D. Silver et al.: Mastering the game of Go without human knowledge, *Nature*, Vol. 550, 354–359, Oct. 2017 (cited on pp. 58, 60).

[339] P. Y. Simard, D. Steinkraus and J. C. Platt: Best practices for convolutional neural networks applied to visual document analysis, *Proc. 7th International Conference on Document Analysis and Recognition*, 958–963, Edinburgh, Aug. 2003 (cited on pp. 64, 162).

[340] K. Simonyan and A. Zisserman: Very deep convolutional networks for large-scale image recognition, *Proc. 3rd International Conference on Learning Representations*, Y. Bengio and Y. LeCun (Ed.), San Diego, CA, May 2015 (cited on p. 60).

[341] B. Smolka: Combined forward and backward anisotropic diffusion filtering of color images, In L. Van Gool (Ed.): *Pattern Recognition*, Lecture Notes in Computer Science, Vol. 2449, 314–320, Springer, Berlin, 2002 (cited on p. 80).

[342] University of Southern California: USC-SIPI image database, 1977, Available at https://sipi.usc.edu/database/database.php, last visited March 30, 2022 (cited on pp. 24, 193, 214).

[343] J.-L. Starck, F. Murtagh, E. Candès and D. L. Donoho: Gray and color image contrast enhancement by the curvelet transform, *IEEE Transactions on Image Processing*, Vol. 12, No. 6, 706–717, June 2003 (cited on p. 74).

[344] G. Steidl, J. Weickert, T. Brox, P. Mrázek and M. Welk: On the equivalence of soft wavelet shrinkage, total variation diffusion, total variation regularization, and SIDEs, *SIAM Journal on Numerical Analysis*, Vol. 42, No. 2, 686–713, 2004 (cited on p. 63).

[345] J. Sulam, A. Aberdam, A. Beck and M. Elad: On multi-layer basis pursuit, efficient algorithms and convolutional neural networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 42, No. 8, 1968–1980, Aug. 2019 (cited on pp. 48, 63, 120).

[346] J. Sun and Z. Xu: Color image denoising via discriminatively learned iterative shrinkage, *IEEE Transactions on Image Processing*, Vol. 24, No. 11, 4148–4159, June 2015 (cited on p. 74).

[347] I. Sutskever, J. Martens, G. Dahl and G. Hinton: On the importance of initialization and momentum in deep learning, *Proc. 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester (Ed.), Vol. 28, Proceedings of Machine Learning Research, 1139–1147, Atlanta, GA, June 2013 (cited on pp. 49, 53, 139).

[348] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich: Going deeper with convolutions, *Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition*, 1–9, Boston, MA, June 2015, IEEE Computer Society Press (cited on pp. 48, 60).

[349] D. S. Taubman and M. W. Marcellin (Ed.): JPEG 2000: Image Compression Fundamentals, Standards and Practice, Kluwer, Boston, 2002 (cited on p. 67).

[350] Y. W. Teh, M. Welling, S. Osindero and G. E. Hinton: Energy-based models for sparse overcomplete representations, *Journal of Machine Learning Research*, Vol. 4, 1235–1260, Dec. 2003 (cited on p. 63).

[351] M. Terris, A. Repetti, J.-C. Pesquet and Y. Wiaux: Building firmly nonexpansive convolutional neural networks, *Proc. 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8658–8662, Barcelona, Spain, May 2020 (cited on p. 64).

[352] M. Thorpe and Y. van Gennip: Deep limits of residual neural networks, *Russian Mathematical Surveys*, Vol. 10, No. 1, Article no. 6, Dec. 2022 (cited on p. 114).

[353] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo and C.-W. Lin: Deep learning on image denoising: an overview, *Neural Networks*, Vol. 131, 251–275, Nov. 2020 (cited on p. 65).

[354] A. N. Tikhonov: Solution of incorrectly formulated problems and the regularization method, *Soviet Mathematics Doklady*, Vol. 4, 1035–1038, 1963 (cited on pp. 2, 15, 36, 116).

[355] C. Tomasi and R. Manduchi: Bilateral filtering for gray and color images, *Proc. Sixth International Conference on Computer Vision*, 839–846, Bombay, India, Jan. 1998, Narosa Publishing House (cited on p. 65).

[356] M. Tomczak: Spatial interpolation and its uncertainty using automated anisotropic inverse distance weighting (IDW) – cross-validation/jackknife approach, *Journal of Geographic Information and Decision Analysis*, Vol. 2, No. 2, 18–30, 1998 (cited on p. 186).

[357] C. Tretter: Spectral Theory of Block Operator Matrices and Applications, Imperial College Press, London, 2008 (cited on p. 137).

[358] D. Tschumperlé and R. Deriche: Vector-valued image regularization with PDEs: a common framework for different applications, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 4, 506–516, Apr. 2005 (cited on p. 24).

[359] A. Tuor, J. Drgona and D. Vrabie: Constrained neural ordinary differential equations with stability guarantees, *arXiv:2004.10883v1 [eess.SY]*, Apr. 2020 (cited on p. 62).

[360] D. Ulyanov, A. Vedaldi and V. Lempitsky: Deep image prior, *Proc. 2018 IEEE Conference on Computer Vision and Pattern Recognition*, 9446–9454, Salt Lake City, UT, June 2018, IEEE Computer Society Press (cited on pp. 63, 209).

[361] M. Unser: A representer theorem for deep neural networks, *Journal of Machine Learning Research*, Vol. 20, No. 110, 1–30, July 2019 (cited on p. 114).

[362] P. J. van der Houwen and B. P. Sommeijer: On the internal stability of explicit, m-stage Runge-Kutta methods for large m-values, *Zeitschrift für Angewandte Mathematik und Mechanik*, Vol. 60, No. 10, 479–485, 1980 (cited on p. 139).

[363] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin: Attention is all you need, *Proc. 31st International Conference on Neural Information Processing Systems*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Wishwanathan and R. Garnett (Ed.), Vol. 30, Advances in Neural Information Processing Systems, 5998–6008, Long Beach, CA, Dec. 2017 (cited on pp. 58, 60).

[364] D. Vašata, T. Halama and M. Friedjungová: Image inpainting using Wasserstein generative adversarial imputation network, In I. Farkaš, P. Masulli, S. Otte and S. Wermter (Eds.): *Artificial Neural Networks and Machine Learning – ICANN 2021*, Lecture Notes in Computer Science, Vol. 12892, 575–586, Springer, Cham, 2021 (cited on p. 209).

[365] R. Vidal, J. Bruna, R. Giryes and S. Soatto: Mathematics of deep learning, *arXiv:1712.04741v1 [cs.LG]*, Dec. 2017 (cited on p. 62).

[366] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli: Image quality assessment: from error visibility to structural similarity, *IEEE Transactions on Image Processing*, Vol. 13, 600–612, Apr. 2004 (cited on p. 13).

[367] J. Weickert: Anisotropic diffusion filters for image processing based quality control, In A. Fasano and M. Primicerio (Eds.): *Proc. Seventh European Conference on Mathematics in Industry*, 355–362, Teubner, Stuttgart, 1994 (cited on p. 163).

[368] J. Weickert: Theoretical foundations of anisotropic diffusion in image processing, *Computing Supplement*, Vol. 11, 221–236, 1996 (cited on pp. 22, 91–93, 101, 156, 163, 185, 210).

[369] J. Weickert: Anisotropic Diffusion in Image Processing, Teubner, Stuttgart, 1998 (cited on pp. 2, 15, 17, 23, 31, 32, 34, 65, 92, 95, 117, 134, 135, 168, 237).

[370] J. Weickert: Coherence-enhancing diffusion filtering, *International Journal of Computer Vision*, Vol. 31, No. 2/3, 111–127, Apr. 1999 (cited on p. 24).

[371] J. Weickert: Personal communication, 2018 (cited on p. 3).

[372] J. Weickert and B. Benhamouda: A semidiscrete nonlinear scale-space theory and its relation to the Perona–Malik paradox, In F. Solina, W. G. Kropatsch, R. Klette and R. Bajcsy (Eds.): *Advances in Computer Vision*, 1–10, Springer, Wien, 1997 (cited on p. 123).

[373] J. Weickert and T. Brox: Diffusion and regularization of vector- and matrix-valued images, In M. Z. Nashed and O. Scherzer (Eds.): *Inverse Problems, Image Analysis, and Medical Imaging*, Contemporary Mathematics, Vol. 313, 251–268, AMS, Providence, 2002 (cited on pp. 24, 94, 95, 169, 170, 177).

[374] J. Weickert, S. Grewenig, C. Schroers and A. Bruhn: Cyclic schemes for PDE-based image analysis, *International Journal of Computer Vision*, Vol. 118, No. 3, 275–299, July 2016 (cited on p. 33).

[375] J. Weickert, S. Ishikawa and A. Imiya: Linear scale-space has first been proposed in Japan, *Journal of Mathematical Imaging and Vision*, Vol. 10, No. 3, 237–252, May 1999 (cited on p. 18).

[376] J. Weickert and C. Schnörr: A theoretical framework for convex regularizers in PDE-based computation of image motion, *International Journal of Computer Vision*, Vol. 45, No. 3, 245–264, Dec. 2001 (cited on p. 95).

[377] J. Weickert, G. Steidl, P. Mrázek, M. Welk and T. Brox: Diffusion filters and wavelets: what can they learn from each other?, In N. Paragios, Y. Chen and O. Faugeras (Eds.): *Handbook of Mathematical Models in Computer Vision*, 3–16, Springer, New York, 2006 (cited on p. 63).

[378] J. Weickert and M. Welk: Tensor field interpolation with PDEs, In J. Weickert and H. Hagen (Eds.): *Visualization and Processing of Tensor Fields*, 315–325, Springer, Berlin, 2006 (cited on pp. 15, 24, 67, 156).

[379] J. Weickert, M. Welk and M. Wickert: L^2-stable nonstandard finite differences for anisotropic diffusion, In A. Kuijper, K. Bredies, T. Pock and H. Bischof (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 7893, 390–391, Springer, Berlin, 2013 (cited on pp. 29, 30, 32, 79, 105, 156, 164, 176, 178, 179, 227, 229, 230).

[380] M. Weiler and G. Cesa: General E(2)-equivariant steerable CNNs, *Proc. 33rd International Conference on Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox and R. Garnett (Ed.), Vol. 32, Advances in Neural Information Processing Systems, 14334–14345, Vancouver, Canada, Dec. 2019 (cited on p. 162).

[381] M. Weiler, M. Geiger, M. Welling, W. Boomsma and T. Cohen: 3D steerable CNNs: Learning rotationally equivariant features in volumetric sdata, *Proc. 32nd International Conference on Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett (Ed.), Vol. 31, Advances in Neural Information Processing Systems, 10402–10413, Montréal, Canada, Dec. 2018 (cited on p. 162).

[382] M. Weiler, F. A. Hamprecht and M. Storath: Learning steerable filters for rotation equivariant CNNs, *Proc. 2018 IEEE Conference on Computer Vision and Pattern Recognition*, 849–858, Salt Lake City, UT, June 2018, IEEE Computer Society Press (cited on pp. 64, 161–163).

[383] P. Weinzaepfel, H. Jégou and P. Pérez: Reconstructing an image from its local descriptors, *Proc. 2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 337–344, Colorado Springs, CO, June 2011, IEEE Computer Society Press (cited on p. 68).

[384] M. Welk: Diffusion, pre-smoothing and gradient descent, In A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin and L. Simon (Eds.): *Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, Vol. 12679, 78–90, Springer, Cham, 2021 (cited on pp. 21, 24, 168).

[385] M. Welk, G. Steidl and J. Weickert: Locally analytic schemes: a link between diffusion filtering and wavelet shrinkage, *Applied and Computational Harmonic Analysis*, Vol. 24, 195–224, 2008 (cited on p. 119).

[386] M. Welk, J. Weickert and G. Gilboa: A discrete theory and efficient algorithms for forward-and-backward diffusion filtering, *Journal of Mathematical Imaging and Vision*, Vol. 60, No. 9, 1399–1426, Nov. 2018 (cited on pp. 21, 74).

[387] M. Welk, J. Weickert and G. Steidl: From tensor-driven diffusion to anisotropic wavelet shrinkage, In H. Bischof, A. Leonardis and A. Pinz (Eds.): *Computer Vision – ECCV 2006, Part I*, Lecture Notes in Computer Science, Vol. 3951, 391–403, Springer, Berlin, 2006 (cited on p. 92).

[388] J. Weng, N. Ahuja and T. S. Huang: Cresceptron: a self-organizing neural network which grows adaptively, *Proc. 1992 International Joint Conference on Neural Networks*, Vol. 3, 576–581,

Baltimore, MD, June 1992, IEEE Computer Society Press (cited on p. 46).

[389] P. J. Werbos: Applications of advances in nonlinear sensitivity analysis, In R. F. Drenick and F. Kozin (Eds.): *System Modeling and Optimization*, Lecture Notes in Control and Information Sciences, Vol. 38, 762–770, Springer, Berlin, Heidelberg, 1982 (cited on p. 59).

[390] P. J. Werbos: Backwards differentiation in AD and neural nets: past links and new opportunities, In M. Bücker, G. Corliss, U. Naumann, P. Hovland and B. Norris (Eds.): *Automatic Differentiation: Applications, Theory, and Implementations*, Lecture Notes in Computational Science and Engineering, Vol. 50, 15–34, Springer, Berlin, Heidelberg, 2006 (cited on pp. 59, 210).

[391] E. T. Whittaker: A new method of graduation, *Proceedings of the Edinburgh Mathematical Society*, Vol. 41, 65–75, 1923 (cited on pp. 2, 15, 36, 116).

[392] T. Wiatowski and H. Bölcskei: A mathematical theory of deep convolutional neural networks for feature extraction, *IEEE Transactions on Information Theory*, Vol. 64, No. 3, 1845–1866, Nov. 2017 (cited on p. 63).

[393] T. Williams and R. Li: Wavelet pooling for convolutional neural networks, *Proc. 6th International Conference on Learning Representations*, Vancouver, Canada, Apr. 2018 (cited on p. 63).

[394] A. P. Witkin: Scale-space filtering, *Proc. Eighth International Joint Conference on Artificial Intelligence*, Vol. 2, 945–951, Karlsruhe, West Germany, Aug. 1983 (cited on p. 18).

[395] D. E. Worrall, S. J. Garbin, D. Turmukhambetov and G. J. Brostow: Harmonic networks: deep translation and rotation equivariance, *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 5028–5037, Honolulu, HI, July 2017, IEEE Computer Society Press (cited on pp. 64, 162).

[396] Y. Wu et al.: Google's neural machine translation system: bridging the gap between human and machine translation, *arXiv:1609.08144v2 [cs.CL]*, Oct. 2016 (cited on p. 48).

[397] J. Xie, L. Xu and E. Chen: Image denoising and inpainting with deep neural networks, *Proc. 26th International Conference on Neural Information Processing Systems*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger (Ed.), Vol. 25, Advances in Neural Information Processing Systems, 350–358, Lake Tahoe, NV, Dec. 2012 (cited on p. 209).

[398] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He: Aggregated residual transformations for deep neural networks, *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 1492–1500, Honolulu, HI, July 2017, IEEE Computer Society Press (cited on pp. 161, 174).

[399] H. Yan, J. Du, V. Tan and J. Feng: On robustness of neural ordinary differential equations, *Proc. 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, Apr. 2020 (cited on p. 62).

[400] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang and H. Li: High-resolution image inpainting using multi-scale neural patch synthesis, *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 6721–6729, Honolulu, HI, July 2017 (cited on pp. 67, 209).

[401] Y.-L. You and M. Kaveh: Fourth-order partial differential equations for noise removal, *IEEE Transactions on Image Processing*, Vol. 9, No. 10, 1723–1730, Oct. 2000 (cited on pp. 22, 132, 165, 177).

[402] I. T. Young, J. J. Gerbrands and L. J. van Vliet: Fundamentals of Image Processing, Delft University of Technology, 1995 (cited on pp. 105, 193, 214).

[403] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu and T. S. Huang: Generative image inpainting with contextual attention, *Proc. 2018 IEEE Conference on Computer Vision and Pattern Recognition*, 5505–5514, Salt Lake City, UT, June 2018 (cited on pp. 67, 209).

[404] X. Yuan, P. He, Q. Zhu and X. Li: Adversarial examples: attacks and defenses for deep learning, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 30, No. 9, 2805–2824, Sept. 2019 (cited on pp. 1, 61).

[405] S. Zagoruyko and N. Komodakis: Wide residual networks, *arXiv:1605.07146v4 [cs.CV]*, June 2017 (cited on p. 238).

[406] M. D. Zeiler and R. Fergus: Visualizing and understanding convolutional networks, In D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars (Eds.): *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, Vol. 8689, 818–833, Springer, Berlin, 2014 (cited on pp. 42, 48, 60).

[407] G. Zhai and X. Min: Perceptual image quality assessment: A survey, *Science China Information Sciences*, Vol. 63, Article no. 211301, Nov. 2020 (cited on p. 13).

[408] C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals: Understanding deep learning (still) requires rethinking generalization, *Communications of the ACM*, Vol. 64, No. 3, 107–115, Mar. 2021 (cited on p. 2).

[409] H. Zhang, X. Gao, J. Unterman and T. Arodz: Approximation capabilities of neural ODEs and invertible residual networks, *Proc. 37th International Conference on Machine Learning*, H. Daumé III and A. Singh (Ed.), Vol. 119, Proceedings of Machine Learning Research, 11086–11095, Vienna, Austria, July 2020 (cited on p. 62).

[410] K. Zhang: Existence of infinitely many solutions for the one-dimensional Perona–Malik model, *Calculus of Variations and Partial Differential Equations*, Vol. 26, 171–199, Mar. 2006 (cited on p. 21).

[411] K. Zhang, W. Zuo, Y. Chen, D. Meng and L. Zhang: Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising, *IEEE Transactions on Image Processing*, Vol. 26, No. 7, 3142–3155, July 2017 (cited on p. 65).

[412] L. Zhang and H. Schaeffer: Forward stability of ResNet and its variants, *Journal of Mathematical Imaging and Vision*, Vol. 62, 328–351, Apr. 2020 (cited on pp. 49, 114, 130, 134, 135, 166).

[413] R. Zhang, P. Isola, A. A. Efros, E. Shechtman and O. Wang: The unreasonable effectiveness of deep features as a perceptual metric, *Proc. 2018 IEEE Conference on Computer Vision and Pattern Recognition*, 586–595, Salt Lake City, UT, June 2018, IEEE Computer Society Press (cited on p. 13).

[414] X.-P. Zhang: Thresholding neural network for adaptive noise reduction, *IEEE Transactions on Neural Networks*, Vol. 12, No. 3, 567–584, May 2001 (cited on p. 74).

[415] X.-P. Zhang and M. D. Desai: Nonlinear adaptive noise suppression based on wavelet transform, *Proc. 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 3, IEEE Computer Society Press, 1589–1592, Seattle, WA, May 1998 (cited on p. 74).

[416] M. Zhu, B. Chang and C. Fu: Convolutional neural networks combined with Runge-Kutta methods, *Proc. 7th International Conference on Learning Representations*, New Orleans, LA, May 2019 (cited on pp. 130, 137, 139).

[417] M. Zhu, W. Min, Q. Wang, S. Zou and X. Chen: PFLU and FPFLU: two novel non-monotonic activation functions in convolutional neural networks, *Neurocomputing*, Vol. 429, 110–117, Mar. 2021 (cited on p. 130).

F

GLOSSARY

Adam . . . . Adaptive Moment Estimation.

AI . . . . . Artificial Intelligence.

BFB . . . . . Balanced Forward-backward Diffusivity.

BSDS500 . . Berkeley Segmentation Database.

CG . . . . . Conjugate Gradients.

CNN . . . . Convolutional Neural Network.

dB . . . . . Decibels.

EED . . . . . Edge-enhancing Anisotropic Diffusion.

FAB . . . . . Forward-and-backward Diffusivity.

FAS . . . . . Full Approximation Scheme.

FED . . . . . Fast Explicit Diffusion.

FMG . . . . Full Multigrid Scheme.

FSI . . . . . Fast Semi-iterative Scheme.

GPU . . . . . Graphics Processing Unit.

IAD . . . . . Integrodifferential Anisotropic Diffusion.

IID . . . . . Integrodifferential Isotropic Diffusion.

JPEG . . . . Joint Photographic Experts Group.

JPEG2000 . . Joint Photographic Experts Group 2000.

L-BFGS . . . Limited Memory Broyden–Fletcher–Goldfarb–Shanno Algorithm.

LP . . . . . Luma Preference Mode.

LPAQ . . . . Light PAQ, compression algorithm.

LSTM . . . . Long Short Term Memory.

MNIST . . . Modified National Institute of Standards and Technology Database.

MSE . . . . . Mean Square Error.

MS-EED . . Multiscale Edge-enhancing Anisotropic Diffusion.

NLPE . . . . Non-local Pixel Exchange.

NODE . . . Neural Ordinary Differential Equation.

ODE . . . . Ordinary Differential Equation.

PDE . . . . . Partial Differential Equation.

PS . . . . . Probabilistic Sparsification.

PSNR . . . . Peak Signal-to-noise Ratio.

R-EED . . . Rectangular Subdivision with Edge-enhancing Diffusion.

ReLU . . . . Rectified Linear Unit.

ResNet . . . Residual Network, specific neural network architecture.

RGB . . . . . Red-green-blue Colour Space.

RJIP . . . . . Regular Joint Inpainting and Prediction.

RNN . . . . Recurrent Neural Network, specific neural network architecture.

SymResNet Symemtric Residual Network, specific neural network architecture.

TO . . . . . Tonal Optimisation.

TV . . . . . Total Variation.

U-net . . . . U-net, specific neural network architecture.

XOR . . . . . Logical exclusive or operation.

YCbCr . . . Colour space with luma (Y) and chroma (Cb,Cr) components

G

LIST OF SYMBOLS

$*$	Convolution operator, defined in Equation (2.7).
$\lvert\cdot\rvert$	Absolute value or shorthand notation for Euclidean norm.
$\lVert\boldsymbol{v}\rVert_p$	General vector norm, index p denotes type of norm. Defined in Equation (2.15).
$\lVert\boldsymbol{A}\rVert_2$	Spectral norm of a matrix, defined in Equations (2.20) and (2.21).
$\lVert\boldsymbol{A}\rVert_F$	Frobenius norm of a matrix, defined in Equation (2.22).
$\perp$	Orthogonal vectors.
$\parallel$	Parallel vectors.
$\boldsymbol{\nabla}$	Gradient operator, defined in Equation (2.2).
$\boldsymbol{\nabla}^\top$, div	Divergence operator, defined in Equation (2.3).
$\boldsymbol{\nabla}^\perp$	Orthogonal gradient, introduced in Section (3.1.1).
$\boldsymbol{\nabla}_\sigma$	Regularised gradient operator, introduced in Equation (3.17).
$\mathbf{0}$	Vector filled with zeroes.
$\mathbf{1}$	Vector filled with ones.
a	Diffusion tensor entry, defined in Equation (3.2).
$\boldsymbol{A}$	Discrete differential operator.
b	Diffusion tensor entry, defined in Equation (3.2). Also used as bias variable in neural networks, defined in Equation (3.92).
$\boldsymbol{b}$	Right-hand side of a linear system. Also used for neural network bias vectors.
c	Diffusion tensor entry, defined in Equation (3.2). Also used as indicator function for continuous inpainting masks.
$\boldsymbol{c}$	Discrete inpainting mask.

$\boldsymbol{C}$	General set of neural network convolutions. Also used as a diagonal matrix for inpainting masks.
d	Inpainting mask density.
d_x, d_y	Sampling distances for regular masks.
$\boldsymbol{D}$	Diffusion tensor as defined in Fick's law (3.1).
$\mathcal{D}$	General data term or general differential operator.
$\boldsymbol{D}_h^+, \boldsymbol{D}_h^-$	Forward and backward difference matrices with grid size h.
$\boldsymbol{e}$	Standard unit vector of length one.
$E(\cdot)$	Variational energy functional, introduced in Section 3.2.
f	Continous noisy input image or signal for denoising, masked image or signal for inpainting.
$\boldsymbol{f}$	Discrete noisy input image or signal for denoising, masked image or signal for inpainting.
$F(\cdot)$	Integrand of variational energy functional, introduced in Section 3.2.
$g(s^2)$	Diffusivity function, introduced in Section 3.1.1.
$g_{\max}$	Maximum value of a diffusivity g.
$\boldsymbol{G}$	Diganoal diffusivity matrix.
$G(x, y)$	Two-dimensional Gaussian function.
h, h_x, h_y, H	Grid sizes of discrete signals and images, introduced in Section 2.5.
$\boldsymbol{H}(u)$	Hessian matrix.
i, j	Spatial indices for discrete signals and images.
$\boldsymbol{I}$	Identity matrix.
$\mathcal{I}$	Inpainting network in Chapter 10.
Id	Identity function.
$\boldsymbol{j}$	Diffusion flux as defined in Fick's law (3.1).

$\boldsymbol{J}$ . . . . . Structure tensor, defined in Equation 5.3. The extension $\boldsymbol{J}_\gamma$ denotes the multiscale structure tensor in Chapter 5.

k . . . . . Time step index, introduced in Section 3.1.2. Also used as a spatial index when i, j are in use.

K . . . . . Inpainting mask, domain of known data for inpainting tasks. Also used for the total number of time steps, and K_σ denotes a Gaussian with standard deviation σ.

$\boldsymbol{K}$ . . . . . Discrete filter matrices in diffusion models and neural networks.

ℓ . . . . . Scale index. Also used for fractional time steps.

L . . . . . General differential operator or Lipschitz constant of functions. Also used for total number of scales and fractional time steps.

$\mathcal{L}$ . . . . . Neural network loss function, introduced in Section 3.4.3.

m . . . . . Alternative notation for the number of explicit diffusion steps. Also used as image channel index.

M . . . . . Order of a differential equation or operator. Also used for the number of image channels.

$\mathcal{M}$ . . . . . Mask learning network in Chapter 10.

n, n_x, n_y . . . Number of sampling positions for discrete signals and images. The symbol n is alternatively used to denote noise standard deviation in Chapter 5 when σ is already in use.

$\boldsymbol{n}$ . . . . . Outer normal vector at image boundary $\partial\Omega$. Also used to denote images or signals of pure noise.

$\mathcal{N}$ . . . . . General neural network in Section 3.4.3.

$\mathcal{O}(\cdot)$ . . . . . Landau notation, defined in Equation (2.10).

p, q . . . . . Alternative spatial indices for discrete signals and images. Also used as candidate pixel fractions for probabilistic sparsification.

P . . . . . Lipschitz constant for multiscale models where L is in use.

$\boldsymbol{P}$ . . . . . Prolongation operator in a multigrid algorithm, introduced in Section 7.4.

$\boldsymbol{Q}$ . . . . . Undecimated Haar wavelet filters, defined in Section 4.1. Also used as system matrix of explicit schemes in Section 3.1.2.

$\boldsymbol{Q}$ . . . . . Non-decimated Haar wavelet filter matrices in Chapter 4.

$\boldsymbol{r}$ . . . . . Discrete residual vector of a PDE, introduced in Section 7.4.

$\boldsymbol{R}$ . . . . . Restriction operator in a multigrid algorithm, introduced in Section 7.4.

$\mathcal{R}$ . . . . . General regularisation term.

$S(s)$ . . . . . Wavelet shrinkage function, introduced in Section 3.3

$\boldsymbol{S}$ . . . . . General discrete solver, introduced in Section 7.4. Also used as vector-valued shrinkage function.

sgn . . . . . Sign function.

sup . . . . . Supremum operator.

T . . . . . Stopping time of a diffusion process, introduced in Section 3.1.2.

tr . . . . . Trace operator.

u . . . . . Continous denoised or inpainted output image or signal. Evolving image or signal in diffusion processes and neural networks.

$\boldsymbol{u}$ . . . . . Discrete denoised or inpainted output image or signal. Evolving image or signal in diffusion processes and neural networks.

v . . . . . Continous ground truth image or signal.

$\boldsymbol{v}$ . . . . . Discrete ground truth image or signal.

$\boldsymbol{v}_1, \boldsymbol{v}_2$ . . . . Eigenvectors of the diffusion or structure tensor, introduced in Section 3.1.1.

$w(\cdot)$ . . . . . Weighting function for Shepard interpolation.

$\boldsymbol{w}$ . . . . . Neural network filter weights, introduced in Section 3.4. Also wavelet coefficient vectors in Chapter 4.

$\boldsymbol{W}$ . . . . . Neural network filter matrices, introduced in Section 3.4. Also used as non-decimated à trous Haar wavelet filter matrices in Chapter 4.

$x, y, \boldsymbol{x}, \boldsymbol{y}$ . . . Spatial variables in scalar and vector form. Also used as general variables.

α . . . . . Various definitions; see respective chapters.

β . . . . . Various definitions; see respective chapters.

β_1, β_2 . . . . Momentum parameters of the Adam optimiser, introduced in Section 3.4.3.

γ . . . . . Various definitions; see respective chapters.

$\partial_t, \partial_x, \partial_y$ . . . Partial derivatives, defined in Equation (2.1).

∂_u . . . . . Gâteaux derivative, defined in Equation (2.6).

$\partial_v, \partial_{\boldsymbol{n}}$ . . . . Directional derivatives, defined in Equation (2.5).

$\partial\Omega$ . . . . . Boundary of signal or image domain Ω.

Δ . . . . . Laplace operator, defined in Equation (2.4).

Δ^2 . . . . . Biharmonic operator, introduced in Section 2.1.

ε . . . . . Small numerical regularisation constant.

θ . . . . . Rotation angle for anisotropic Shepard interpolation in Chapter 9. Also used as super time step size for FED (see Section 3.1.2) and as threshold parameter for wavelet shrinkage functions (see Section 3.3).

$\boldsymbol{\theta}$ . . . . . Set of trainable parameters, introduced in Section 3.4.3.

$\boldsymbol{\vartheta}$ . . . . . Parameter update vector in momentum methods, introduced in Equation 3.106.

λ . . . . . Contrast parameter for nonlinear diffusivities, introduced in Section 3.1.1.

μ . . . . . Mean value.

ν_1, ν_2 . . . . . Eigenvalues of the diffusion or structure tensor, introduced in Section 3.1.1.

ρ	Regularisation parameters.
$\rho(\boldsymbol{A})$	Spectral radius of a matrix, defined in (2.25).
σ	Standard deviation of a Gaussian. Presmoothing parameter in the case of diffusion; see Section 3.1.1 and Chapters 5 and 8. Alternatively used as noise standard deviation in denoising experiments.
τ	Time step size of diffusion processes, introduced in Section 3.1.2.
$\varphi(s)$	Activation function in neural networks, introduced in Equation 3.92.
$\Phi(s)$	Flux function, defined in Equation (3.14). Alternatively the scaling function for Wavelet shrinkage, introduced in Section 3.3
$\Psi(s^2)$	Variational penaliser, introduced in Section 3.2. Also used for the mother wavelet function introduced in Section 3.3.
ω	Weight for discrete multiscale models, introduced in Section 5.2.
Ω	Rectangular image domain or one-dimensional signal domain.

LIST OF FIGURES

2.1 Unit balls for selected vector norms 9
2.2 Continuous and discrete image examples 12

3.1 Diffusivities and associated flux functions 20
3.2 Visualisations of diffusion models 26
3.3 Mother wavelet and scaling function 40
3.4 ReLU activation 43
3.5 Fully connected neural network 44
3.6 Convolutional neural network 45
3.7 Visualisation of a two-dimensional CNN 47
3.8 Visualisation of a U-net 50
3.9 Activations and derivatives 53
3.10 Historic timeline of artificial intelligence 58
3.11 Denoising with diffusion models 66
3.12 Inpainting without and with spatial optimisation . . . 69

4.1 Hard, soft, and garrote shrinkage functions 78
4.2 Effect of parameter choices on the FAB diffusivity . . 81
4.3 Trained shrinkage functions for $\sigma = 25$ 84
4.4 Trained shrinkage functions for $\sigma = 50$ 85
4.5 Contribution of scales to reconstruction quality 86
4.6 Relations between trained parameters and scale . . . 86
4.7 Relations between trained parameters and noise . . . 87
4.8 Comparison against classical shrinkage functions . . . 88
4.9 Visual comparison of reconstructions for $\sigma = 25$ . . . 89
4.10 Visual comparison of reconstructions for $\sigma = 50$ . . . 89

5.1 Learned weight and contrast parameters 98
5.2 Learned intermediate parameters 99
5.3 Ablation study over steps and scales 100
5.4 Denoising performance on *peppers* 102
5.5 Denoising performance on *cameraman* 102
5.6 Analysis of eigenvalues of diffusion tensors 103
5.7 Inpainting results on random mask 106
5.8 Inpainting results on optimised masks 107

6.1 One-dimensional diffusion block 118

7.1 Generalised one-dimensional diffusion block 133
7.2 Du Fort–Frankel block 139
7.3 FSI block 140
7.4 Neural representations of implicit diffusion 141

7.5 U-net and multigrid networks . . . 145
7.6 Denoising quality of time constant networks with a single channel . . . 152
7.7 Denoising quality of time dynamic networks with a single channel . . . 153
7.8 Denoising quality of time dynamic networks with multiple channels . . . 154
7.9 Full multigrid strategy . . . 157
7.10 EED inpainting with different networks . . . 158

8.1 Generalised two-dimensional diffusion block . . . 168
8.2 Fully coupled multi-channel diffusion block . . . 172
8.3 Fully coupled multiscale diffusion block . . . 175
8.4 Analysis of rotation invariance . . . 179
8.5 Denoising results for differently rotated datasets . . . 180

9.1 Local adaptation of the weighting function. . . . 189
9.2 Quality of anisotropic and isotropic Shepard interpolation . . . 194
9.3 Anisotropic and isotropic Shepard interpolation on *kodim23* . . . 195
9.4 Anisotropic and isotropic Shepard interpolation on grey and colour versions of *peppers*. . . . 197
9.5 Effect of tonal optimisation . . . 199
9.6 Average PSNR for all codec versions . . . 201
9.7 Visual comparison of colour codec variants . . . 202
9.8 Comparison of codec variants on colour images . . . 204
9.9 Comparison of our codec variants against the results of Mohideen et al. . . . 206

10.1 Overview of our model structure . . . 211
10.2 Homogeneous diffusion inpainting results . . . 216
10.3 Biharmonic diffusion inpainting results . . . 217
10.4 EED inpainting results . . . 218
10.5 Comparison of average inpainting quality . . . 219
10.6 Comparison of efficiency . . . 220

LIST OF TABLES

3.1 Overview of popular diffusion models 24

5.1 Average PSNR on the test set 103

6.1 Dictionary of nonlinear modelling functions 121
6.2 Plots of functions resulting in monotone activation functions 124
6.3 Formulas for the function plots in Table 6.2 125
6.4 Plots of functions resulting in nonmonotone activation functions 126
6.5 Formulas for the function plots in Table 6.4 127

7.1 Overview of our connections between numerical and neural concepts 159

8.1 Diffusion models, variational energies, and network architectures 177

9.1 Comparison of PSNR values on selected images . . . 196

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Individual modifications to this style have been performed by Tobias Alt, some of them based on adaptations that have been kindly provided by Dr. Pascal Peter.

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

https://miede.de/postcards/

Thank you very much for your feedback and contribution.