EPFL

Thèse n° 10 667

Towards Trustworthy Deep Learning for Image Reconstruction

Présentée le 8 mars 2024

Faculté des sciences et techniques de l'ingénieur
Laboratoire d'imagerie biomédicale
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Alexis Marie Frederic GOUJON

Acceptée sur proposition du jury

Prof. P. Frossard, président du jury
Prof. M. Unser, directeur de thèse
Prof. I. Selesnick, rapporteur
Prof. R. Nowak, rapporteur
Prof. D. Van de Ville, rapporteur

École polytechnique fédérale de Lausanne

2024

Abstract

The remarkable ability of deep learning (DL) models to approximate high-dimensional functions from samples has sparked a revolution across numerous scientific and industrial domains that cannot be overemphasized. In sensitive applications, the good performance of DL is unfortunately sometimes overshadowed by unexpected behaviors, including hallucinations in medical image reconstruction. Serious concerns have thus been raised regarding the extent to which one can trust the output of DL models. Restoring trust is challenging since the same depth that fuels the performance causes DL models to be black boxes. The parameters of the model are indeed only remotely connected to the function they parameterize, and enforcing constraints on the model to obtain guarantees on its output usually wipes out the performance boost of DL. In this thesis, we pursue the goal of improving the trustworthiness of several DL methods while maintaining performance. Our approach tackles the problem via the design of expressive, stable and interpretable spline-based parameterizations across various contexts.

The contributions of this thesis are divided into three parts. In the first part, we concentrate on parameterizations for low-dimensional regression tasks. There, depth is not beneficial and one can have it all—stability, expressivity and interpretability—with linear combinations of well-chosen atoms. This is first shown with the design of shortest-support multi-spline bases, and then with the study of the stability of a local parameterization of continuous and piecewise-linear functions (CPWL).

In the second part, we focus on deep parameterizations to cope with higher-dimensional problems. We first study the composition operation within CPWL neural networks (NN) and give some new insights into the role of the activation function in the expressivity of the NN. We then propose to use Lipschitz-constrained learnable linear spline activations to build expressive and provably stable deep NN. We characterize some universal properties of our framework, develop an efficient procedure to train the activations under the constraint, and, lastly, show experimental improvements over competing frameworks with similar constraints on various tasks, including plug-and-play image reconstruction with provably nonexpansive denoisers.

In the third and final part, we refine the parameterization by focusing on image reconstruction tasks. We propose a framework to learn convex regularizers, which rely on our learnable Lipschitz-constrained spline activations. The parameterization yields lightweight and transparent—in contrast to black boxes—models with theoretical guarantees on the reconstruction. Our method exhibits state-of-the-art performance for CT and MRI recon-

struction among convex regularization methods. Lastly, we extend the framework to learn weakly-convex regularizers to boost performance while maintaining most guarantees.

Keywords: deep learning, inverse problems, medical imaging, image reconstruction, regularization, plug-and-play methods, sparsity, splines, continuous and piecewise-linear functions, Lipschitz constant

Résumé

Les dernières avancées dans le domaine de l'apprentissage profond ont permis une véritable révolution dans de nombreux domaines, des sciences fondamentales au développement de nouvelles technologies. Malgré ces progrès remarquables, les algorithmes d'apprentissage profond présentent parfois des comportements très inattendus, menant à des phénomènes d'hallucination en imagerie médicale par exemple. De ce fait, au-delà des performances, la question de la confiance que l'on peut accorder aux résultats des algorithmes a pris une place fondamentale. Un facteur aggravant est que la profondeur des réseaux de neurones, nécessaire à l'obtention de bonnes performances, rend la compréhension du fonctionnement de l'algorithme difficile. Dans cette thèse, l'objectif est d'améliorer la fiabilité de plusieurs méthodes d'apprentissage profond en minimisant l'impact négatif sur les performances. Pour ce faire, notre approche s'appuie sur de nouvelles paramétrisations expressives, stables et interprétables.

Les contributions de cette thèse sont divisées en trois parties. Dans la première partie et en guise de première étape, nous nous concentrons sur des problèmes de régression avec un nombre faible de variables. Dans ce cas, une paramétrisation profonde n'est pas nécessairement bénéfique et il est possible d'obtenir la stabilité, l'expressivité et l'interprétabilité avec des combinaisons linéaires de fonctions de bases bien choisies. Ceci est d'abord démontré par la conception de bases multi-splines à support minimal, puis par l'étude de la stabilité d'une paramétrisation locale de fonctions continues et linéaires par morceaux.

Dans la deuxième partie, nous considérons des paramétrisations plus profondes pour faire face à des problèmes de plus grande dimensionnalité. Nous étudions d'abord l'opération de composition dans les réseaux de neurones (RN) d'un point de vue géométrique, ce qui aboutit à une meilleure compréhension du rôle des modules non linéaires dans l'expressivité des RN. Nous proposons ensuite d'utiliser des fonctions d'activations modulables avec contrôle de leur constante de Lipschitz pour construire des RN expressifs et stables, donc théoriquement plus fiables. Nous caractérisons certaines propriétés universelles de notre paramétrisation et développons une procédure efficace pour apprendre les fonctions d'activation contraintes. Enfin, nous illustrons les avantages pratiques de notre méthode dans le cas de la reconstruction d'image avec des garanties théoriques de stabilité.

Dans la troisième et dernière partie, nous affinons la paramétrisation pour nous focaliser exclusivement sur des problèmes de reconstruction d'images. Nous proposons une méthode pour apprendre des fonctions de régularisation convexes, qui s'appuient sur notre module de fonctions d'activations modulables avec contraintes sur leur constante de Lipschitz. Notre choix de paramétrisation permet de ne recourir qu'à un nombre faible de paramètres, et mène à un algorithme de reconstruction interprétable avec des garanties théoriques sur la reconstruction. Notre méthode améliore la qualité de reconstruction par rapport aux autres méthodes de régularisation convexe sur diverses modalités d'imagerie médicale. Enfin, nous étendons notre méthode à l'apprentissage de fonctions de régularisation faiblement convexes, ce qui permet d'améliorer les performances tout en conservant la plupart des garanties de fiabilité.

Acknowledgements

I am very thankful to Prof. Michael Unser for giving the opportunity me to join his lab. Back in 2020, I had not been exposed yet to his research areas of expertise—from signal and image processing to splines—but yet, he took a risk and offered me a Ph.D. position. Throughout my stay at EPFL, I have benefited from his relentless optimism and support, for which I am very grateful. He also gave me the freedom to explore many research topics—as long as I could incorporate some splines, which turned out not to be a very limiting constraint! My sincere thanks also go to Prof. Frossard, Prof. Selenisck, Prof. Van De Ville and Prof. Nowak for agreeing to be part of my thesis jury and taking the time to review my work.

The last years would certainly have been less colorful without my colleagues at BIG. I would like to sincerely thank all current and past members of the lab for eager scientific discussions and refreshing lunch breaks. Both the scientific collaboration and friendly atmosphere have been very enjoyable on many different levels. The acknowledgments I am now writing are not sufficient to convey my gratefulness to each member, and I apologize for that. I am also very thankful to the permanent members of the lab—Claudia, Daniel and Philippe—for their kindness and help whenever needed.

Lastly, my tender thanks go to my wife Anne-Claire. She has helped me in all possible aspects during this Ph.D., from endless scientific discussions to very insightful feedback on my writings, and so much more.

Contents

Abstract (English) **i**

Résumé (Français) **iii**

Acknowledgements **v**

Abbreviations **xiii**

---

1 Introduction **1**
- 1.1 The Deep-Learning Revolution . . . 1
  - 1.1.1 The Roots of Deep Learning . . . 1
  - 1.1.2 Deep Learning for Image Reconstruction . . . 2
- 1.2 Beyond Performance, What about Trustworthiness? . . . 6
  - 1.2.1 Black-box Nature of Deep Neural Networks . . . 7
  - 1.2.2 Instabilities of Deep Neural Networks . . . 9
- 1.3 Improving Trustworthiness . . . 12
  - 1.3.1 The Three Pillars of Trustworthiness . . . 12
  - 1.3.2 Controlling the Black Box to Build Stable DNNs . . . 13
  - 1.3.3 Model-Based Deep Learning for Image Reconstruction: Improved Interpretability and Challenges . . . 15
- 1.4 Outline and Contributions . . . 17

I The World of Splines for Low-Dimensional Problems **21**

2 Shortest-support Multi-Spline Bases **23**
- 2.1 Summary . . . 23
- 2.2 Introduction . . . 24
  - 2.2.1 Generalized Sampling in Shift-Invariant Spaces . . . 24
  - 2.2.2 Polynomial Splines . . . 24
  - 2.2.3 Multi-Splines . . . 25
- 2.3 Formulation of the Problem . . . 26

2.3.1 Riesz Basis . . . 26
2.3.2 Reproducing Polynomials . . . 27
2.3.3 Compact Support . . . 28
2.4 Shortest Bases . . . 28
2.5 Multi-Spline Shortest Bases . . . 34
2.5.1 Consecutive Multi-Spline Spaces . . . 35
2.5.2 Existence and Construction of mB-Splines . . . 37
2.6 Applications . . . 41
2.6.1 Generalized Sampling in Multi-Spline Spaces . . . 41
2.6.2 Derivative Sampling with High-Degree Multi-Splines in $S_{2p} + S_{2p+1}$ 42
2.6.3 Classical Interpolation . . . 46
2.6.4 Bézier Curves and Computer Graphics in $S_1 + S_2 + S_3$ and $S_1 + S_2$ 48
2.6.5 Nonconsecutive Bi-spline Spaces . . . 48
2.7 Conclusion . . . 49

3 Stable Parameterization of Continuous and Piecewise-Linear Functions 51
3.1 Introduction . . . 52
3.1.1 Continuous and Piecewise-Linear Functions for Supervised Learning 52
3.1.2 Linear Expansion of Continuous and Piecewise-Linear Functions . 52
3.1.3 Stability of the Parameterizations . . . 54
3.1.4 Contributions and Outline . . . 56
3.2 Mathematical Preliminaries . . . 56
3.2.1 Simplicial Continuous and Piecewise-Linear Functions . . . 56
3.2.2 Riesz Basis and Riesz Ratio . . . 59
3.2.3 Lipschitz Constant . . . 60
3.3 Affine Functions on Simplices . . . 61
3.4 Riesz Ratio on Arbitrary Triangulations . . . 65
3.4.1 Triangulations with Any Number of Vertices . . . 65
3.4.2 Triangulations with Finitely Many Vertices . . . 68
3.5 Exact Riesz Bounds for Linear Box Splines . . . 70
3.5.1 Linear Box Splines . . . 70
3.5.2 Derivation of the Exact Riesz-Basis Bounds . . . 72
3.6 Lipschitz Bounds . . . 75
3.6.1 Lipschitz Properties of ReLU Networks . . . 75
3.6.2 Lipschitz Properties of the Local Parameterization . . . 76
3.7 Conclusion . . . 78
3.8 Appendix – Interpolation Condition Number of the Nonlocal Parameterization 80
3.9 Appendix – The Generalized Hinging-Hyperplane Generating Function of Linear Box Splines . . . 80

II Going Deeper with Stability Guarantees 83

4 On the Number of Regions of Continuous and Piecewise-Linear Neural Networks 85

4.1 Introduction . . . 86
4.2 Mathematical Preliminaries . . . 89
4.2.1 CPWL Functions . . . 89
4.2.2 Regions of CPWL Functions and Convex Partitions . . . 90
4.2.3 Arrangement of Convex Partitions . . . 92
4.3 Maximum Number of Regions Produced by CPWL NNs . . . 93
4.3.1 Upper Bound on the Number of Regions of Arrangements . . . 93
4.3.2 Single Hidden-Layer: Bound for the Sum and Vectorization Operations 95
4.3.3 Multiple Hidden-Layers: Compositional Bounds . . . 97
4.3.4 Application to Some Popular CPWL NNs . . . 99
4.4 Expected Number of Regions Produced by CPWL NNs Along 1D Paths . 100
4.4.1 Knot Density . . . 102
4.4.2 Knot Density of CPWL Layers . . . 104
4.4.3 Bounds on the Expected Knot Density of CPWL NNs . . . 106
4.5 Conclusion . . . 110
4.6 Appendix – Proofs for Section 4.3 . . . 111
4.6.1 Number of Convex vs Projection Regions . . . 111
4.6.2 Upper Bound on the Number of Regions of Arrangements . . . 112
4.6.3 Sum and Vectorization . . . 115
4.6.4 Compositional Bounds . . . 116
4.7 Appendix – Proofs for Section 4.4.2 . . . 120

5 Spline Activations for Stable and Expressive Deep Neural Networks 123

5.1 Introduction . . . 124
5.2 Layer-wise Lipschitz-Constrained Neural Networks . . . 126
5.2.1 Preliminaries . . . 126
5.2.2 Universality of 1-Lipschitz ReLU Networks . . . 126
5.2.3 Layer-wise 1-Lipschitz NNs . . . 129
5.2.4 1-Lip Linear Layers . . . 129
5.2.5 1-Lip Activation Functions . . . 130
5.3 2 Linear Regions are not Enough . . . 133
5.3.1 Limitations of ReLU . . . 134
5.3.2 Representation Power of 2-Linear Region Splines . . . 137
5.4 3 Linear Regions Suffice . . . 138
5.4.1 A Representer Theorem . . . 139
5.4.2 3 Linear Regions Suffice . . . 142
5.5 Comparison with GroupSort and Summary . . . 145
5.6 Conclusions and Open Problems . . . 146
5.7 Appendix – Second-Order Total Variation . . . 147

6 Spline Activations for Stable Plug-and-Play Image Reconstruction 149
6.1 Introduction to Plug-and-Play Methods . . . 150
6.1.1 Motivations . . . 150
6.1.2 Convergent Plug-and-Play Methods . . . 151
6.2 Preliminaries: Averaged Operators and Fixed-point Iterations . . . 153
6.3 Properties of the Reconstruction Map for PnP with Lipschitz Constrained Denoisers . . . 154
6.4 Implementing Lipschitz Constraints on Learnable Linear Splines . . . 156
6.4.1 Learnable Linear Splines . . . 156
6.4.2 Constrained and Learnable Linear Splines . . . 157
6.5 Illustration on Toy Problems . . . 159
6.5.1 One-Dimensional Function Fitting . . . 160
6.5.2 Wasserstein GAN Training . . . 161
6.6 Image Reconstruction: from Denoising to MRI and CT . . . 163
6.6.1 Denoising . . . 163
6.6.2 Medical Image Reconstruction . . . 164
6.7 Conclusion . . . 169
6.8 Appendix . . . 170
6.8.1 Properties of the Reconstruction Maps . . . 170
6.8.2 Properties of SplineProj . . . 171
6.8.3 Scale Invariance of $\mathrm{TV}^{(2)}$ . . . 172

III From Convex to Weakly-convex Data-driven Regularization 173

7 A Neural-Network-Based Convex Regularizer for Image Reconstruction 175
7.1 Introduction . . . 175
7.1.1 Linear Inverse Problems . . . 175
7.1.2 Deep-Learning Methods . . . 176
7.1.3 The Quest for Trustworthiness . . . 177
7.2 Architecture of the Regularizer . . . 179
7.2.1 General Setting . . . 179
7.2.2 Gradient-Step Neural Network . . . 180
7.3 Characterization of Good Profile Functions . . . 180
7.3.1 Existence of Minimizers and Stability of the Reconstruction . . . 180
7.3.2 Expressivity of Profile Functions . . . 182
7.4 Implementation . . . 185
7.4.1 Training a Multi-Gradient-Step Denoiser . . . 185
7.4.2 Implementation of the Constraints . . . 186
7.4.3 From Gradients to Potentials . . . 189
7.4.4 Boosting the Universality of the Regularizer . . . 189
7.4.5 Reconstruction Algorithm . . . 190

7.5 Connections to Deep-Learning Approaches . . . 190
7.5.1 Plug-and-Play and Averaged Denoisers . . . 191
7.5.2 Deep Convex Regularizers . . . 194
7.6 Experiments . . . 195
7.6.1 Training of CRR-NNs . . . 195
7.6.2 Denoising: Comparison with Other Methods . . . 195
7.6.3 Biomedical Image Reconstruction . . . 197
7.6.4 Under the Hood of the Learnt Regularizers . . . 202
7.7 Conclusion . . . 205
7.8 Appendix . . . 205
7.8.1 Hyperparameter Tuning . . . 205
7.8.2 Convergence Curves . . . 208
7.8.3 Activations and Filters . . . 208
7.8.4 Reconstructed images . . . 209

8 Learning Weakly Convex Regularizers for Convergent Image-Reconstruction Algorithms 213
8.1 Summary . . . 213
8.2 Introduction . . . 213
8.2.1 Linear Inverse Problems . . . 213
8.2.2 The Convex Non-Convex Framework for Denoising . . . 214
8.2.3 Extension to Ill-Posed Inverse Problems . . . 215
8.2.4 Other Deep-Learning-Based Variational Methods with Some Guarantees . . . 215
8.2.5 Outline and Main Contributions . . . 216
8.3 Weakly Convex Regularizers . . . 217
8.4 Design of a Learnable and Provably 1-Weakly Convex Regularizer for Denoising . . . 218
8.4.1 Regularizer Architecture . . . 219
8.4.2 Multi-Noise-Level Denoiser . . . 223
8.4.3 Training Procedure . . . 224
8.4.4 Training and Denoising Performance . . . 226
8.4.5 Interpretation as Sparsity Prior . . . 230
8.5 Extension to Generic Inverse Problems . . . 232
8.5.1 Accelerated Gradient Descent . . . 233
8.5.2 Experiments . . . 238
8.6 Conclusion . . . 243
8.7 Appendix - Additional Reconstructed Images . . . 244

9 Conclusion 249
9.1 Summary of the Main Contributions . . . 249
9.1.1 The World of Splines . . . 249

9.1.2 Going Deeper with Stability Guarantees 250
9.1.3 From Convex to Weakly-convex Data-driven Regularization 251
9.2 Future Directions, Open Problems and Perspectives 252

Bibliography **257**

Curriculum Vitae **283**

Abbreviations

1-Lip	1-Lipschitz
ACR	Adversarial convex regularizer
AdaGD	Adaptive gradient descent
ADMM	Alternating-direction method of multipliers
AELR	Average number of effective linear regions
AGD	Accelerated gradient descent
AV	Absolute value
BCOP	Block convolution orthogonal parameterization
BM3D	Block-matching and 3D filtering
BSD	Berkeley segmentation dataset
BV	Bounded variation
CNC	Convex non-convex
CNN	Convolutional neural network
CPWL	Continuous and piecewise linear
CReLU	Concatenated rectified linear unit
CRR	Convex ridge regularizer
CS	Compressed sensing
CT	Computed tomography
DEQ	Deep equilibrium
DFT	Discrete Fourier transform
DL	Deep learning
DNN	Deep neural network
DRS	Douglas–Rachford splitting
DSNN	Deep spline neural network
DWT	Discrete wavelet transform
e.g.	*Exempli gratia* (Latin for "for example")
ELU	Exponential linear unit
FBP	Filtered back projection
FBS	Forward-backward splitting
FIR	Finite impulse response
FISTA	Fast iterative shrinkage-thresholding algorithm
FoE	Fields of experts

GAN	Generative adversarial network
GD	Gradient descent
GHH	Generalized hinging hyperplane
GNP	Gradient norm preserving
HH	Householder
HTV	Hessian total variation
ICNN	Input convex neural network
IDWT	Inverse discrete wavelet transform
i.e.	*Id est* (Latin for "that is")
i.i.d.	Independent and identically distributed
IR	Image reconstruction
ISTA	Iterative shrinkage-thresholding algorithm
KL	Kurdyka-Łojasiewicz
LLS	Learnable linear spline
LW	Layer-wise
MRF	Markov random field
MRI	Magnetic resonance imaging
MSE	Mean squared error
NLP	Natural language processing
NN	Neural network
PD	Proton-density
PDFS	Proton-density with fat suppression
PGD	Proximal gradient descent
PnP	Plug and play
PSNR	Peak signal-to-noise ratio
PReLU	Parametric rectified linear unit
PWLU	Piecewise linear unit
RED	Regularization by denoising
ReLU	Rectified linear unit
ROI	Region of interest
RPGD	Relaxed projected gradient descent
SAGD	Safeguarded accelerated gradient descent
SSIM	Structural similarity index measure
s.t.	Such that
TNRD	Trainable nonlinear reaction diffusion
TV	Total variation
VN	Variational network
WCRR	Weakly convex ridge regularizer

1 Introduction

The following introduction presents a high-level overview of the context of the thesis. Detailed and more technical introductions can be found at the beginning of each chapter.

1.1 The Deep-Learning Revolution

The recent advances in artificial intelligence have been fueled in many fields by the deep-learning (DL) revolution. While several fundamental components of DL have existed for some time, it was not until the mid-2010s that DL emerged as a cross-domain paradigm, paving the way for breakthroughs in both scientific research and industrial applications. In science, DL has now become a pivotal computational tool with remarkable applications in biology [1], earth science [2], astrophysics [3], chemistry [4], material science [5], economics [6] and in many other branches of science. The impact of DL also goes beyond scientific advancements, catalyzing the development of disruptive technologies in transportation [7] with for instance self-driving cars [8] and drones [9], in healthcare [10] with for instance medical imaging [11], and in text processing with natural language processing (NLP)[12] systems like ChatGPT, among others. These achievements may just be the beginning, with the potential for even more groundbreaking developments in the future.

1.1.1 The Roots of Deep Learning

DL is a versatile framework to craft predictive models for a wide variety of tasks, including classification, to predict discrete class labels, and regression, to predict real-valued quantities. The success of DL stems from its ability to build, from training data, a prediction map, referred to as the input-to-output map, that yields remarkably accurate predictions on new data, even for problems with high dimensional inputs. This is achieved through the utilization of deep neural networks (DNNs) as parameterization tools. More precisely, a DNN architecture turns a set of parameters—finite-dimensional vectors that can be handled by a computer—into an input-to-output map that can be

used for prediction. Broadly speaking, parameterizations allow one to optimize over a space of input-to-output maps via the optimization of a finite number of parameters. The specificity of DNNs lies in the depth of parametrization: DNNs produce input-to-output maps made of the composition of multiple simple modules, including linear layers and activation functions. In this way, the input is sequentially processed to produce the output. The optimization of the DNN parameters relies on gradient-based algorithms that aim at maximizing the predictive performance of the model on the training data. The training of a DNN often requires significantly many training samples and computation steps. For these reasons, many advancements in DL have been catalyzed by the ever-increasing availability of data and computational resources. Over the years, the design and training of DNNs have significantly improved, and there now exists a set of well-established guidelines and tricks. However, certain aspects still demand expertise and application-specific knowledge.

In this thesis, we mainly focus on the applications of DL for image reconstruction.

1.1.2 Deep Learning for Image Reconstruction

Image Reconstruction, Inverse Problems and Challenges

Image reconstruction (IR), also referred to as image recovery, denotes the process of forming an image from observations—also called measurements—that may be corrupted, incomplete and not directly interpretable by a human.

When the observations form an image, the task boils down to image restoration [13]. In this case, the goal is to enhance the degraded image that was affected by various issues, including noise (denoising), blur (deblurring), low resolution (super-resolution), missing parts (inpainting), or grayscale representation requiring colorization, see Figure 1.1 for some examples of image degradation. The applications of image restoration are wide-ranging, finding significant utility in diverse fields. For instance, it plays a crucial role in biological imaging, including electron, fluorescence, and deconvolution microscopy [14, 15].

Image reconstruction also encompasses tasks that involve less direct measurements, with many applications in medical imaging, including computed tomography (CT) scan, magnetic resonance imaging (MRI), and ultrasound imaging as the most prominent ones [16]. In these cases, one can retrieve the interior structure of a human body from external, indirect measurements. The challenge here is to mathematically invert the measurement process. For example, in CT, the measurements, which form a so-called sinogram, are a set of line integral (tomographic projections) obtained from the propagation of X-rays within the slice of interest, with different incident angles [17], see Figure 1.2 for an illustration.

Mathematically, image reconstruction is often formalized as an inverse problem. The term “inverse problem” refers to the recovery of a ground truth signal from measurements,

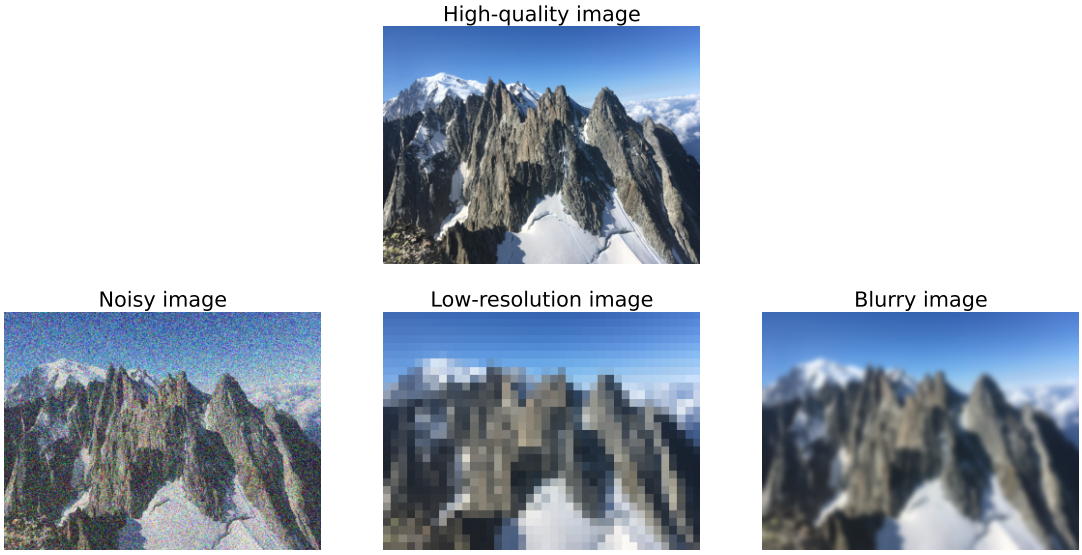

Figure 1.1: Examples of degraded images.

and thus it encompasses problems beyond imaging. Note that many imaging problems yield linear inverse problems since the measurement operator is (well modeled by) a linear operator [18]. Image reconstruction often leads to ill-posed inverse problems, for instance when the number of measurements is not sufficient to exactly recover the ground truth. This occurs in applications in which there are strong motivations to collect only a few measurements, either to limit the acquisition time or to reduce some risk, e.g. in X-ray to limit tissue damage. A direct inversion of the measurement process for ill-posed problems is delicate since different signals may lead to the observed measurements. This is illustrated with sparse-view CT in Figure 1.2. Even when well-posed, many inverse problems are ill-conditioned, in which case a direct inversion of the measurement process amplifies some components of the noise contained in the measurements and yields low-quality images. Overcoming these difficulties necessitates the application of advanced algorithms to reconstruct consistent and high-quality images.

All reconstruction algorithms presented and discussed in the remainder of the introduction are visually represented in Figure 1.3.

Classical Reconstruction Methods with Variational Regularization

The issues of ill-posedness and ill-conditioning in inverse problems can be mitigated with regularization methods, which allow for reconstructing images that combine two desirable properties.

- **Data consistency**. The reconstruction is expected to be consistent with the measurements, in the sense that simulated measurements with the reconstruction

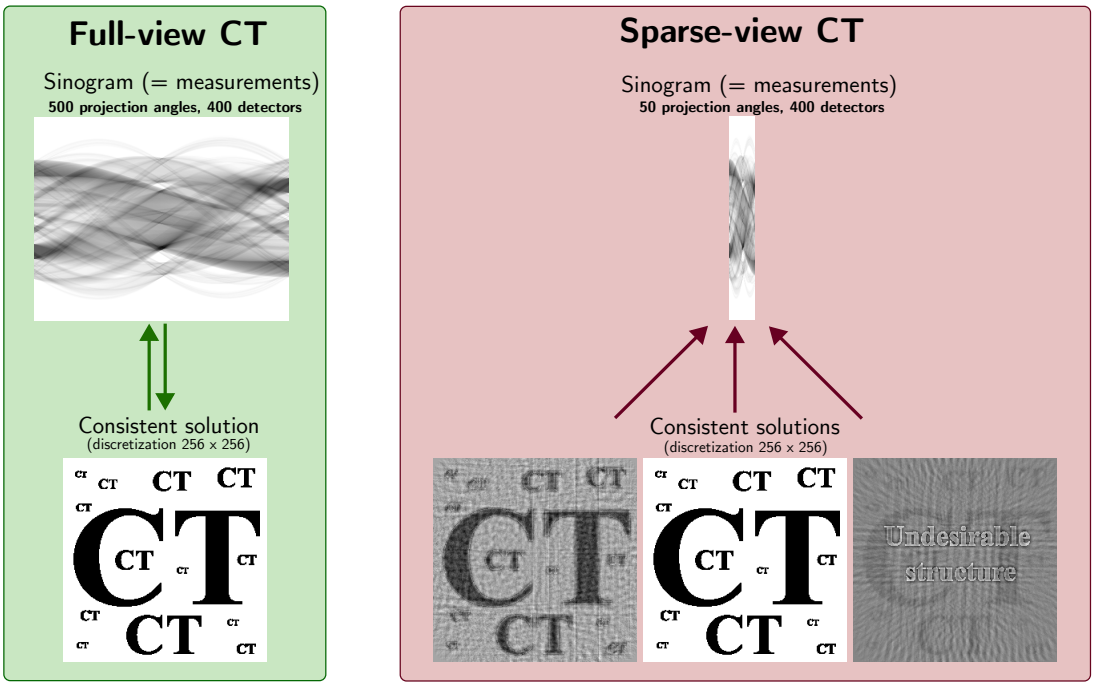

Figure 1.2: CT image reconstruction of a synthetic 2D slice in a noiseless scenario. The reconstructions are computed to be fully consistent with the corresponding sinograms. Left: with sufficiently many projections, the problem is well-posed and there exists a unique consistent solution. Right: with sparse-view measurements, infinitely many reconstructions are consistent with the measurements. (Color convention: high attenuation areas are represented in black and areas transparent to X-ray are represented in white.)

should be sufficiently close* to the observed ones. Data consistency can be quantified by various metrics, for instance, the Euclidian distance between the observed and simulated measurements.

- **High image quality**. For ill-posed and ill-conditioned inverse problems, data consistency is not sufficient since many images with good data consistency contain aliasing artifacts and noise, see Figure 1.2 for an illustration. The idea is thus to pick the most realistic image among the sufficiently consistent ones. In classical regularization methods, a regularity metric is thus introduced to quantify the intrinsic likelihood of any image, regardless of the measurements.

Regularization is implemented by reformulating the inverse problem as an optimization task so that the reconstruction is defined as the image that maximizes a joint measure of data consistency and regularity. The main challenge here is to design an effective regularity metric that penalizes undesirable images. One popular instance is the Tikhonov regularization, which promotes some smoothness in the reconstruction [19]. Another instance, more effective for IR tasks and especially in medical imaging, is the family

*Exact data consistency is not necessarily desirable when the measurements are noisy.

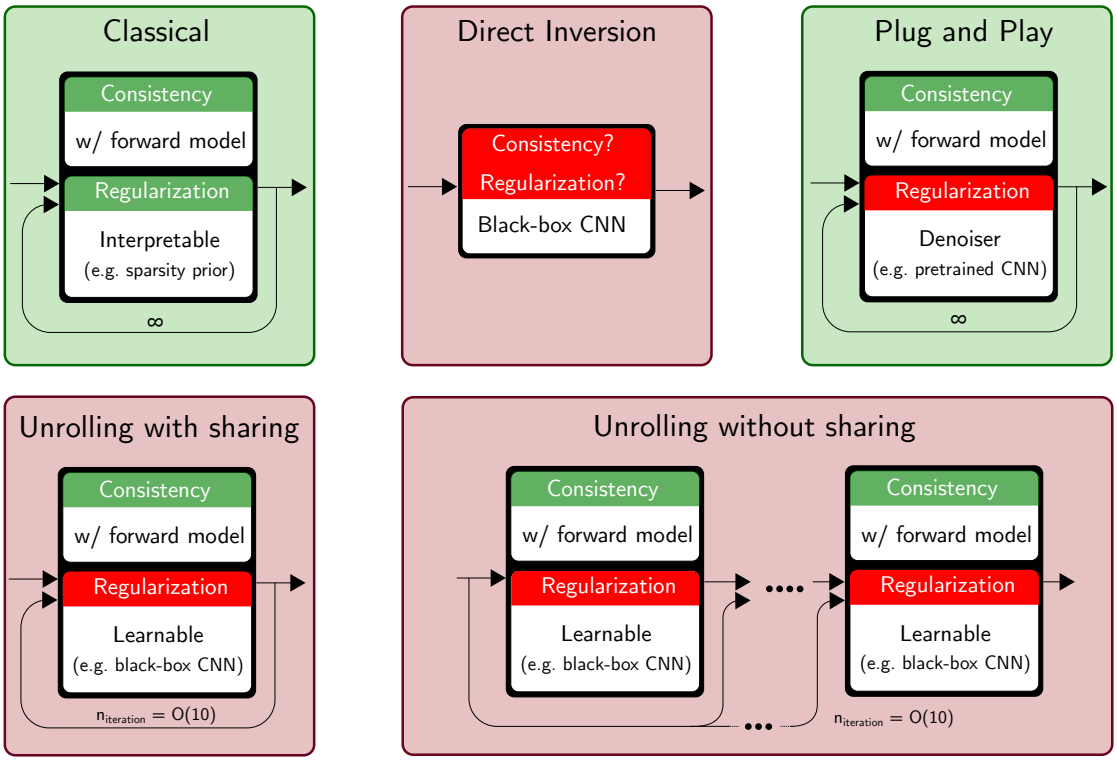

Figure 1.3: Overview of the reconstruction methods discussed in the introduction. All methods take the measurements as input and output a reconstruction. The methods that require end-to-end training are represented within a red box, the iterative methods feature a feedback loop, the ∞ symbol refers to methods that compute a fixed point, the modules with a red-background header typically incorporate data-driven components and hence should be carefully analyzed to assess the trustworthiness of the method.

of sparsity-promoting regularizers [20, 21], which includes the popular total-variation (TV) one [22]. Such regularizers favor images with few nonzero components in a well-chosen transformed representation, e.g. a wavelet or a finite difference one. Classical regularization methods are popular because they offer a number of desirable properties. They are interpretable since it is clear what properties they promote, for instance, TV regularization promotes piecewise constant images. They usually come with a set of convergent iterative algorithms to efficiently compute the reconstruction [23–28]. They offer some guarantees on the recovery of the ground truth under some conditions, on the consistency of the reconstruction, and on the stability of the measurement-to-reconstruction map, see for instance [29–31]. However, classical regularizers also have limitations. They tend to be too simplistic to fully capture the complexity of image distributions and therefore remain simplistic to address missing observations and fill the gaps. The performance limitations do not alter the trust in the method since regularization-related artifacts are usually straightforward to identify, and trained practitioners will not interpret them as a feature of the original signal. For example, reconstructions with Tikhonov regularization often have blurred edges, and TV regularization is known to

promote piecewise constant images, making it prone to introducing staircase artifacts.

DL-Based Image Reconstruction – the First Generation

The breakthroughs of DL in computer vision have brought about a paradigm shift in image reconstruction. The rapid growth in publications solving image reconstruction tasks with DL methods depicted in Figure 1.4 showcases its popularity, making it challenging to stay up-to-date with the latest advancements.

The success of DL-based image reconstruction relies on the utilization of convolutional neural networks (CNNs) combined with abundant training data to build implicit data-driven regularization. CNNs are DNNs in which the linear layers are convolutions. CNNs are highly efficient in processing images and have now become a fundamental tool for image reconstruction.

On a high level, DL-based reconstruction methods leverage the data seen during training to tackle ill-posedness and ill-conditioning. The first generation of DL methods proposed in medical image reconstruction would perform a direct inversion of the measurement process [32–35]. This is achieved by training a DNN to directly reconstruct the image from the measurements or a low-quality reconstruction in an end-to-end manner. Other end-to-end learning methods include algorithm unrolling [36], which are discussed in more detail in Section 1.3. One challenge with end-to-end methods is that they require abundant data with high-quality target reconstructions. One way to do so, for example in CT, is to use high-quality images reconstructed from full-view projections and standard dose measurements. Then, the model is only given access to subsampled measurements, to simulate sparse-view or limited-angle projections, or to the measurements degraded by synthetic noise, to simulate low-dose CT. End-to-end learning has demonstrated remarkable improvements in image quality and reconstruction speed compared to classical methods. In particular, it made it possible to reconstruct images from fewer measurements (sparse-view CT) or more noisy ones (low-dose CT) [34].

Compared to classical methods, such DL-based methods have, however, some important limitations, including a lack of interpretability, a lack of theoretical guarantees, e.g. no control of the data consistency, and some erratic behavior (lack of stability). These shortcomings are analyzed in detail in Section 1.2, and reveal that the performance of DL methods comes at the cost of reduced trustworthiness. The recent efforts to restore trustworthiness are then described in Section 1.3.

1.2 Beyond Performance, What about Trustworthiness?

The tremendous success of DL comes with serious reserves when it comes to sensitive applications. It is indeed often unclear to what extent the output of a DNN should

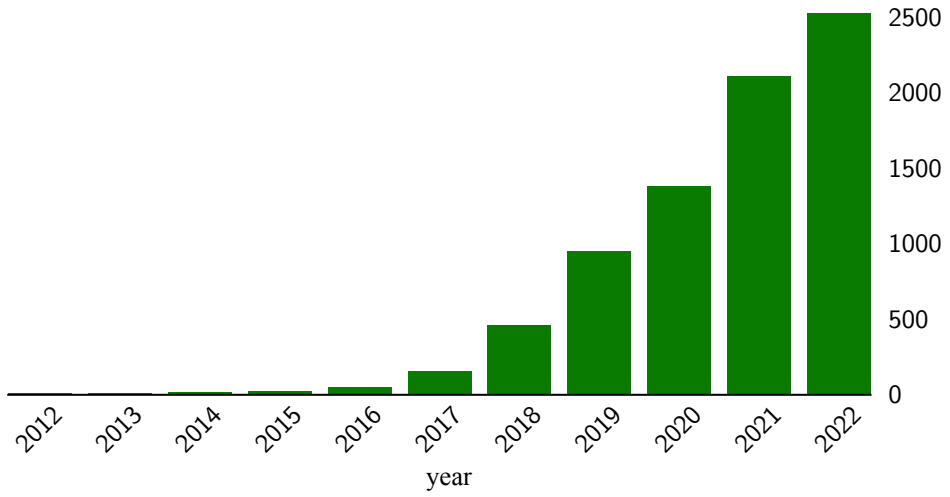

Figure 1.4: Number of published papers related to image reconstruction and deep learning over the years. The numbers correspond to the search "topic=deep learning and image and (restoration or reconstruction)" performed on July 27th 2023 in Clarivate Web-of-Science(WoS). Note that many recent frameworks rely on deep learning without explicitly mentioning it.

be trusted. For applications with critical safety concerns, such as in medical imaging, DL-based algorithms must provide strong evidence that they can be trusted before being deployed.

In this section, we discuss why DNNs are hard to trust by design and then discuss the implications for IR. Restoring trust is a very active topic that will be discussed in Section 1.3 and further explored in this thesis for IR applications.

1.2.1 Black-box Nature of Deep Neural Networks

Parameterization

DNNs offer an efficient parametrization to perform local computations. This includes the computation of the output given an input, with the forward pass, and of derivatives of the output with respect to the parameters or the input, with the back-propagation algorithm [37, 38]. Unfortunately, on a global scale, the input-to-output map generated by a DNN is very challenging to inspect. This is first caused by the overwhelming number of parameters, which can easily reach millions, if not billions in the most recent DNNs. Second, even if one could inspect the parameters, their connection to the input-to-output map is highly nontrivial due to the successive compositions performed within the DNN, which makes it not human interpretable, and what the internal DNN's modules are doing usually remains a mystery. Even the composition of one-dimensional functions is already not intuitive, as illustrated in Figure 1.5. Classic functional analysis tools, such as the Fourier transform, fail to provide any useful insight with such nonlinear parametrizations. Ultimately, the role of each parameter is not known, to the point that some parameters

may not even play a role in the input-to-output map. For example, in NNs equipped with the rectified linear unit (ReLU), during the training stage, neurons can "die" because they consistently output 0, leaving a whole subset of parameters definitively useless [39]. And this is not even easy to diagnose.

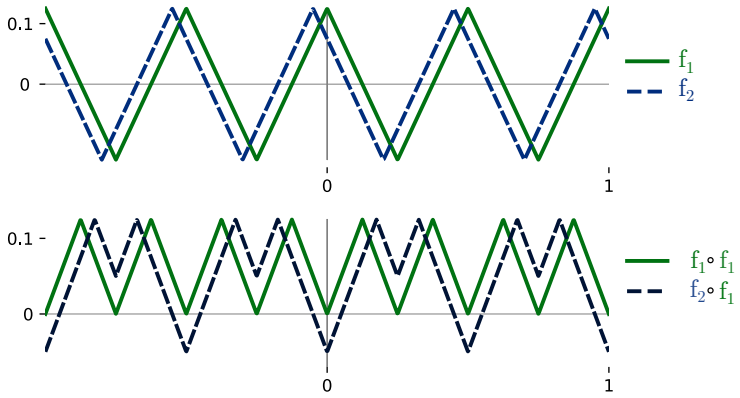

Figure 1.5: Effect of a small shift on the composition of two one-dimensional functions. While the composition operation yields great expressivity to DNNs, it is not intuitive, even in 1D and makes the internal functioning of DNNs hard to understand.

Training

One reason why a trained DNN is hard to trust is that the dependence of its parameters on the optimization algorithm and the training data is poorly understood. The learning of the parameters involves highly non-convex optimization tasks, which can lead to numerous NP-hard problems that are impractical to solve exactly [40–42]. Consequently, training a DNN necessitates the use of various empirical techniques and heuristics with few theoretical justifications, resulting in a lack of guarantees for the trained DNN and limited trustworthiness. It should be noted, however, that current training methods are typically efficient for a DNN to fit training data with the remarkable property that it generalizes well to new data [43, 44]. Experimental validation may help to build trust, but it requires to have at hand a realistic and high-quality dataset, which may not always be available.

Controlling the Black Box, or Not?

The opaque parameterization and training of DNNs make it nonobvious to impose certain constraints on its output or on the input-to-output map, which is often a minimal requirement to build trust. Many of the pioneering works that brought DL into various fields of science would typically omit some physical constraints or only approximately enforce them [45–47].

For example, in mechanics and physics, DNNs can be used to emulate the dynamics of systems from training data. Such data-driven models are unfortunately not guaranteed to output solutions consistent with the laws of physics, e.g. they may violate conservation laws or some known properties of the system (incompressibility of a flow, rigidity of a body...) or the equations governing the dynamics of the system. Similarly, in IR, end-to-end reconstruction methods do not guarantee any consistency of the reconstruction with the measurements, which is a huge loss compared to classical methods with convex regularization for instance. Enforcing hard constraints in DL is often nontrivial and therefore often omitted, although it is becoming an active topic of research in many fields, and will be further discussed in Section 1.3. One difficulty pertains to the lack of understanding of the input-to-output map, which makes some global properties of the mapping inaccessible in practice. To give a concrete example, it is known that the computation of the Lipschitz constant of a DNN, which measures the maximum rate of change of the output with respect to a change in the input, leads to NP-hard problems not tractable in practice [48]. This is an actual problem because the control of the Lipschitz constant of a DNN is necessary for many applications, and can for instance provide robustness guarantees for classifiers or stability guarantees for IR iterative methods (see Section 1.3 for more details).

The black-box nature of DNNs is not only a theoretical problem; DNNs sometimes exhibit very unexpected behaviors in practice.

1.2.2 Instabilities of Deep Neural Networks

Adversarial Attacks and Other Instabilities

One way to assess whether DNNs are robust predictors is to test their performance under adversarial attacks. An adversarial example is a carefully crafted input that is close to a legitimate one but for which the model outputs a completely different prediction [49–51]. There exists a large variety of algorithms to design such attacks, and they manage to fool many DL pipelines [52]. Most of the early works provide white-box attacks in the sense that the attacker needs to access the DNN's architecture and its parameters to fool it, for example with gradient-based optimization [50, 51]. With image classifiers, it was found that small perturbations, not even visible to the human eye, could drastically change the class predicted, see Figure 1.6 for such an example. Even more puzzling, the DNN can show arbitrarily high confidence while mistaking. This unexpected phenomenon reveals that the good performance of a DNN locally, i.e. on a training/test set, does not imply generalization over the whole input domain. Trustworthiness is therefore far from being granted with DNNs, even though white-box attacks remain somehow theoretical.

There now exist more realistic attacks, including black-box attacks, that craft adversarial examples by querying the DNN on a sequence of well-chosen inputs without accessing

the internal structure of the model [53, 54], and are therefore able to fool APIs. Note that black-box attacks also remain delicate to interpret because they only seek worst-case examples, regardless of their probability of occurring in an actual scenario. Evaluating the probability of failure of a DNN is very challenging in practice.

Other evidence of the unstable nature of DNNs is revealed with out-of-distribution tests. The DNN is tested on data that differ in nature from the training data, in such a way that the task does not look different or even harder to humans. For classification problems, one can for instance add slightly blurry images or images with new styles and textures. Such distribution shifts do not alter the semantics found in the images but suffice to drastically reduce the performance of a DNN [55]. This phenomenon underlines that it is not well-understood how DNNs build their prediction since it might not generalize well to tasks that are similar to the one used for training.

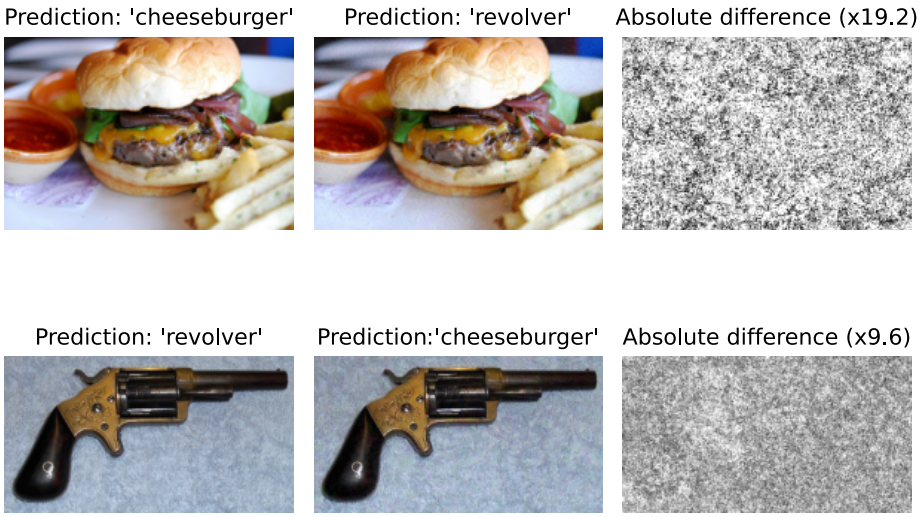

Figure 1.6: Adversarial attack with projected gradient descent [56] performed on the ResNet50 classifier available in PyTorch. The first column depicts a legitimate input, the second one an adversarial example, and the last one the absolute difference between the legitimate input and the adversarial example summed up along the 3 color channels.

Hallucinations in Image Reconstruction

The problem of instabilities of DL is also observed in regression problems and is a major concern for image reconstruction. The addition of adversarial noise to the measurements or a small change of the measurement operator drastically reduces the performance of DNNs trained in an end-to-end fashion to reconstruct MRI and CT images [57]. More recent studies suggest that sparsity-promoting classical reconstruction methods are also sensitive to perturbations [58, 59]. However, the instabilities with such methods are of different nature. Classical regularization methods have a clear interpretation, and when they malfunction they produce well-understood artifacts that can be identified by

radiologists. For instance, total-variation regularization may produce staircase artifacts and yield aliasing artifacts. With DNNs, small perturbations lead to artifacts in the image that may be hard to identify because they are typically (i) structured, (ii) realistic and (iii) vary a lot from one DNN to another [57]. Incomplete measurements in some ill-posed IR problems favor DNNs to hallucinate, in the sense that they build realistic structures given only too little information with no guarantees of their presence in the ground-truth signal. The great representation power of DNNs makes them sometimes overstep the role of a regularizer. Particularly telling examples were seen in the fastMRI challenge [60], in which radiologists were evaluating end-to-end DL reconstruction methods proposed by competing teams. Radiologists raised serious concerns after detecting hallucinations in MRI reconstructions. While the DNNs yielded very good performance in terms of the standard metrics, they would sometimes incorporate local abnormalities into a normal brain structure, including a sulcus or vessel, see Figure 1.7. Another example can be found in [61], in which a DL IR method has been found to wipe out tumors [61]. DNNs can therefore truly lead to error in the diagnosis since they can mimic realistic structures that are not present and potentially abnormal.

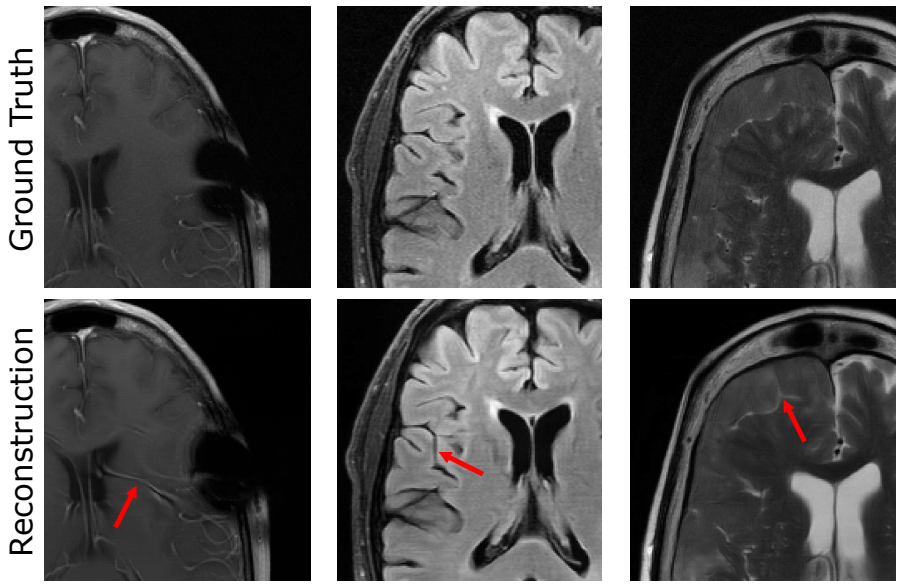

Figure 1.7: Illustration of hallucinations with end-to-end DL-based methods for MRI image reconstruction. The reconstruction methods used were among the best-performing ones in the fastMRI challenge [60]. The figure is a modified version of [60, Figure 6] licensed under CC BY-ND 4.0.

Despite the good performance of DNNs in many fields, their unexpected behaviors and the lack of understanding of their functioning make it necessary to build trust before deploying them in sensitive applications.

1.3 Improving Trustworthiness

1.3.1 The Three Pillars of Trustworthiness

Restoring trust in DL models while keeping the performance boost is a critical challenge and a very active topic of research [62, 63]. Many concepts revolving around trustworthiness have been revisited and specified in the context of DL, including verification [64], testing [65], interpretability [66], explainability [67], robustness [68] and reliability [69]. In the context of this thesis, we propose to analyze the trustworthiness of IR methods according to three components.

1. **Empirical trustworthiness, built on tests.** Trust is first built upon experimental testing to assess quantitatively and qualitatively the performance of a method. Unfortunately, it is usually not sufficient because tests always rely on some assumptions, e.g. regarding the measurement operator, and test sets may contain some biases. In addition, the metrics to quantify performance also have limitations, especially with images, for which assessing the reconstruction quality is not obvious, and the aggregation of the performance over a data set is not trivial either.
2. **Theoretical trustworthiness, built on guarantees.** In many IR tasks, some natural constraints emerge, see Section 1.2.1 and 1.3.3 for more details. A trustworthiness method should provably meet such constraints. Note that the constraints come in various forms, and may apply to the prediction, e.g. data consistency, or to the reconstruction pipeline, e.g. (Lipschitz) continuity of the reconstruction with respect to the measurements, or convergence for iterative methods.
3. **Human trustworthiness, built on interpretability.** This corresponds to being able, as a human (or at least as a field expert), to understand the way the model is built and the way it makes predictions. Field experts should have some interpretability of the method to understand its strengths and limitations, e.g. predict when it might fail or on the contrary for which setups it should generalize well.

The quest for state-of-the-art performance left the second and third pillars mainly disregarded in the first generation of DL methods, with potentially serious problems for sensitive applications, as showcased in Section 1.2.1. In fields where DL is sufficiently mature, such as IR, it is time to better balance the importance of the three pillars.

In the remainder of this section, the discussion on trustworthiness is limited only to topics that will further be explored throughout this thesis. We first discuss how functional analysis can shed some light on black-box DNNs and eventually lead to DNNs with stability guarantees. Then, we discuss the recent improvements in the trustworthiness of DL-based IR methods and highlight the remaining challenges.

1.3.2 Controlling the Black Box to Build Stable DNNs

Spline Theory to Open the Black Box

Many popular DNNs employ the rectified linear unit (ReLU) as the activation function. These DNNs create input-to-output mappings that are continuous and piecewise linear (CPWL) [70], equivalently referred to as linear splines [71]. This means that ReLU DNNs divide the input domain into regions where the mapping simply boils down to an affine transformation, see Figure 1.8 for an illustration. This straightforward functional perspective has sparked extensive discussions in the last decade. DNNs can now be viewed as local linear regressors or, equivalently, as generating input-dependent affine transformations [71]. In the case of CNNs, this interpretation extends further. CNNs perform linear filtering with kernels that adapt to the input data and that can be visualized and further analyzed [71]. For classifiers, it is possible to use the CPWL properties of DNNs to locally compute and visualize the decision boundary [72]. On a theoretical level, the CPWL property of DNNs has been instrumental in exploring the impact of depth and width in terms of their expressive capabilities. Studies have revealed that increasing depth can exponentially augment the number of linear regions [70, 73]. Although this result highlights the important role of depth in NNs, it also confirms that depth may produce mappings that are so complex that their exhaustive depiction, region by region, is not feasible for combinatorial reasons.

The spline perspective on DNNs improves their interpretability on a high level, but it once again underlines their complexity, especially for very deep NNs, and calls for better control of DNNs.

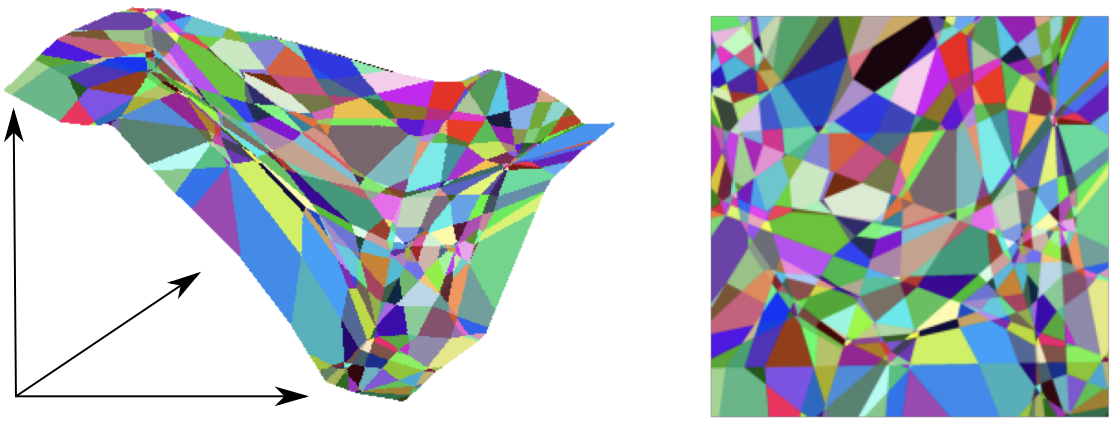

Figure 1.8: An $\mathbb{R}^2 \to \mathbb{R}$ CPWL function parameterized by a ReLU NN, and its corresponding partition of the input space.

Lipschitz-control for Robustness Guarantees

The discussion in Section 1.2 showed that unnoticeable perturbations of the input of a DNN may completely change its prediction. From a functional perspective, this can be quantified by the Lipschitz constant, which, by definition, measures the maximum rate of change of the output with respect to a change in the input. Hence, a possible way to enhance the stability of DNNs is to control their Lipschitz constant, ensuring that it remains not too large. For example, it is possible to guarantee adversarial robustness at a given input for Lipschitz continuous classifiers[†]. In particular, there cannot be adversarial examples that lie within a safe distance of the legitimate input. This distance is lower-bounded by the ratio of the margin at the input—which is readily computable and measures the confidence of the DNNs regarding the class predicted—by the Lipschitz constant. Hence, the smaller the Lipschitz constant is, the stronger the robustness guarantee against attacks we have. Unfortunately, the computation of the Lipschitz constant of a DNN is not feasible in practice, see details in Section 1.2.1. In addition, even if it was accessible, a standard unconstrained training would likely result in such a large Lipschitz constant that the bound on the safety distance would be extremely small and overly pessimistic at many locations. To address these issues, two approaches have emerged to better control the Lipschitz constant of DNNs in the training stage.

- **Soft Lipschitz constraints.** Soft constraints favor the learning of input-to-output maps with a small Lipschitz constant. For example, one can penalize the norm of the Jacobian of the DNN during training on a finite set of points—which is an estimate of the Lipschitz constant—via the addition of a regularization term in the training loss [75, 76]. This turns out to be empirically satisfactory in many settings [77, 78] but does not yield any theoretical guarantee.
- **Hard Lipschitz constraints.** Imposing a hard constraint is more challenging. Instead of controlling directly the Lipschitz constant of the whole DNN, one can control an upper bound on it that is efficiently computable. In practice, the canonical approach is to impose hard Lipschitz constraints on each layer [79]. This layer-wise approach follows from the fact that (i) the computation of the Lipschitz constant of each linear layer and each activation function is tractable and that (ii) the Lipschitz constant is sub-multiplicative with respect to the composition operation. Hence, layer-wise Lipschitz constraints yield an upper bound for the DNN's Lipschitz constant. In the last years, significant improvements have been made to better constrain linear layers while training [80–83]. In addition, the ReLU activation was found to be theoretically and empirically ill-suited to the hard-constraint approach [82]. This shortcoming has been addressed with the introduction of new activation functions or higher dimensional nonlinear modules that can easily be Lipschitz controlled [82, 84, 85].

[†]The Lipschitz constant at stake here is the one of the input-to-logits map, the input-to-class map being no continuous it is never Lipschitz-continuous.

The use of DNNs with control of their Lipschitz constants goes far beyond the robustness of classifiers. It is also very relevant to design invertible DNNs [86, 87], to stabilize the training of GANs [79, 88] and most importantly to us, it is a key ingredient for plug-and-play IR methods [81], which we discuss in details in Section 1.3.3.

Despite the recent progress, it is still unclear to what extent Lipschitz constraints can be combined with performance. Many theoretical questions also remain with the layer-wise approach. Although it seems to rely on a very pessimistic upper bound, it is not known if it actually limits theoretically the expressivity, for instance under 2-norm constraints, which is the setup relevant to IR methods.

1.3.3 Model-Based Deep Learning for Image Reconstruction: Improved Interpretability and Challenges

The CNN-based direct inversion methods discussed in Section 1.1.2 lack interpretability, can behave erratically (lack of stability) and fail to offer any theoretical guarantees. These issues can be partially mitigated with model-based DL, which provides a more sensible incorporation of DL in IR. The core idea is to craft separate data-consistency modules and regularization modules.

- **Data-consistency modules.** The data-consistency modules favor reconstructions that are consistent with the measurements. In many medical IR tasks, the acquisition process can be numerically modeled by a so-called forward model. In this scenario, the data-consistency modules do not need to be data-driven and can be fully physics-driven via the use of the forward model.
- **Regularization modules.** DL is powerful for designing and learning regularization modules that can reduce noise and aliasing artifacts. Regularization modules for IR rely on CNNs and, in the end, yield a data-driven implicit regularization.

The training of the regularization modules and their interplay with the data-consistency modules can be implemented in various ways.

Plug-and-Play Methods: Regularization with Off-the-Shelf Denoisers

The core idea of plug-and-play (PnP) methods is to form the implicit regularization prior from an off-the-shelf denoiser [89]. In classical methods, reconstructions are typically computed with iterative algorithms that sequentially resort to (i) a data-consistency operator that is built from the forward model, e.g. the gradient or the proximal operator of a measure of the data consistency, and (ii) a regularization operator that incorporates some prior knowledge on the image via the regularizer, e.g. the gradient or proximal operator of the regularizer. PnP algorithms extend such algorithms by crafting a regularization

operator from an image denoiser. A first instance is given by the proximal PnP methods, which extend proximal algorithms. The proximal operator of the regularizer, which can be interpreted as a denoiser, is replaced by the off-the-shelf denoiser. PnP methods can also extend gradient-based algorithms, for instance with the regularization by denoising (RED) framework [90]. The gradient of the regularizer, which can be interpreted as a noise extractor, is replaced by the residual operator of the off-the-shelf denoiser. PnP methods extend the popular iterative thresholding algorithms developed in the early 2000s [23, 24, 91], and the general formulation was given a decade later in [89], which showed that the handcrafted BM3D denoiser could be used to solve IR problems with improvements over classical methods. Since then, the performance of PnP methods has benefited from the continuous improvement of image denoisers boosted by the DL revolution [92].

Compared to direct inversion methods with CNNs, PnP methods have some clear advantages.

- **Universality.** PnP methods are more universal and easier to deploy since the same denoiser can be used in a wide variety of settings and accommodate different noise levels.
- **High-level interpretability.** PnP methods also improve interpretability because one can identify which module is responsible for data consistency and which one handles regularization. On another level, a PnP method outputs the fixed point of some recursive adjustment procedure so that the reconstruction can be interpreted as the image where data consistency and regularization balance each other, which is reminiscent of the classical IR algorithms.

PnP methods also come with many challenges that can alter their trustworthiness.

- **Lack of theoretical guarantees.** PnP with standard off-the-shelf denoisers offers no theoretical guarantees regarding (i) the convergence of the iterations, (ii) the consistency of the reconstructed image and (iii) the stability of the measurement-to-reconstruction map. Convergence can be granted in various ways, including by iteratively relaxing the denoiser [93, 94], by imposing some Lipschitz constraints on the denoiser [78, 81, 95] or by using denoisers that are the gradient of some scalar-valued map so that the PnP algorithm can be recast into a non-convex minimization problem [76, 96–99]. To date, however, the only known approach to address all three challenges for noninvertible forward models relies on the use of nonexpansive denoisers [78, 81, 95, 100]. Unfortunately, such a property is very nontrivial to obtain with CNN denoisers while keeping the performance boost offered by DL. The vast majority of frameworks violate such a constraint and relax it or turn it into a soft constraint [78, 81, 101]. This strategy is empirically satisfactory but wipes out all theoretical guarantees. Some recent versions of PnP achieve convergence

guarantees with expansive denoisers [76, 102], but the catch is that there remain no theoretical guarantees on the reconstruction, e.g. data consistency is not ensured and the scheme becomes sensitive to initialization [76].

- **Incomplete interpretability.** The denoisers typically used in PnP are black-box models so their internal functioning remains not interpretable. In addition, the fixed-point interpretation of PnP reconstruction—balance between data consistency and level of noise—is not necessarily strong. Indeed, many PnP frameworks with no or relaxed constraints may have a very diverse set of fixed points, not even connected. In the end, the quality of a fixed point might strongly depend on the algorithm and its initialization [76].

Algorithms Unrolling

Algorithm unrolling provides a way to turn any classical iterative algorithm into a DNN [36, 103, 104]. The DNN is made of a finite number of iterations of the algorithm in which the classical regularization operator is replaced at each iteration by learnable modules—which may or may not be shared across iterations. In this way, the data-consistency blocks of the DNN are formed with the forward model, while the regularization can be data-driven. The key difference with PnP methods is that the DNN is trained in an end-to-end manner and is task-specific. This makes unrolling less universal, but allows for a better coupling between data consistency and regularization, resulting in improved performance. Most unrolling schemes are very fast to reconstruct images because the number of iterations unrolled is typically small (around 10). Compared to direct CNN inversion, it was observed that unrolling requires less trainable parameters and less training data, and tends to generalize better. Unfortunately, regarding trustworthiness, unrolling typically comes with no theoretical guarantees on the data consistency and stability, see Figure 1.7 for examples of hallucination in MRI image reconstruction. The interpretation is also reduced by the end-to-end training and the opaque regularization modules, which tend to become deeper and deeper CNNs with always more parameters. In the end, it is by no means clear whether the output is data consistent since data consistency and regularization modules are not independent anymore.

1.4 Outline and Contributions

In this thesis, we pursue the goal of improving the trustworthiness of several DL methods while maintaining performance. Our approach tackles the problem via the design and mathematical analysis of novel parameterizations that are sufficiently expressive, for performance, provably stable, for theoretical guarantees, and interpretable, for human trust. The thesis is divided into three parts and a general overview is given in Figure 1.9.

The contributions are now briefly presented. For a more technical perspective, we refer

the reader to the Conclusion of the thesis.

Part I: The World of Splines for Low-Dimensional Problems

In the first part, as a preliminary step, we concentrate on parameterizations for low-dimensional regression tasks. There, depth is not beneficial and one can have it all—expressivity, stability and interpretability—with linear combinations of well-chosen atoms. This is first shown with the design of shortest-support multi-spline bases, and then with the study of the stability of a local parameterization of continuous and piecewise-linear functions (CPWL).

Relevant Publications

- A. Goujon, S. Aziznejad, A. Naderi and M. Unser "Shortest-support multi-spline bases for generalized sampling", *Journal of Computational and Applied Mathematics*, volume 395, paper 113610, October 2021.

- A. Goujon, J. Campos and M. Unser "Stable parameterization of continuous and piecewise-linear functions", *Applied and Computational Harmonic Analysis*, volume 67, paper 101581, November 2023.

- M. Pourya, A. Goujon and M. Unser "Delaunay-Triangulation-Based Learning with Hessian Total-Variation Regularization", *IEEE Open Journal of Signal Processing*, volume 4, page 167-178, February 2023.

Part II: Going Deeper with Stability Guarantees

In the second part, we focus on deep parameterizations to cope with higher-dimensional problems. We begin with the analysis of the geometry of CPWL DNNs and reveal the importance of the activation function to build expressive DNNs. We then propose to use Lipschitz-constrained learnable linear spline activations to build expressive and provably stable DNNs. We show that the splines need to have at least 3 linear regions to have the optimal representation power and show their theoretical advantage over popular activations used to build Lipschitz-constrained DNNs. We then develop an efficient procedure to train the constrained activation functions by introducing a reparameterization technique that allows for incorporating the constraints on the parameters into the computational graph of the DNN. Lastly, we use our framework to build nonexpansive denoisers for convergent and stable plug-and-play image reconstruction. We show experimental improvements over other DNNs with similar constraints but different activation functions on various tasks, from toy problems to CT and MRI image reconstruction.

Relevant Publications

- A. Goujon, A. Etemadi and M. Unser "On the number of regions of piecewise linear neural-networks", *Journal of Computational and Applied Mathematics*, volume 441, paper 115667, 2024.
- {A. Goujon[‡], S. Neumayer[‡]}, P. Bohra and M. Unser "Approximation of Lipschitz functions using deep spline neural networks", *SIAM Journal on Mathematics of Data Science*, volume 5, page 306-322, issue 2, June 2023.
- S. Ducotterd, A. Goujon, P. Bohra, D. Perdios, S. Neumayer and M. Unser "Improving Lipschitz-constrained neural networks by learning activation functions", submitted to *Journal of Machine Learning Research*.
- P. Bohra, D. Perdios, A. Goujon, S. Emery and M. Unser, "Learning Lipschitz-controlled activation functions in neural networks for plug-and-play image reconstruction methods", *NeurIPS 2021 Workshop on Deep Learning and Inverse Problems*.

Part III: From Convex to Weakly-convex Data-driven Regularization

In the third and final part, we refine the parameterization of Part II to address image reconstruction problems with explicit regularization. We propose a framework to learn convex regularizers, which rely on our learnable Lipschitz-constrained spline activations. The parameterization yields lightweight and transparent models—in contrast to black-box models—with theoretical guarantees on the reconstruction. Our method exhibits state-of-the-art performance for CT and MRI reconstruction among convex regularization methods. Lastly, we generalize the framework to learn weakly convex regularizers. While this extension maintains most guarantees and interpretability, it boosts the performance, surpassing the popular BM3D denoiser and approaching the performance of state-of-the-art PnP methods, which, by contrast, lack trustworthiness.

Relevant Publications

- A. Goujon, S. Neumayer, P. Bohra, S. Ducotterd and M. Unser "A neural-network-based convex regularizer for inverse problems", *IEEE Transactions on Computational Imaging*, volume 9, page 781-795, August 2023.
- A. Goujon, S. Neumayer and M. Unser "Learning weakly convex regularizers for convergent image-reconstruction algorithms", accepted in *SIAM Journal on Imaging Sciences*, 2023.

[‡]equal contribution

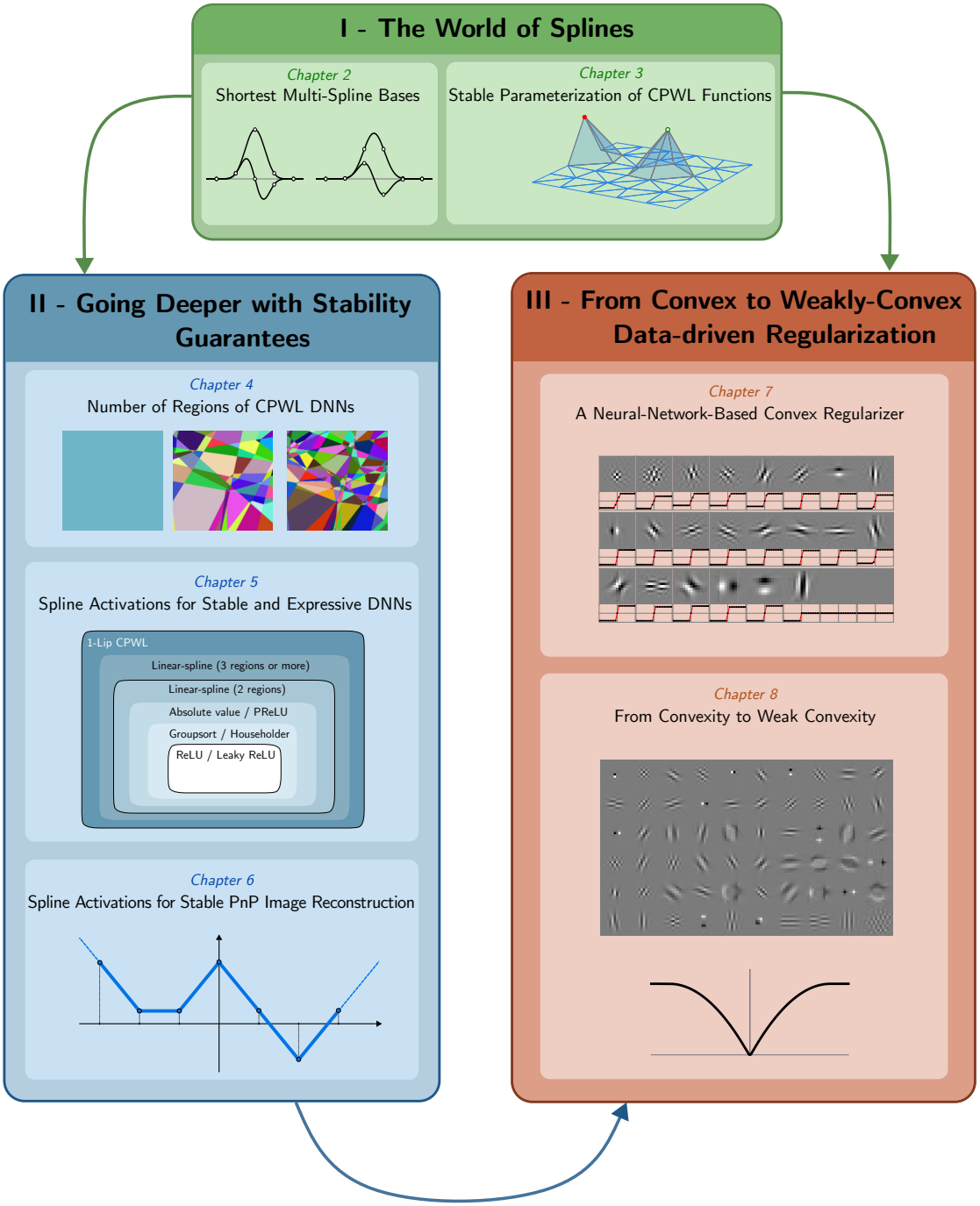

Figure 1.9: General overview of the thesis. Abbreviations: continuous and piecewise linear (CPWL), deep neural network (DNN), plug and play (PnP).

Part I The World of Splines

Stable and Expressive Parameterizations for Low-Dimensional Problems

Our quest for stable, expressive and interpretable parameterizations begins in a simplified setup. The problems of interest are limited to low-dimensional regression problems, allowing us to momentarily leave aside the curse of dimensionality. In this context, and as argued in Chapter 3, the use of DNNs is not necessarily relevant. The parameterized function spaces can simply be chosen as the span of some well-chosen basis functions. To build performant and trustworthy parameterizations, we propose to design spline-based basis functions with special attention put on two desirable properties. First, to ensure good expressivity, we develop parameterizations with approximation properties that are optimal given some criteria. Second, to ensure stability, we use parameterizations with a well-conditioned parameter-to-function map, also referred to as the synthesis operator. Specifically, this is ensured via the design of Riesz bases.

The restriction of this first part to low-dimensional problems will serve in the remainder of the thesis in two aspects. In low dimensions, one can have it all: expressivity, interpretability and stability. This highly contrasts with many deep parameterizations, which lack such theoretical properties. The low-dimensional parameterizations discussed in Part I can thus set a target in terms of trustworthiness when designing parameterizations for high-dimensional problems. On another level, low-dimensional parameterizations can be used to design nonlinear modules within DNNs. In particular, in Part II and III we will show theoretically and empirically how learnable spline activation functions can boost the performance of some trustworthy deep-learning based image reconstruction methods.

2 Shortest-support Multi-Spline Bases

The text of this chapter is adapted from the published article

A. Goujon, S. Aziznejad, A. Naderi and M. Unser "Shortest-support multi-spline bases for generalized sampling", *Journal of Computational and Applied Mathematics*, volume 395, paper 113610, October 2021.

2.1 Summary

In this chapter, we introduce the notion of multi-splines and we discuss their role in general sampling. Generalized sampling consists of the recovery of a function f, from the samples of the responses of a collection of linear shift-invariant systems to the input f. The reconstructed function is typically a member of a finitely generated integer-shift-invariant space that can reproduce polynomials up to a given degree M. While this property allows for an approximation power of order $(M+1)$, it comes with a tradeoff on the length of the support of the basis functions. Specifically, we prove that the sum of the length of the support of the generators is at least $(M+1)$. Following this result, we introduce the notion of shortest basis of degree M, which is motivated by our desire to minimize computational costs. We then demonstrate that any basis of shortest support generates a Riesz basis. Finally, we introduce a recursive algorithm to construct the shortest-support basis for any multi-spline space. It provides a generalization of both polynomial and Hermite B-splines. This framework paves the way for novel applications such as fast derivative sampling with arbitrarily high approximation power.

2.2 Introduction

2.2.1 Generalized Sampling in Shift-Invariant Spaces

Since the formulation of Nyquist-Shannon's celebrated sampling theorem [106], the reconstruction of a function from discrete measurements has been extended in many ways [107, 108]. In particular, Papoulis proposed the framework of generalized sampling [109], where he showed that any bandlimited function f is uniquely determined by the sequences of discrete measurements (generalized samples)

$$g_n(kT) = (h_n * f)(kT) = \langle f, \psi_n(\cdot - kT) \rangle, \quad n = 1, ..., N, \quad k \in \mathbb{Z}, \tag{2.1}$$

where $(g_n(t))_{n=1,...,N}$ are the outcome of N linearly independent systems applied to f. The sampling is assumed to proceed at $1/N$ the Nyquist rate (*i.e.*, $T = NT_{\text{Nyq}} = 2N\pi/\omega_{\max}$, where $\omega_{\max}$ is the maximum frequency of f). The functions $\psi_n(t) = h_n(-t)$, $t \in \mathbb{R}$, are called the analysis functions. They are the time-reversed versions of the impulse responses. The sampling theorem was also generalized to many different function spaces such as integer-shift-invariant spaces [110, 111], including spline spaces [112–114]. Following this extension and Papoulis' theory, Unser and Zerubia introduced a framework to perform generalized sampling without the bandlimited constraint [115, 116] which includes important cases such as interlaced and derivative sampling in spline spaces. In this chapter, we adopt the same framework and propose to reconstruct a function f from discrete samples $g_n(k), k = 1, ..., N$ in an integer-shift-invariant space generated by a finite collection of generators as in some recent works [117–119]. The structure of such reconstruction spaces has been thoroughly studied [120–122] and there exist theoretical results that lead to the critical choice of relevant generating functions [123]. As a minimal requirement to get a good approximation space, the generating functions should satisfy jointly the partition-of-unity condition [124]. In addition, there exists a tradeoff between the approximation power of the space and the size of the support of the generating functions [125].

2.2.2 Polynomial Splines

A polynomial spline is a piecewise polynomial function defined over the real line. Of special interest are the splines of degree n because they provide one free parameter per segment. They are defined by distinct knots and polynomial pieces of degree n that are connected smoothly so that the global function has continuous derivatives up to order $(n-1)$. The splines whose knots are uniformly spaced are called cardinal splines and they are relevant to many applications such as image processing [126]. In the 50s, Isaac Schoenberg laid the foundation of cardinal splines [127, 128] when he showed that the set S_n of cardinal splines of degree n could be generated by a single function [129], the B-spline of degree n. In this chapter we will consider the causal B-spline and denote

it by β_+^n. This simple building block is also the shortest nonzero spline of degree n. Interestingly, the B-splines can be constructed recursively with the relation

$$\beta_+^{n+1} = \beta_+^n * \beta_+^0, \tag{2.2}$$

starting from β_+^0, which is the rectangular window over $[0, 1)$

$$\beta_+^0(x) = \begin{cases} 1, & 0 \leq x < 1 \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

The convolution by β_+^0 can be decomposed in two successive operations: an integration (which transforms a spline of degree n into a spline of degree $(n+1)$) followed by a finite difference (which gives back a compactly supported function). Indeed, $(f * \beta_+^0)(x) = \Delta\{\int_{-\infty}^{x} f(t)\mathrm{d}t\}$, where $\Delta\{f\} = (f(\cdot) - f(\cdot - 1))$ is the finite difference of f. Along with their great reproducing properties and shortest support, B-splines allow an efficient and practical implementation, which is exploited in many fields [130–133].

2.2.3 Multi-Splines

To perform generalized sampling, it is natural to look at multi-spline spaces since they offer additional degrees of freedom. A cardinal multi-spline space is defined as the sum of $N \in \mathbb{N}$ spline spaces: $S_{\mathbf{n}} = S_{n_1} + \cdots + S_{n_N}$, $\mathbf{n} = (n_1, ..., n_N)$ and $n_1 < \cdots < n_N \in \mathbb{N}$. From now on, any spline will be assumed to be a cardinal spline unless stated otherwise. It is worth noting that, in the case of consecutive spaces specified by $n_k = n_1 + (k-1)$, the resulting space is exactly the space of piecewise polynomials of degree n_N that are in $C^{n_1-1}(\mathbb{R})$, the space of functions with $(n_1 - 1)$ continuous derivatives (see Proposition 2.2). Some multi-spline spaces have proved to be of great interest for derivative sampling, where the goal is to reconstruct a signal from the samples of the function and of its first-order derivative. We should mention the well-known bicubic Hermite splines (h_1, h_2), first introduced by Schoenberg and Lipow in [134]. They constitute a basis of $S_2 + S_3$ with the shortest support and provide the direct interpolation formula

$$\forall f \in S_2 + S_3, \quad \forall x \in \mathbb{R} : f(x) = \sum_{k \in \mathbb{Z}} \Big(f(k) h_1(x-k) + f'(k) h_2(x-k) \Big), \tag{2.4}$$

where $f' = f^{(1)}$ is the derivative of f. The excellent approximation capabilities and minimal-support property of the Hermite splines [135] give a strong incentive to investigate more general multi-spline spaces. The bicubic Hermite splines are the backbone of many computer-graphics applications and closely linked to Bézier curves [136–140]. Schoenberg and Lipow also found two fundamental functions to reconstruct any function in $S_4 + S_5$ from its samples and the samples of its first-order derivative. Nonetheless, those functions are not well-suited to practical applications since they are not compactly supported.

Building on top of an impressive body of work from various communities, we propose a systematic study of shortest bases for any multi-spline space. In particular, the main goal is to generalize the concept of B-splines to any multi-spline space.

The chapter is organized as follows: in Section 2.3, we formulate the problem in the framework of finitely generated shift-invariant spaces. We then state the properties that relevant generating functions should satisfy. In Section 2.4, we show that the conditions imposed can only be met if the sum of the support of the generating functions is large enough. In Section 2.5, we present a method to construct shortest-support bases for any multi-spline space. This has important implications in practice, which we illustrate in Section 2.6 where we give practical examples to implement generalized sampling with the new set of functions, including interpolation, derivative sampling, and a new way to envision Bézier curves.

2.3 Formulation of the Problem

Let $\boldsymbol{\phi} = (\phi_1, \phi_2, \ldots, \phi_N)$ be a finite collection of functions in $L_2(\mathbb{R})$, Lebesgue's space of square-integrable functions. The integer-shift-invariant subspace of $L_2(\mathbb{R})$ generated by $\boldsymbol{\phi}$ is denoted by $\mathrm{S}(\boldsymbol{\phi})$ and is defined as

$$\mathrm{S}(\boldsymbol{\phi}) = \mathrm{S}(\phi_1) + \mathrm{S}(\phi_2) + \cdots + \mathrm{S}(\phi_N), \tag{2.5}$$

where

$$\mathrm{S}(\phi_n) = \overline{\mathrm{Span}}\left(\{\phi_n(\cdot - k)\}_{k\in\mathbb{Z}}\right) \subseteq L_2(\mathbb{R}), \quad n = 1, \ldots, N. \tag{2.6}$$

We shall not restrict ourselves to multi-spline spaces for now and rather consider finitely generated integer-shift-invariant spaces. To formulate the problem, we recall three properties of $\boldsymbol{\phi}$ that have been imposed in previous works for practical applications. Multi-spline spaces will then naturally stand out as practical and important reconstruction spaces (Sections III and IV).

2.3.1 Riesz Basis

Definition 2.1. *The set of functions $\{\phi_n(\cdot - k) : k \in \mathbb{Z}, n = 1, \ldots, N\} \subset L_2(\mathbb{R})$ is said to be a Riesz basis with bounds $A, B \in \mathbb{R}$ with $0 < A \leq B < +\infty$ if, for any vector of square-summable sequences $\boldsymbol{c} = (c_1, ..., c_N) \in (\ell_2(\mathbb{Z}))^N$, we have that*

$$A \left\|\boldsymbol{c}\right\|_{\ell_2} \leq \left\| \sum_{k\in\mathbb{Z}} \boldsymbol{c}[k]^T \boldsymbol{\phi}(\cdot - k) \right\|_{L_2(\mathbb{R})} \leq B \left\|\boldsymbol{c}\right\|_{\ell_2}, \tag{2.7}$$

where $\|\boldsymbol{c}\|_{\ell_2} = \left(\sum_{n=1}^{N} \|c_n\|_{\ell_2}^2\right)^{\frac{1}{2}}$, $\boldsymbol{\phi} = (\phi_1, \phi_2, \ldots, \phi_N)$ *and where* A *and* B *are the tightest constants.*

When this property is satisfied, we say that $\boldsymbol{\phi}$ generates a Riesz basis. The Riesz-basis property guarantees that any $f \in \mathrm{S}(\boldsymbol{\phi})$ has the unique and stable representation ([141])

$$f(\cdot) = \sum_{k\in\mathbb{Z}} \boldsymbol{c}[k]^T \boldsymbol{\phi}(\cdot - k) = \sum_{k\in\mathbb{Z}} \sum_{n=1}^{N} c_n[k]\phi_n(\cdot - k). \tag{2.8}$$

This property is well characterized in the Fourier domain via the Gramian matrix-valued function

$$\hat{\boldsymbol{G}}(\omega) = \sum_{k\in\mathbb{Z}} \hat{\boldsymbol{\phi}}(\omega + 2k\pi)\hat{\boldsymbol{\phi}}(\omega + 2k\pi)^H = \sum_{k\in\mathbb{Z}} \langle \boldsymbol{\phi}, \boldsymbol{\phi}^T(\cdot - k)\rangle \mathrm{e}^{-\mathrm{j}\omega k}, \tag{2.9}$$

where the inner product is defined as $\langle f, g\rangle = \int_{\mathbb{R}} f(t)g^*(t)\mathrm{d}t$, * is the complex conjugate operator, and ^{H} is the conjugate transpose operator. Equality (2.9) follows from Poisson's formula applied to the sampling at the integers of the matrix-valued autocorrelation function $t \mapsto \langle \boldsymbol{\phi}, \boldsymbol{\phi}^T(\cdot - t)\rangle = (\boldsymbol{\phi} * \boldsymbol{\phi}^{H\vee})(t)$[142]. The Fourier equivalent of the Riesz-basis condition is [121]

$$0 < A^2 = \operatorname*{ess\,inf}_{\omega\in[0,2\pi)} \lambda_{\min}(\omega) \leq \operatorname*{ess\,sup}_{\omega\in[0,2\pi)} \lambda_{\max}(\omega) = B^2 < +\infty, \tag{2.10}$$

where $\lambda_{\min}(\omega)$ and $\lambda_{\max}(\omega)$ are the smallest and largest eigenvalues of $\hat{\boldsymbol{G}}(\omega)$.

2.3.2 Reproducing Polynomials

Definition 2.2. *The space* $\mathrm{S}(\boldsymbol{\phi})$ *is said to reproduce polynomials of degree up to* M *if, for all* $m = 0, 1, ..., M$*, there exist vector sequences* $\boldsymbol{c}_m$ *(not necessarily in* $(\ell_2(\mathbb{Z}))^N$*) such that* [*]

$$\forall x \in \mathbb{R}, \quad x^m = \sum_{k\in\mathbb{Z}} \boldsymbol{c}_m[k]^T \boldsymbol{\phi}(x - k). \tag{2.11}$$

Strang and Fix showed that the property of the reproduction of polynomials of degree up to M is directly linked to the approximation power of the reconstruction space [143]. More precisely, let

$$\mathrm{S}_h(\boldsymbol{\phi}) = \{f(\cdot/h) : f \in \mathrm{S}(\boldsymbol{\phi})\} \tag{2.12}$$

be the h-dilate of $\mathrm{S}(\boldsymbol{\phi})$. The space $\mathrm{S}(\boldsymbol{\phi})$ is said to have an approximation power of order M if any sufficiently smooth and decaying function can be approached by an element of $\mathrm{S}_h(\boldsymbol{\phi})$ with an error decaying as $O(h^M)$. The so called "Strang-Fix conditions" give

[*]for $m = 0$, we use in (2.11) the convention that $x^m = 1$, including for $x = 0$.

sufficient conditions to have a space with an approximation power of order M [135, 144, 145]. In particular, for compactly supported and integrable generating functions, it is sufficient to have the space $\mathrm{S}(\boldsymbol{\phi})$ reproduce polynomials of degree up to $(M-1)$. A straightforward implication is that the spline space S_n has an approximation power of order $(n+1)$ since

(i) it can reproduce polynomials of degree up to n;

(ii) it can be generated by the compactly supported function β_+^n.

The multi-spline space $S_{n_1} + \cdots + S_{n_N}$ inherits the highest approximation power of its spline spaces. Its approximation power is $(n_N + 1)$, since $S_{n_N} \subset S_{n_1} + \cdots + S_{n_N}$.

2.3.3 Compact Support

The evaluation of $f \in \mathrm{S}(\boldsymbol{\phi})$ at a given $x \in \mathbb{R}$ from its discrete representation $\boldsymbol{c} \in (\ell_2(\mathbb{Z}))^N$ requires a number of computations more or less proportional to the support size of $\boldsymbol{\phi}$. So, ideally, we want to minimize the support of $\boldsymbol{\phi}$ while maintaining a good approximation power [125]. The support of a function $f \in L_2(\mathbb{R})$ is written as $\operatorname{supp} f = \overline{\{x \in \mathbb{R} : f(x) \neq 0\}}$. If it is a compact subset of $\mathbb{R}$, then the support size is defined as $|\mathrm{supp}(f)| = \int_{\mathbb{R}} \mathbb{1}_{\operatorname{supp} f}(t)\mathrm{d}t$, where $\mathbb{1}_{\operatorname{supp} f}$ is the indicator function of $\operatorname{supp} f$. For a finite collection of compactly supported functions $\boldsymbol{\phi} = (\phi_1, ..., \phi_N)$, the natural extension for the support size is

$$|\mathrm{supp}(\boldsymbol{\phi})| = \sum_{n=1}^{N} |\mathrm{supp}(\phi_n)|. \tag{2.13}$$

In Section 2.4, we present theoretical results that clarify the relation between the desired properties.

2.4 Shortest Bases

For a single generator ϕ such that $\mathrm{S}(\phi)$ reproduces polynomials of degree up to M, Schoenberg stated that $|\mathrm{supp}(\phi)| \geq M+1$ [127]. The result was proved in [146] for $N = 2$. We now extend the proof to any $N \in \mathbb{N} \setminus \{0\}$.

Theorem 2.1 (Minimal support)**.** *If* $\mathrm{S}(\boldsymbol{\phi}) = \mathrm{S}(\phi_1, \phi_2, \ldots, \phi_N)$ *reproduces polynomials of degree up to* M*, then* $|\mathrm{supp}(\boldsymbol{\phi})| \geq M + 1$*. In addition, if there is equality, then*

$$\sum_{k\in\mathbb{Z}} \sum_{n=1}^{N} \mathbb{1}_{\operatorname{supp}\phi_n}(x+k) = |\mathrm{supp}(\boldsymbol{\phi})| \quad \textit{for almost every } x \in \mathbb{R}. \tag{2.14}$$

Proof. If $\boldsymbol{\phi}$ is not compactly supported, then the inequality is clear. Now, we can assume that $\boldsymbol{\phi}$ is compactly supported. This implies that, for any $x \in \mathbb{R}$, the sum $\sum_{k\in\mathbb{Z}} \boldsymbol{c}[k]^T \boldsymbol{\phi}(x-k) = \sum_{k\in\mathbb{Z}} \sum_{n=1}^{N} c_n[k]\phi_n(x-k)$ has only a finite number of nonzero terms that are identified by the set

$$\Lambda(x) = \{(n,k) \in \{1,\dots,N\} \times \mathbb{Z} : \quad x \in \operatorname{supp} \phi_n(\cdot - k)\}, \tag{2.15}$$

and its cardinality

$$\lambda(x) = \#(\Lambda(x)) = \sum_{k\in\mathbb{Z}} \sum_{n=1}^{N} \mathbb{1}_{\operatorname{supp}\phi_n}(x+k) \in \mathbb{N}. \tag{2.16}$$

Equation (2.16) follows from the fact that $\mathbb{1}_{\operatorname{supp}\phi_n}(x+k)$ is 1 if and only if $(n,k) \in \Lambda(x)$ and 0 otherwise. The function $x \mapsto \lambda(x)$ is 1-periodic and bounded because $\operatorname{supp}\phi_n$ are compact subsets of $\mathbb{R}$. Its average over one period reads (note that the sums are in fact all finite)

$$\overline{\lambda} = \int_0^1 \sum_{n=1}^{N} \sum_{k\in\mathbb{Z}} \mathbb{1}_{\operatorname{supp}\phi_n}(x+k)\mathrm{d}x = \sum_{n=1}^{N} \sum_{k\in\mathbb{Z}} \int_0^1 \mathbb{1}_{\operatorname{supp}\phi_n}(x+k)\mathrm{d}x \tag{2.17}$$

$$= \sum_{n=1}^{N} \int_{-\infty}^{\infty} \mathbb{1}_{\operatorname{supp}\phi_n}(x)\mathrm{d}x = |\operatorname{supp}(\boldsymbol{\phi})|, \tag{2.18}$$

where we applied Fubini's Theorem in (2.17). Because λ is bounded and takes values in $\mathbb{N}$, it only takes a finite number of values. Consequently, there exists a set $\mathrm{A} \subset [0,1]$ of nonzero measure such that λ is constant on A and no greater than its average, as in

$$\forall x \in A : \quad \lambda(x) = \lambda_A \leq \overline{\lambda} = |\operatorname{supp}(\boldsymbol{\phi})|. \tag{2.19}$$

The function $\#(\Lambda)$ restricted to A is constant, but this does not imply that Λ is constant on A. Noting that A is bounded and that the ϕ_n are compactly supported, the image of A under Λ, denoted by $\Lambda(A)$, is a finite set. Therefore, there exists $\mathrm{B} \subset \mathrm{A} \subset [0,1]$ of nonzero measure such that Λ is constant on B. This means that the set $\mathrm{S}(\boldsymbol{\phi})_{|B}$ of functions of $\mathrm{S}(\boldsymbol{\phi})$ restricted to B is spanned by λ_A functions $(\phi_n(\cdot - k))_{(n,k)\in\Lambda(B)}$.
Moreover, due to the reproducing property, the polynomials of degree up to M restricted to B form a linear subspace of $\mathrm{S}(\boldsymbol{\phi})_{|B}$ whose dimension is $(M+1)$, because B is infinite. Then, we must have that $\lambda_A \geq M+1$ and, since $\lambda_A \leq |\operatorname{supp}(\boldsymbol{\phi})|$, we deduce the announced bound $|\operatorname{supp}(\boldsymbol{\phi})| \geq M+1$.
If λ is not *a.e.* constant, then A can be chosen so that $\lambda_A < \overline{\lambda} = |\operatorname{supp}(\boldsymbol{\phi})|$ and $\mathrm{S}(\boldsymbol{\phi})_{|B}$ is spanned by fewer than $|\operatorname{supp}(\boldsymbol{\phi})|$ functions. The reproduction property implies that $|\operatorname{supp}(\boldsymbol{\phi})| > M+1$. This means that the equality $|\operatorname{supp}(\boldsymbol{\phi})| = M+1$ is possible only if λ is *a.e.* constant. □

Following Theorem 2.1, we can introduce the central notion of shortest-support basis.

Definition 2.3. *A collection of functions $\boldsymbol{\phi} \in (L_2(\mathbb{R}))^N$ is said to be a shortest-support basis of degree M if $\mathrm{S}(\boldsymbol{\phi})$ reproduces polynomials of degree up to M with the shortest support, i.e. with $|\mathrm{supp}(\boldsymbol{\phi})| = M + 1$.*

The qualifier of basis comes from Theorem 2.2.

Theorem 2.2 (Shortest support and Riesz basis)**.** *Any shortest basis generates a Riesz basis.*

Before proving the theorem, we define the kth slice of any function f as

$$\forall x \in \mathbb{R}: \quad \mathrm{S}_k\{f\}(x) = \begin{cases} f(x+k), & x \in [0,1) \\ 0, & \text{otherwise,} \end{cases} \tag{2.20}$$

and the set of nonzero slices of all the generating functions as

$$\mathcal{T}(\boldsymbol{\phi}) = \{\mathrm{S}_k\{\phi_n\} : \mathrm{S}_k\{\phi_n\} \not\equiv 0 \text{ and } k \in \mathbb{Z}, n = 1, ..., N\}. \tag{2.21}$$

The proof will also invoke Lemma 2.1.

Lemma 2.1. *Let $\boldsymbol{\phi} \in (L_2(\mathbb{R}))^N$ be compactly supported. If $\mathcal{T}(\boldsymbol{\phi})$ is a set of linearly independent functions, then $\boldsymbol{\phi}$ generates a Riesz basis.*

Proof. The generating functions can be expressed in terms of their slices as $\phi_n(x) = \sum_{k\in\mathbb{Z}} \mathrm{S}_k\{\phi_n\}(x-k)$. The Riesz-basis property is best characterized in the Fourier domain with the Gramian matrix (note that, $\boldsymbol{\phi}$ being compactly supported, all the sums are in fact finite), which leads to

$$\begin{aligned}
(\hat{\boldsymbol{G}}(\omega))_{mn} &= \sum_{q\in\mathbb{Z}} \langle \phi_m, \phi_n(\cdot - q)\rangle \mathrm{e}^{-\mathrm{j}\omega q} \\
&= \sum_{q\in\mathbb{Z}} \sum_{k_1\in\mathbb{Z}} \sum_{k_2\in\mathbb{Z}} \langle \mathrm{S}_{k_1}\{\phi_m\}, \mathrm{S}_{k_2}\{\phi_n\}(\cdot - q - (k_2 - k_1))\rangle \mathrm{e}^{-\mathrm{j}\omega q} \\
&= \sum_{k_1\in\mathbb{Z}} \sum_{k_2\in\mathbb{Z}} \langle \mathrm{S}_{k_1}\{\phi_m\}, \mathrm{S}_{k_2}\{\phi_n\}\rangle \mathrm{e}^{\mathrm{j}\omega(k_2-k_1)} \\
&= \langle \sum_{k_1\in\mathbb{Z}} \mathrm{S}_{k_1}\{\phi_m\} \mathrm{e}^{-\mathrm{j}\omega k_1}, \sum_{k_2\in\mathbb{Z}} \mathrm{S}_{k_2}\{\phi_n\} \mathrm{e}^{-\mathrm{j}\omega k_2} \rangle \\
&= \langle \tilde{\phi}_m(\omega, \cdot), \tilde{\phi}_n(\omega, \cdot)\rangle,
\end{aligned} \tag{2.22}$$

where $\tilde{\phi_n}(\omega, \cdot)$ is the finite weighted sum of slices

$$\tilde{\phi_n}(\omega, x) = \sum_{k\in\mathbb{Z}} \mathrm{S}_k\{\phi_n\}(x) e^{-j\omega k}. \tag{2.23}$$

If, now, $\mathcal{T}(\boldsymbol{\phi})$ is a set of linearly independent functions, then, for any $\omega \in \mathbb{R}$, the functions $(\tilde{\phi}_n(\omega, \cdot))_{n=1,\dots,N}$ are linearly independent because the sums are finite. This means that $\hat{\boldsymbol{G}}(\omega)$ is the Gramian matrix of a linearly independent family of functions, which is known to be equivalent to $\det \hat{\boldsymbol{G}}(\omega) > 0$. In addition $g : \omega \mapsto \det(\hat{\boldsymbol{G}}(\omega))$ is a finite weighted sum of $\mathrm{e}^{\mathrm{j}\omega k}$ since $\boldsymbol{\phi}$ is compactly supported. It is therefore continuous and 2π-periodic. The image of $[0, 2\pi]$ under g is therefore a closed interval such that

$$0 < \operatorname*{ess\,inf}_{\omega\in[0,2\pi]} \min_{\omega\in[0,2\pi]} \det(\hat{\boldsymbol{G}}(\omega)) < \max_{\omega\in[0,2\pi]} \det(\hat{\boldsymbol{G}}(\omega)) < +\infty. \tag{2.24}$$

Noting that $\det(\hat{\boldsymbol{G}}(\omega))$ is the product of the eigenvalues of $\hat{\boldsymbol{G}}(\omega)$, Condition (2.10) is satisfied, which means that $\boldsymbol{\phi}$ is a Riesz basis. □

Note that the converse of Lemma 2.1 is not necessarily true. For a counterexample, consider the function in (2.25) made of two side-by-side rectangles of different height, so that

$$\forall x \in \mathbb{R} : \phi(x) = \begin{cases} 1, & x \in [0, 1) \\ \alpha, & x \in [1, 2) \\ 0, & \text{otherwise.} \end{cases} \tag{2.25}$$

In this case, with a single generator, the Gramian matrix is just a scalar and reads $\hat{g}(\omega) = (1 + \alpha^2) + 2\alpha \cos \omega$, which verifies, for any $\omega \in \mathbb{R}$, that

$$(1 - |\alpha|)^2 \leq |\hat{g}(\omega)| \leq (1 + |\alpha|)^2. \tag{2.26}$$

So for $|\alpha| \neq 1$, ϕ is a Riesz basis with bound $A = (1 - |\alpha|)$ and $B = (1 + |\alpha|)$. Yet, $\mathcal{T}(\boldsymbol{\phi})$ is clearly not a set of linearly independent functions since the second slice is a scaled version of the first one. For a more practical counterexample, see [147, Proposition 2.2.].

Lemma 2.2. *Let $\boldsymbol{\phi} \in (L_2(\mathbb{R}))^N$. If $\boldsymbol{\phi}$ is a shortest-support basis, then $\mathcal{T}(\boldsymbol{\phi})$ is a set of linearly independent functions.*

Proof. It is equivalent to prove the contrapositive of the lemma, which states that if $\mathcal{T}(\boldsymbol{\phi})$ is not a set of linearly independent functions, then $\boldsymbol{\phi}$ is not a shortest-support basis. To that end, suppose that $\mathcal{T}(\boldsymbol{\phi})$ is not a set of linearly independent functions. This means that one can find a slice, say $\mathrm{S}_{k_0}\{\phi_{q_0}\}$, that depends linearly on the others. Now, consider the integer-shift-invariant space generated by the set of functions $\mathcal{T}(\boldsymbol{\phi})\backslash\{\mathrm{S}_{k_0}\{\phi_{q_0}\}\}$. Note that the new generating functions differ now both in size (support size of at most 1) and in number (possibly greater than N). On one hand, the new integer-shift-invariant space is larger than the initial space and, in particular, is still able to reproduce polynomials of degree up to M. On the other hand, the sum of the support size of the generating functions is smaller than $|\mathrm{supp}(\boldsymbol{\phi})|$ because a nonzero slice was removed. So, $\boldsymbol{\phi}$ cannot be of minimal support. □

We can now prove Theorem 2.2.

Proof of Theorem 2.2. Let $\boldsymbol{\phi} \in (L_2(\mathbb{R}))^N$ be compactly supported. By contraposition, if it is not a Riesz basis, then $\mathcal{T}(\boldsymbol{\phi})$ is not a set of linearly independent functions (Lemma 2.1). Then, by Lemma 2.2, $\boldsymbol{\phi}$ cannot be of minimal support. □

To conclude this section, we present two results for finitely generated integer-shift-invariant spaces in preparation to a characterization of multi-spline spaces (Theorem 2.4). The unit sample sequence is written $\delta[\cdot]$ and is defined by $\delta[k] = \begin{cases} 1, & k = 0 \\ 0, & k \neq 0 \end{cases}$, and its matrix version $\boldsymbol{\delta}_{N\times N}$ is defined by $(\boldsymbol{\delta}_{N\times N})_{pq}[\cdot] = \begin{cases} \delta[\cdot], & p = q \\ 0, & p \neq q \end{cases}$.

Lemma 2.3. *Let $N, M \in \mathbb{N}$, $\boldsymbol{C} \in (\mathbb{R}^{\mathbb{Z}})^{N\times M}$, $\boldsymbol{B} \in (\mathbb{R}^{\mathbb{Z}})^{M\times N}$. If the sequence of matrices $\boldsymbol{B}$ is compactly supported and $\boldsymbol{C} * \boldsymbol{B} = \boldsymbol{\delta}_{N\times N}$, then $M \geq N$.*

Proof. There exists $s \in \mathbb{N}$ such that $\operatorname{supp} \boldsymbol{B} \subset \{-s, ..., s\} \subset \mathbb{N}$. The behavior of $\boldsymbol{C}[k]$ when $|k| \to \infty$ is not known, and it is easier to work with the truncated version $\boldsymbol{C}_m = \mathbb{1}_{\{-s,...,ms\}} \times \boldsymbol{C}$, where $m \in \mathbb{N}$ is a large enough integer $m > 2N + 1$. The sequence of matrices $\boldsymbol{C}_m * \boldsymbol{B}$ is compactly supported and satisfies $\operatorname{supp} \boldsymbol{C}_m * \boldsymbol{B} \subset \{-2s, ..., (m+1)s\}$. Following the properties of convolution of compact sequences, we have, for any $k = 0, ..., (m-1)s$, that $\boldsymbol{C}_m * \boldsymbol{B}[k] = \boldsymbol{C} * \boldsymbol{B}[k] = \boldsymbol{\delta}_{N\times N}[k]$. Therefore, one can write that

$$\boldsymbol{C}_m * \boldsymbol{B} = \boldsymbol{\delta}_{N\times N} + \sum_{\substack{-2s\leq k<0 \\ (m-1)s+1\leq k\leq (m+1)s}} \mathbf{M}_k \delta[\cdot - k], \tag{2.27}$$

where $\mathbf{M}_k \in \mathbb{R}^{N\times N}$ are matrices that account for the fact that $\boldsymbol{C}_m$ is a truncated version of $\boldsymbol{C}$. This then translates into the following z-transform matrix relation (note that all sequences are compactly supported so the z-transforms are well-defined)

$$\hat{\boldsymbol{C}}_m(z)\hat{\boldsymbol{B}}(z) = \mathbf{I}_{N\times N} + \sum_{\substack{-2s\leq k<0 \\ (m-1)s<k\leq (m+1)s}} z^{-k}\mathbf{M}_k \tag{2.28}$$

$$= \mathbf{M}_{N\times N} + \sum_{k=-2s}^{-1} z^{-k}\mathbf{M}_k + \sum_{k=(m-1)s+1}^{(m+1)s} z^{-k}\mathbf{M}_k \tag{2.29}$$

$$= z^{-2s}\boldsymbol{A}(z), \tag{2.30}$$

where $\boldsymbol{A}(z)$ can be decomposed as

$$\boldsymbol{A}(z) = z^{2s}\mathbf{I}_{N\times N} + \boldsymbol{P}(z) + z^{(m+1)s+1}\boldsymbol{Q}(z), \tag{2.31}$$

where $\boldsymbol{P}(z)$ and $\boldsymbol{Q}(z)$ are polynomial matrices of degree $(2s-1)$. The determinant of $\boldsymbol{A}(z)$ can be expressed in terms of the columns of $\mathbf{I}_{N\times N}, \boldsymbol{P}(z)$, and $\boldsymbol{Q}(z)$ (denoted respectively $\mathbf{e}_k, \boldsymbol{p}_k(z)$, and $\boldsymbol{q}_k(z)$), so that

$$z \mapsto \det A(z) = \det(z^{2s}\mathbf{e}_1 + \boldsymbol{p}_1(z) + z^{(m+1)s}\boldsymbol{q}_1(z), \dots, z^{2s}\mathbf{e}_N + \boldsymbol{p}_N(z) + z^{(m+1)s}\boldsymbol{q}_N(z)). \tag{2.32}$$

Knowing that the determinant is n-linear with respect to the columns, $z \mapsto \det \boldsymbol{A}(z)$ is a polynomial function of degree at most $(m+3)sN$. We now want to prove that it cannot be identically zero. To that end, we expand the determinant with respect to the columns and find that there is a unique term of the form λz^{2sN}. It is obtained by picking for $k = 1, \dots, N$ the column $\mathbf{e}_k z^{2s}$. The coefficient in front of z^{2sN} is therefore $\det(\mathbf{e}_1, \dots, \mathbf{e}_N) = 1 \neq 0$. Indeed, for other combinations of columns in the expansion, we would have that

- if at least one column of the form $z^{(m+1)s}\boldsymbol{q}_k(z)$ is chosen, then it results in a term of degree at least $(m+1)s > (2N+2)s > 2sN$;
- else, at least one column of the form $\boldsymbol{p}_k(z)$ is chosen. Since the degree of $\boldsymbol{p}_k(z)$ is lower than $2s$, the resulting term in the expansion has a degree lower than $2sN$.

In the end, we proved that $z \mapsto \det \boldsymbol{A}(z)$ cannot be identically zero. Therefore, there exists $z_0 \in \mathbb{R}$ so that $\operatorname{rank}(\boldsymbol{A}(z_0)) = N$. It implies that $N = \operatorname{rank}(\hat{\boldsymbol{C}}_m(z_0)\hat{\boldsymbol{B}}(z_0)) \leq \min(\operatorname{rank}(\hat{\boldsymbol{C}}_m(z_0)), \operatorname{rank}(\hat{\boldsymbol{B}}(z_0))) \leq \min(M, N) \leq M$. □

Lemma 2.4. *Let $\boldsymbol{\psi} \in (L_2(\mathbb{R}))^M$ and $\boldsymbol{\eta} \in (L_2(\mathbb{R}))^N$ be two collections of compactly supported functions that are able to reproduce each other (the reproducing sequences might not be in $\ell_2(\mathbb{Z})$). If $\boldsymbol{\eta}$ is a shortest-support basis, then $M \geq N$.*

Proof. By hypothesis, there exist vector sequences $\boldsymbol{c}_p \in (\mathbb{R}^{\mathbb{Z}})^M$ such that

$$\eta_p = \sum_{k\in\mathbb{Z}} \boldsymbol{c}_p[k]^T \boldsymbol{\psi}(\cdot - k) = \boldsymbol{c}_p^T * \boldsymbol{\psi},$$

which reads in matrix form

$$\boldsymbol{\eta} = \boldsymbol{C} * \boldsymbol{\psi}, \quad \boldsymbol{C} \in (\mathbb{R}^{\mathbb{Z}})^{N\times M}. \tag{2.33}$$

Similarly, one can write that

$$\boldsymbol{\psi} = \boldsymbol{B} * \boldsymbol{\eta}, \quad \boldsymbol{B} \in (\mathbb{R}^{\mathbb{Z}})^{M\times N}. \tag{2.34}$$

From Lemma 2.2, we know that the nonzero slices of $\boldsymbol{\eta}$ are linearly independent (shortest-support basis). This implies that, to generate the compactly supported function $\boldsymbol{\psi}$, the sequence of matrices $\boldsymbol{B}$ must be compactly supported as well since the only way to

generate the zero function on a segment for $\boldsymbol{\eta}$ is to set the active coefficient of $\boldsymbol{B}$ to 0. Now, one can mix the equations and find that

$$\boldsymbol{\eta} = \boldsymbol{C} * (\boldsymbol{B} * \boldsymbol{\eta}) = (\boldsymbol{C} * \boldsymbol{B}) * \boldsymbol{\eta}. \tag{2.35}$$

The associativity of the convolution operations is justified by the fact that both $\boldsymbol{\eta}$ and $\boldsymbol{B}$ are compactly supported, meaning that, for a given argument x, all sums are finite. Because the slices of $\boldsymbol{\eta}$ are linearly independent, $\boldsymbol{\eta}$ can reproduce itself in a unique way, which gives

$$\boldsymbol{C} * \boldsymbol{B} = \boldsymbol{\delta}_{N\times N}, \tag{2.36}$$

We can now conclude that $M \geq N$ with Lemma 2.3. □

2.5 Multi-Spline Shortest Bases

With a single generator, the unique shortest basis of degree $n \in \mathbb{N}$ (up to a scaling and a shift operation) is the B-spline of degree n, which is a generator of S_n. For multiple generators, it is natural to consider spaces generated by a finite number of B-splines $\boldsymbol{\beta}_{\mathbf{n}} = \left(\beta_+^{n_1}, ..., \beta_+^{n_N}\right)$, where $\mathbf{n} = (n_1, \ldots, n_N)$ and $n_1 < \ldots < n_N$. In this way, the reproducing and approximation properties are inherited from the higher-degree spline $\beta_+^{n_N}$. Yet, multi-spline spaces are not generated optimally by the classical B-splines.

Proposition 2.1. *Let $N \in \mathbb{N} \setminus \{0\}$ and $\mathbf{n} = (n_1, \ldots, n_N)$ with $n_1 < \cdots < n_N \in \mathbb{N}$. If $N > 1$, then $\boldsymbol{\beta}_{\mathbf{n}} = \left(\beta_+^{n_1}, ..., \beta_+^{n_N}\right)$ is neither a shortest-support basis nor a Riesz basis.*

Proof. • The space $\mathrm{S}(\boldsymbol{\beta}_{\mathbf{n}})$ can reproduce polynomials of degree at most n_N due to the inclusion $\mathrm{S}(\beta_+^{n_N}) \subset \mathrm{S}(\boldsymbol{\phi})$. Moreover, the sum of the support of $\boldsymbol{\beta}_{\mathbf{n}}$ is $\sum_{m=1}^{N}(n_m + 1) > n_N + 1$, which shows that the basis is not a shortest-support one.

• From the proof of Lemma 2.1, the Gramian matrix can be written

$$(\hat{\boldsymbol{G}}(\omega))_{pq} = \langle \tilde{\beta}^{n_p}(\omega, \cdot), \tilde{\beta}^{n_q}(\omega, \cdot) \rangle, \tag{2.37}$$

where $\tilde{\beta}^{n_p}(\omega, \cdot)$ is the finite weighted sum of slices

$$\tilde{\beta}^{n_p}(\omega, x) = \sum_{k\in\mathbb{Z}} \mathrm{S}_k\{\beta^{n_p}\}(x)\mathrm{e}^{-\mathrm{j}\omega k}. \tag{2.38}$$

It is known that $\beta_+^{n_p}$ satisfies the partition of unity, meaning that, for any $x \in \mathbb{R}, \sum_{k\in\mathbb{Z}} \beta_+^{n_p}(x - k) = 1$. In terms of slices, it means that $\tilde{\beta}^{n_p}(0, x) = \mathbb{1}_{[0,1)}(x)$. The functions $(\tilde{\beta}^{n_p}(0, \cdot))_{p=1,...,N}$ are therefore not linearly independent (because they are equal) and $\det \hat{\boldsymbol{G}}(0) = 0$. As stated in the proof of Lemma 2.1, $\omega \mapsto \det \hat{\boldsymbol{G}}(\omega)$ is a

continuous function (because the B-splines are compactly supported), meaning that

$$\operatorname*{ess\,inf}_{\omega\in[0,2\pi]} \det \hat{\boldsymbol{G}}(\omega) = \min_{\omega\in[0,2\pi]} \det \hat{\boldsymbol{G}}(\omega) = 0. \tag{2.39}$$

Following (2.10), $\boldsymbol{\beta}_{\mathbf{n}}$ cannot be a Riesz basis.

□

For $N > 1$, only few shortest bases are known, with the most prominent being the Hermite splines presented by Lipow and Schoenberg [134]. They are solution of the direct interpolation problem

$$\text{find } \eta_p \in S_{n,N} : \eta_p^{(\nu)}(k) = \begin{cases} 1, & \text{if } \nu = p \text{ and } k = 0, \\ 0, & \text{otherwise,} \end{cases} \tag{2.40}$$

with $k \in \mathbb{Z}, \quad \nu, p = 0, ..., (N-1)$, and $S_{n,N} = S_n + \cdots + S_{n+N-1}$. The function η_p has all its derivatives set to zero at the integers, except for the pth derivative that is one at zero. The multi-spline space must be chosen so that η_p is sufficiently differentiable, yielding the condition $n \geq N$. When $n = N$, shortest-support functions were found (the Hermite splines, see plots [148] for instance) but, unfortunately, in a higher-order approximation space, *i.e.* for $n > N$, the functions are not compactly supported anymore. For instance, for derivative sampling (interpolate f and f'), the smaller order of approximation solution ($N = 2$) is given by the cubic Hermite-spline generators of $S_2 + S_3$.

2.5.1 Consecutive Multi-Spline Spaces

The derivatives up to order $(n-1)$ of a compactly-supported spline of degree n must vanish on the edges of the support. This constraint cannot be satisfied if the function is too short. In particular, the shortest nonzero function of S_n has a support size of $(n+1)$ and, interestingly, it is precisely the B-spline of degree n. In the special case of a consecutive multi-spline space $S_{n,N} = S_n + S_{n+1} + \cdots + S_{n+N-1}$, this result can be directly extended. To that end, we define the space

$$P_m^{m'} = \{p \in C^{m'}(\mathbb{R}) : p \text{ is a polynomial of degree } m \text{ on each } [k, k+1), k \in \mathbb{Z}\}. \tag{2.41}$$

Note that the space $P_m^{m'}$ can be viewed as a spline space with knots of multiplicity $(m - m' - 1)$ ([149, Section 5.11]). In our setting with simple knots, $P_m^{m'}$ is rather regarded as multi-spline space (Proposition 2.2).

Proposition 2.2. *Let $n, N > 0$. Then $S_{n,N} = P_{n+N-1}^{n-1}$.*

Proof. The definition of a spline of degree n implies that, for $q = 0, ..., N-1$, we have that $S_{n+q} \subset P_{n+N-1}^{n-1}$, from which we deduce that $S_{n,N} \subset P_{n+N-1}^{n-1}$.

The other inclusion is proven by induction over N, with the induction hypothesis

$$H_N : \forall n \in \mathbb{N}, P_{n+N-1}^{n-1} \subset S_{n,N}. \tag{2.42}$$

- For $N = 1$ and any $n \in \mathbb{N} \setminus \{0\}$, the result is directly given by the definition of $S_{n,1} = S_n = P_n^{n-1}$.
- Suppose that H_N holds for $N \in \mathbb{N}^*$. Let $p \in P_{n+N}^{n-1}$. We have that $p^{(n-1)} \in P_{N+1}^0$ and, consequently, $p^{(n)}$ is a piecewise polynomial function with finite jumps at the knots. There exists $f_0 \in S_0$ that has the same jumps on the knots as $p^{(n)}$. Then, $(p^{(n)} - f_0)$ is continuous on the integers, which implies that $(p^{(n)} - f_0) \in P_N^0$. The induction hypothesis guarantees that $(p^{(n)} - f_0) \in S_{1,N}$ and, therefore, that $p^{(n)} \in S_{0,N+1}$. After n integrations, we finally have that $p \in S_{n,N+1}$, which concludes the induction step and the proof.

□

For a given $L \in \mathbb{N}$, the space of functions in $P_m^{m'}$ that are supported in $[0, L]$ is a vector space of the known finite dimension [150]

$$\dim(\{p \in P_m^{m'} : \operatorname{supp} p \subset [0, L]\}) = ((m - m')L - (m' + 1))_+, \tag{2.43}$$

where $x_+ = \max(0, x)$. Indeed, any $p \in P_m^{m'}$ supported in $[0, L]$ is uniquely defined by L pieces that are polynomials of degree m. So, $L \times (m + 1)$ coefficients have to be set. The smoothness constraints imply that the pieces cannot be set independently. On the first interval $[0, 1)$, the $(m + 1)$ coefficients must be chosen so that $p^{(0)}, ..., p^{(m')}(0) = 0$, which leaves $(m - m')$ degrees of freedom. For the next interval, $(m + 1)$ new coefficients have to be set but the values $p^{(0)}(1), ..., p^{(m')}(1)$ are already fixed, giving only $(m - m')$ new degrees of freedom. We see that each interval provides $(m - m')$ extra degrees of freedom. In the end, there remain LN degrees of freedom. Now, to enforce that $p \in P_m^{m'}$, we must have that $p^{(0)}(L), ..., p^{(m')}(L) = 0$. The total number of degrees of freedom gives the announced dimension $((m - m')L - (m' + 1))_+$.

Corollary 2.1. *Let $n, N, L \in \mathbb{N}$. The set of functions of $S_{n,N}$ that have their support in $[0, L]$ is a vector space of dimension $(LN - n)_+ = \max(0, LN - n)$.*

Corollary 2.2. *Let the Euclidean division of n by N be written as $n = pN + r$. Then, the shortest-support nonzero functions of $S_{n,N}$ have a support size of $(p + 1)$. Moreover, the set $\{f \in S_{n,N} : \operatorname{supp} f \subset [0, p + 1]\}$ is a vector space of dimension $(N - r)$.*

Proof. The set of functions of $S_{n,N}$ that have their support in $[0, L]$ is a vector space of dimension $(LN - n)_+$ (Corollary 2.1). To find at least one non-vanishing function in the vector space, its dimension must be greater than one meaning that $(LN - n) \geq 1 \Leftrightarrow L \geq$

$(n+1)/N = p + (r+1)/N$. Knowing that $L \in \mathbb{N}$ and $r < N$, we conclude that one must have that $L = (p+1)$ to find a nonzero compactly supported function. In this case, the dimension reads $((p+1)N - n) = (N + pN - n) = (N - r)$. □

With a single generator, the shortest-support basis is provided by the shortest function. In a consecutive multi-spline space, one would ideally take $(N-r)$ functions of size $(p+1)$ (the shortest) and complete with r functions of size $(p+2)$. This would result in N functions with a total support size of $(N-r)(p+1) + r(p+2) = Np + r + N = n + N = n_N + 1$, which is the objective for a shortest-support basis. For nonconsecutive multi-spline spaces, similar results should exist, but in a more complicated form.

2.5.2 Existence and Construction of mB-Splines

Definition 2.4. *A finite collection $\boldsymbol{\phi}$ of multi-spline functions is called an mB-spline of degree $\mathbf{n} = (n_1, \ldots, n_N)$ with $n_1 < \cdots < n_N \in \mathbb{N}$, if it is a shortest-support basis of the space $S_{\mathbf{n}}$.*

mB-splines constitute a natural extension of B-splines. Similar to the latter, mB-splines can be constructed recursively for any multi-spline space. Indeed, two basic transformations (the "increment step" and the "insertion step") allow one to convert a shortest-support basis of a given space into a shortest-support basis of a different space. To simplify the explanation, we say that the collection $\boldsymbol{\phi} = (\phi_1, ..., \phi_N) \in (L_2(\mathbb{R}))^N$ of compactly supported functions is standardized if, for $n = 1, \ldots, N$, we have that

(i) $\int_{\mathbb{R}} \phi_n(t)\mathrm{d}t \in \{0, 1\}$,

(ii) $\inf\{t \in \mathbb{R} \colon \phi_n(t) \neq 0\} \in [0, 1)$.

The second condition implies that the generating functions are causal, *i.e.* $\phi_n(t < 0) = 0$. Note that any $\boldsymbol{\phi}$ compactly supported can be standardized without altering $\mathrm{S}(\boldsymbol{\phi})$.

Increment Step

The B-splines β_+^{n+1} can be constructed recursively by noting that

$$\beta_+^{n+1}(x) = \Delta \left\{ \int_{-\infty}^{x} \beta_+^n(t)\mathrm{d}t \right\}, \tag{2.44}$$

where Δ is the finite difference operator $\Delta\{f\}(x) = (f(x) - f(x-1))$. The integration increases the polynomial degree, along with the smoothness at the knots (Step 1), while Δ ultimately returns a compactly supported function (Step 2). For multiple generating

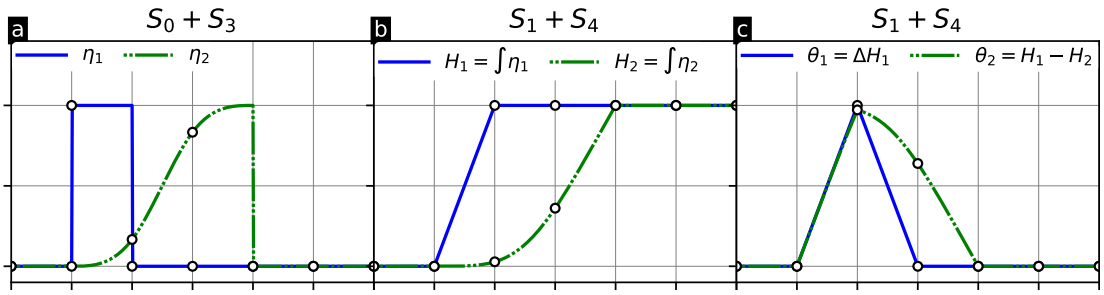

Figure 2.1: Increment step that yields a shortest-support basis of $S_1 + S_4$ starting from $S_0 + S_3$. (a) A shortest-support basis (η_1, η_2) for $S_0 + S_3$ ($|\text{supp}(\boldsymbol{\eta})| = 4$). (b) The integration of η_1 and η_2 results in two generators of $S_1 + S_4$, H_1 and H_2. (c) To get compactly supported functions with the same generating properties, we choose $\theta_1 = \Delta H_1$ and $\theta_2 = (H_1 - H_2)$. We found a shortest support-basis of $S_1 + S_4$ ($|\text{supp}(\boldsymbol{\theta})| = 5$).

functions, a similar two-step recursive approach is proposed. The general process is mathematically detailed below, while an intuitive example is proposed in Figure 2.1.

Suppose $\boldsymbol{\eta} = (\eta_1, ..., \eta_N) \in (L_2(\mathbb{R}))^N$ is an mB-spline of $S_{n_1} + \cdots + S_{n_N}$. The goal is to find an mB-spline of $S_{n_1+1} + \cdots + S_{n_N+1}$. It will be a generator with a support size of $(n_N + 2)$, able to reproduce the B-splines of degree $n_1 + 1, ..., n_N + 1$.

Integration

The collection of functions $\boldsymbol{\eta}$ is able to reproduce the B-splines of degree $n_1, \ldots, n_N$, that is, for any $s \in \{1, ..., N\}$ there exists a vector sequence $\boldsymbol{c}^s = (c_1^s, ..., c_N^s)$ (not necessarily in $(\ell_2(\mathbb{Z}))^N$) so that

$$\forall x \in \mathbb{R} : \beta_+^{n_s}(x) = \sum_{k \in \mathbb{Z}} \boldsymbol{c}^s[k]^T \boldsymbol{\eta}(x - k). \tag{2.45}$$

To justify the calculations to come, we assume that

$$c_1^s, ..., c_N^s \text{ are causal sequences, } \textit{i.e.}, \; c_n^s[k] = 0 \text{ for any } k < 0. \qquad (A_{\mathbf{n}})$$

The assumption $(A_{\mathbf{n}})$ is not overly restrictive because it will hold for the starting basis of our algorithm and then be preserved by the construction process. In the end, all the bases constructed will be able to reproduce the B-splines with causal sequences. Let $\boldsymbol{H} = (H_1, ..., H_N)$ be defined as

$$\boldsymbol{H}(x) = \int_{-\infty}^{x} \boldsymbol{\eta}(t) \mathrm{d}t. \tag{2.46}$$

The integration of equation (2.45), followed by the application of the operator Δ, yields

$$\begin{aligned}\beta_+^{n_s+1}(x) &= \Delta\left\{\sum_{k\in\mathbb{Z}} \boldsymbol{c}^s[k]^T \boldsymbol{H}(x-k)\right\} = \sum_{k\in\mathbb{Z}} \boldsymbol{c}^s[k]^T \Delta\{\boldsymbol{H}\}(x-k) \\ &= \sum_{k\in\mathbb{Z}} \boldsymbol{c}^s[k]^T (\boldsymbol{H}(x-k) - \boldsymbol{H}(x-1-k)) \qquad (2.47) \\ &= \sum_{k\in\mathbb{Z}} (\boldsymbol{c}^s[k]^T - \boldsymbol{c}^s[k-1]^T)\boldsymbol{H}(x-k). \qquad (2.48)\end{aligned}$$

The assumption that $c_1^s, ..., c_N^s$ are causal and the fact that $\boldsymbol{H}$ is also causal (because $\boldsymbol{\eta}$ is compactly supported and standardized) implies that, for any $x \in \mathbb{R}$, the sums in (2.48) have a finite number of nonzero terms. This enables us to switch the order of the operations (sum, integral, and Δ). Note that the sequence $(\boldsymbol{c}^s[k]^T - \boldsymbol{c}^s[k-1]^T)_{k\in\mathbb{Z}}$ is causal. In short, $\boldsymbol{H}$ can reproduce $(\beta_+^{n_1+1}, ..., \beta_+^{n_N+1})$ with causal sequences, but it is obviously not a shortest-support basis because its support is infinite.

Finite Difference

The aim now is to find a basis with the same reproducing properties as $\boldsymbol{H}$, but with minimal support. To that end, we denote by s_0 the index so that η_{s_0} is the shortest function in $\boldsymbol{\eta}$ that satisfies $\int_{\mathbb{R}} \eta_{s_0} \neq 0$. It must exist; if not, the generating $S(\boldsymbol{\eta})$ would only contains zero-mean functions and could not reproduce the B-splines that are not zero-mean. A shortest-support basis $\boldsymbol{\theta} = (\theta_1, ..., \theta_N)$ is then given by

$$\theta_s = \begin{cases} H_s & \text{if } s \neq s_0 \text{ and } \int_{\mathbb{R}} \eta_s(t)\mathrm{d}t = 0 \\ H_s - H_{s_0} & \text{if } s \neq s_0 \text{ and } \int_{\mathbb{R}} \eta_s(t)\mathrm{d}t \neq 0 \\ \Delta H_{s_0} & s = s_0 \end{cases} \qquad (2.49)$$

Because $\boldsymbol{\eta}$ is compactly supported and standardized, the choice of s_0 ensures that

$$|\mathrm{supp}(\theta_s)| = \begin{cases} |\mathrm{supp}(\eta_s)| & s \neq s_0 \\ |\mathrm{supp}(\eta_{s_0})| + 1 & s = s_0 \end{cases} \qquad (2.50)$$

In short, $|\mathrm{supp}(\boldsymbol{\theta})| = 1 + |\mathrm{supp}(\boldsymbol{\eta})| = n_N + 2$. Noting that $H_{s_0} = \sum_{k\in\mathbb{N}} \theta_{s_0}(\cdot - k)$, it is clear that $\boldsymbol{\theta}$ can reproduce $\boldsymbol{H}$ with causal coefficients. It also implies that $\boldsymbol{\theta}$ can reproduce $(\beta_+^{n_1+1}, ..., \beta_+^{n_N+1})$ with causal coefficients (see (2.48)), which justifies the assumption $(A_{\mathbf{n}})$. In conclusion, $\boldsymbol{\theta}$ is a shortest-support basis of $S_{n_1+1} + \cdots + S_{n_N+1}$.

Insertion Step

The present step enables us to add a generator to a shortest-support basis. Suppose $\boldsymbol{\eta} = (\eta_1, ..., \eta_N)$ is a standardized shortest-support basis of $S_{n_1} + \cdots + S_{n_N}$ and let

$\boldsymbol{\eta}' = (\delta, \eta_1, ..., \eta_N)$, where δ is the Dirac distribution. The increment step applied to $\boldsymbol{\eta}'$ yields a shortest-support basis for $S_0 + S_{n_1+1} + \cdots + S_{n_N+1}$. Indeed, the shortest function of $\boldsymbol{\eta}$ being δ, the new basis $\boldsymbol{\theta}' = (\theta'_0, ..., \theta'_N)$ is given by

$$\theta'_n : x \mapsto \begin{cases} \Delta\{\int_{-\infty}^{x} \delta(t)\mathrm{d}t\} = \beta_+^0(x), & n = 0 \\ \int_{-\infty}^{x} \eta_n(t)\mathrm{d}t, & n > 0 \text{ and } \int_{\mathbb{R}} \eta_n(t)\mathrm{d}t = 0 \\ \int_{-\infty}^{x} (\eta_n(t) - \delta(t))\mathrm{d}t, & n > 0 \text{ and } \int_{\mathbb{R}} \eta_n(t)\mathrm{d}t \neq 0. \end{cases} \tag{2.51}$$

Because $\boldsymbol{\eta}$ is compactly supported and standardized, we have that

$$|\mathrm{supp}(\theta'_n)| = \begin{cases} 1, & n = 0 \\ |\mathrm{supp}(\eta_n)|, & \text{otherwise,} \end{cases}$$

which means that $|\mathrm{supp}(\boldsymbol{\theta}')| = |\mathrm{supp}(\eta')| + 1 = n_N + 2$. The process also ensures that $\boldsymbol{\theta}'$ is a shortest-support basis of $S_0 + S_{n_1+1} + \cdots + S_{n_N+1}$.

Theorem 2.3. *Let $n_1 < \cdots < n_N \in \mathbb{N}\backslash\{0\}$. There exists an mB-spline $\boldsymbol{\eta} = (\eta_1, ..., \eta_N) \in (L_2(\mathbb{R}))^N$ of $S_{n_1} + \cdots + S_{n_N}$ that can be constructed recursively with increment and insertion steps.*

Proof. The increment and insertion steps are sufficient to construct an mB-spline for any multi-spline space. Indeed, take $\boldsymbol{\eta_0} = (\beta_+^{n_N - n_{N-1} - 1})$ a shortest support basis for $S_{n_N - n_{N-1} - 1}$. The insertion step gives a shortest-support basis for $S_0 + S_{n_N - n_{N-1}}$. After $(n_{N-1} - n_{N-2} - 1)$ increment steps and one insertion step, the process gives a shortest-support basis for $S_0 + S_{n_{N-1} - n_{N-2}} + S_{n_N - n_{N-2}}$. By iteration, a shortest-support basis for $S_0 + S_{n_2 - n_1} + \cdots + S_{n_N - n_1}$ is obtained. Applying n_1 increment steps, we finally obtain a shortest-support basis for $S_{n_1} + \cdots + S_{n_N}$ □

Examples of mB-splines will be provided in Section 2.6. Note that our algorithm does not always output functions with the most practical form. This is corrected by appropriate linear combinations and, possibly, translations that do not alter the reproducing properties and the support size. For instance, for the space $S_2 + S_3$, our construction will need a simple linear combination to obtain the wellknown bicubic Hermite splines. We conclude this section with a result on the minimal number of generating functions required to generate multi-spline spaces.

Theorem 2.4. *Let $n_1 < \cdots < n_N \in \mathbb{N} \setminus \{0\}$. The space $S_{\mathbf{n}} = S_{n_1} + \cdots + S_{n_N}$ cannot be generated by fewer than N compactly supported generating functions.*

Proof. From Theorem 2.3, there exists an mB-spline of $S_{\mathbf{n}}$ composed of N functions, say, $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_N) \in (S_{\mathbf{n}})^N$. Let $\boldsymbol{\psi} = (\psi_1, ..., \psi_M) \in (S_{\mathbf{n}})^M$ be a collection of compactly supported functions able to generate $S_{\mathbf{n}}$. It means that $\boldsymbol{\eta}$ and $\boldsymbol{\psi}$ can reproduce each other and, by Lemma 2.4, $M \geq N$. □

Note that N is a lower bound and the number of generating function of a shortest-support basis can exceed N. For instance, take $\boldsymbol{\eta} = (\eta_1, \eta_2)$ with

$$\eta_1 : x \mapsto \beta_0(2x) = \mathbb{1}_{[0,1/2)}(x) \tag{2.52}$$

$$\eta_2 : x \mapsto \beta_0(2(x-1/2)) = \mathbb{1}_{[1/2,1)(x)}. \tag{2.53}$$

Since $\eta_1 + \eta_2 = \beta_0$, $\boldsymbol{\eta}$ can reproduce S_0. In addition, the fact that $|\text{supp}(\boldsymbol{\eta})| = 1$ means that it is a shortest-support basis of degree 0 and now it is composed of two generating functions. (Note that the space they generate is larger than S_0).

2.6 Applications

2.6.1 Generalized Sampling in Multi-Spline Spaces

We consider a multi-spline space $S_{\mathbf{n}}$ along with the N-component mB-spline $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_N)$ and some corresponding analysis functions $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_N)$. As we now show, the generalized-sampling formulation presented in [115] can be extended to multiple generators. Let $\mathcal{H}$ be a space considerably larger than $\mathrm{S}(\boldsymbol{\phi})$. Consider $f \in \mathcal{H}$, from which we know only some discrete measurements $(\boldsymbol{g}[n])_{n\in\mathbb{Z}}$ written

$$\boldsymbol{g}[n] = \langle \boldsymbol{\psi}(\cdot - n), f \rangle = (\langle \psi_1(\cdot - n), f \rangle, ..., \langle \psi_N(\cdot - n), f \rangle).$$

To construct an approximation $\tilde{f} \in \mathrm{S}(\boldsymbol{\phi})$ of f, a standard way is to enforce consistency [111, 116], in the sense that f and $\tilde{f}$ must give the same measurements. This formulation generalizes the notion of interpolation. For instance, to interpolate the value of f and its derivative at the sampling locations, take $\psi_1 = \delta$ and $\psi_2 = \delta'$. In such a case, consistency simply means that f and $\tilde{f}$ should have the same value and the same derivative at the grid points. In general, the consistency requirement translates into

$$\begin{aligned}
\langle \boldsymbol{\psi}(\cdot - n), f \rangle &= \langle \boldsymbol{\psi}(\cdot - n), \tilde{f} \rangle \\
&= \sum_{k\in\mathbb{Z}} \langle \boldsymbol{\psi}(\cdot - n), \boldsymbol{\phi}^T(\cdot - k) \rangle \cdot \boldsymbol{c}[k] \\
&= \sum_{k\in\mathbb{Z}} \langle \boldsymbol{\psi}(\cdot - (n-k)), \boldsymbol{\phi}^T \rangle \cdot \boldsymbol{c}[k] \\
&= (\boldsymbol{A}_{\boldsymbol{\Phi}\boldsymbol{\Psi}} * \boldsymbol{c})[n]
\end{aligned} \tag{2.54}$$

where $(\boldsymbol{c}[n])_{n\in\mathbb{Z}}$ is the unique vector sequence representing $\tilde{f} = \sum_{k\in\mathbb{Z}} \boldsymbol{c}[k]^T \boldsymbol{\phi}(\cdot - k)$ and $\boldsymbol{A}_{\boldsymbol{\Phi}\boldsymbol{\Psi}}[n] = \langle \boldsymbol{\psi}(\cdot - n), \boldsymbol{\phi}^T(\cdot) \rangle$ is the matrix-valued sequence of the measurements of the basis functions. To solve our problem, we rely on the theory of signal and systems, including the z-transform. Indeed, with this framework efficient implementation techniques naturally stand out. When the matrix-valued filter $\boldsymbol{A}_{\boldsymbol{\Phi}\boldsymbol{\Psi}}$ is invertible (see [115, Proposition 1] for the invertibility condition), the vector $\boldsymbol{c}$ of sequences can be computed

from the measurements by applying the matrix-valued inverse filter $\boldsymbol{Q}$, like in

$$\boldsymbol{c}[n] = (\boldsymbol{Q} * \boldsymbol{g})[n]. \tag{2.55}$$

Its transfer function verifies in the z-domain $\hat{\boldsymbol{Q}}(z) = \hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}^{-1}(z)$. This matrix filter has not necessarily a finite impulse response (FIR) but it can be decomposed as $\hat{\boldsymbol{Q}}(z) = \frac{1}{\det \hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}(z)} \mathrm{com}(\hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}(z))^T$, where $\mathrm{com}(\hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}})$ denotes the cofactor matrix of $\hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}$. For compactly supported analysis functions, the comatrix $\mathrm{com}(\hat{\boldsymbol{A}}(z))$ is FIR because it is a Laurent polynomial in z, so it is straightforward to implement. On the contrary, $\frac{1}{\det \hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}(z)}$ is often not FIR. Nonetheless, it can usually be implemented efficiently too, using the same techniques as in [133].

Online Interactive Tutorial Some examples are implemented in an online interactive demo [†], a screenshot being provided in Figure 2.3. The user can control the discrete measurements of a function (value, derivative), choose a multi-spline reconstruction space, and see in live the reconstructed function.

2.6.2 Derivative Sampling with High-Degree Multi-Splines in $S_{2p} + S_{2p+1}$

The derivative sampling problem reads for $f \in \mathcal{H}$

$$\text{find } \tilde{f} \in S_{\mathbf{n}} : \begin{cases} \tilde{f}(k) = f(k) \\ \tilde{f}'(k) = f'(k) \end{cases}, k \in \mathbb{Z}. \tag{2.56}$$

The most relevant reconstruction spaces have the form $S_{\mathbf{n}} = S_{2p} + S_{2p+1}$. The underlying reason is that the filter complexity is the same for the spaces $S_{2p} + S_{2p+1}$ and $S_{2p-1} + S_{2p}$, so, the higher degree is preferred (the filter has $2(p-1)$ roots). Note that the same occurs when one performs classical interpolation with B-splines and odd degrees are usually preferred. To the best of our knowledge, when $p > 1$, no solution based on shortest-support bases and recursive filtering has been proposed so far. Our construction of shortest-bases results in the functions η_1 and η_2. They have a support size $(p+1)$ and are plotted in Figure 2.2. Due to the symmetry properties of those functions, the entries of $\hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}(z)$ have poles that come in reciprocal pairs. Consequently, the inverse matrix filter can be implemented with efficient recursive techniques, as detailed in [132, 133].
The case of quintic-degree derivative sampling is detailed now. The basis functions are specified in Table 2.1. The z-transform of the filter $\hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}(z)$ reads

$$\hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}(z) = \begin{bmatrix} \frac{z^{-1}+z^{-2}}{2} & \frac{z^{-1}-z^{-2}}{2} \\ \frac{5(z^{-1}-z^{-2})}{4} & \frac{5(z^{-1}+z^{-2})}{8} \end{bmatrix}. \tag{2.57}$$

[†] https://bigsplinesepfl.github.io/

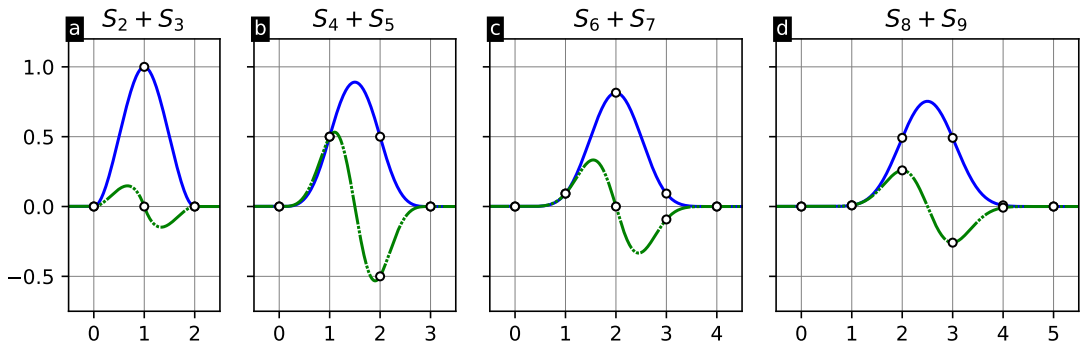

Figure 2.2: Shortest-support bases for derivative sampling, obtained with the shortest-basis algorithm and some linear combinations to get a symmetric and an antisymmetric function. (a) The well-known bicubic Hermite splines. (b)-(c)-(d) New bases for derivative sampling with high-degree splines. These functions are piecewise polynomials of degree 5, 7, 9 with continuity of the derivatives of order 3, 5, 7, respectively.

It follows that the transpose comatrix satisfies

$$\mathrm{com}(\hat{\mathrm{A}}(\mathrm{z}))^{\mathrm{T}} \quad \overset{\mathrm{z}}{\longleftrightarrow} \quad \frac{1}{2}\begin{bmatrix} \frac{5(\delta[\cdot-1]+\delta[\cdot-2])}{4} & -\delta[\cdot-1]+\delta[\cdot-2] \\ -\frac{5(\delta[\cdot-1]-\delta[\cdot-2])}{2} & \delta[\cdot-1]+\delta[\cdot-2] \end{bmatrix} \tag{2.58}$$

and the determinant

$$\frac{z^{-1}}{\det \hat{\boldsymbol{A}}(z)} = \frac{16}{5}\frac{-z}{(1-z_0 z^{-1})(1-z_0^{-1}z^{-1})} \quad \overset{z}{\longleftrightarrow} \quad d[n], \tag{2.59}$$

where $z_0 = (3-2\sqrt{2})$. This means that the convolution of any sequence with d can be implemented recursively. Interestingly, it is the same inverse filter as in cubic-spline interpolation. The reader can therefore refer to [126] for a detailed explanation of the implementation. The expansion coefficients can be evaluated as

$$\begin{aligned} c_1 &= d * \left(\frac{5}{8}\Delta^+\{f\} - \frac{1}{2}\Delta\{f'\}\right) \\ c_2 &= d * \left(-\frac{5}{4}\Delta\{f\} + \frac{1}{2}\Delta^+\{f'\}\right), \end{aligned} \tag{2.60}$$

where $\Delta^+\{f\}[k] = f[k] + f[k-1]$. Finally, the multi-spline that is consistent with the measurements is given by

$$\tilde{f}(x) = \sum_{k\in\mathbb{Z}} c_1[k]\eta_1(x-k) + \sum_{k\in\mathbb{Z}} c_2[k]\eta_2(x-k). \tag{2.61}$$

		slice #	x_k^0	x_k^1	x_k^2	x_k^3	x_k^4	x_k^5	x_k^6	x_k^7
S_2+S_3	η_1	$k=0$			-3	2				
		$k=1$	1		-3	1				
	η_2	$k=0$			-1	1				
		$k=1$		1	-2	1				
S_4+S_5	$4\eta_1$	$k=0$					5	-3		
		$k=1$	2	5		-10	5			
		$k=2$	2	-5		10	-10	3		
	$8\eta_2$	$k=0$					15	-11		
		$k=1$	4	5	-20	-50	95	-38		
		$k=2$	-4	5	20	-50	40	-11		
S_6+S_7	$108\eta_1$	$k=0$							21	-11
		$k=1$	10	49	84	35	-70	-105	112	-27
		$k=2$	88		-168		140		-77	27
		$k=3$	10	-49	84	-35	-70	105	-56	11
	$\frac{918}{5}\eta_2$	$k=0$							42	-25
		$k=1$	17	77	105	-35	-245	-273	539	-185
		$k=2$		-224		560		-924	756	-185
		$k=3$	-17	77	-105	-35	245	-273	133	-25

Table 2.1: Slices of shortest-support bases for derivative sampling. The slices are given as linear combinations of the shifted monomials $x_k^n = (x-k)^n$ if $x \in [k, k+1)$ and $x_k^n = 0$ otherwise.

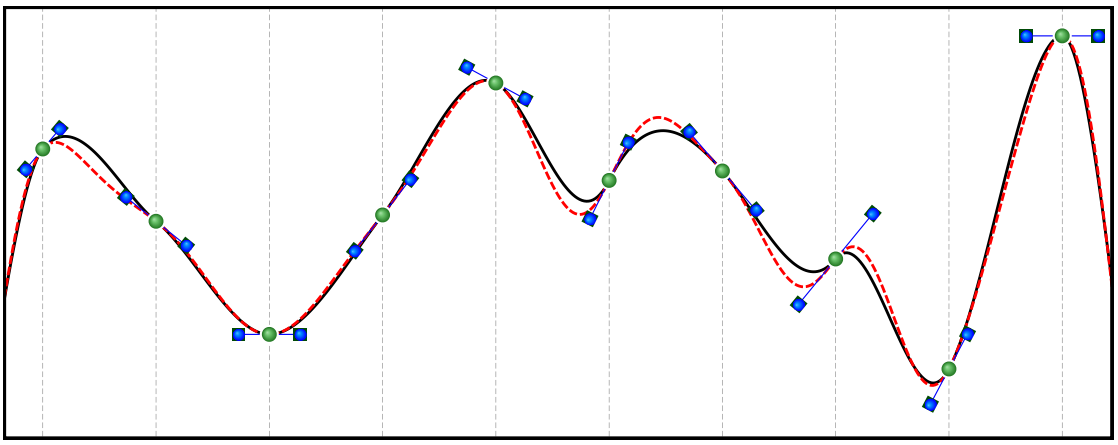

Figure 2.3: Derivative sampling with optimal bases. The solid curve lies in $S_2 + S_3$ (cubic piecewise polynomials with continuous derivative) and the dashed curve lies in $S_4 + S_5$ (quintic piecewise polynomials with continuous third derivative).

Derivative Sampling in $S_2 + S_3 + S_4$

Here, we consider the setting $\boldsymbol{\psi} = (\delta, \delta', \delta(\cdot - 1/2))$, which means that the value of the function to be reconstructed is sampled twice more often than its derivative. The specification of $S_2+S_3+S_4$ as reconstruction space provides then an explicit interpolation

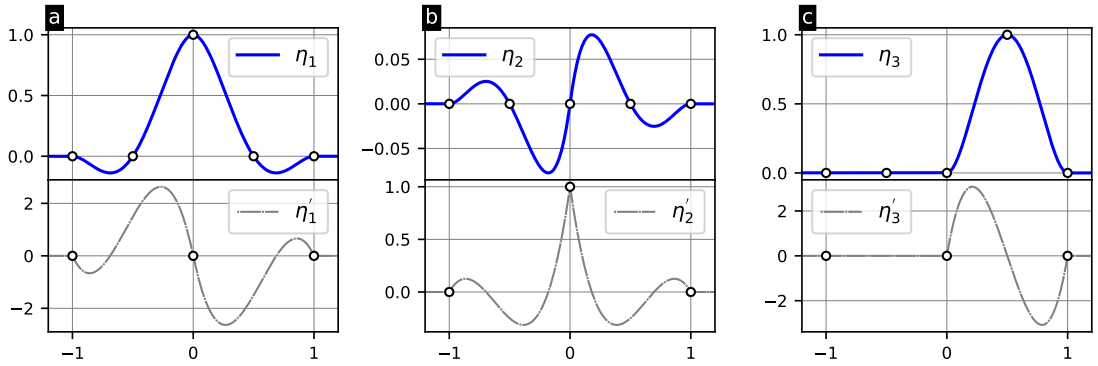

Figure 2.4: Shortest basis of $S_2+S_3+S_4$ associated to the analysis functions $\boldsymbol{\psi} = (\delta, \delta', \delta(\cdot - 1/2))$.

formula, which involves the shortest-support basis $\boldsymbol{\eta}$, plotted in Figure 2.4. This formula reads

$$\tilde{f}(x) = \sum_{k\in\mathbb{Z}} \big(f(k)\eta_1(x-k) + f'(k)\eta_2(x-k) + f(k+1/2)\eta_3(x-k)\big)\,. \tag{2.62}$$

More generally, we observed that the addition of N consecutive spline spaces to $S_2 + S_3$ (*i.e.*, choosing $S_2 + S_3 + \cdots + S_{3+N}$) allows one to perform derivative sampling and interpolate the function N times between the integers with a direct interpolation formula.

Direct Derivative Sampling in $S_2 + \cdots + S_{2p+1}$

The space $S_2 + S_3 + S_4 + S_5$ is also well suited for derivative sampling with $\psi = (\delta, \delta', \delta(\cdot - 1/2), \delta'(\cdot - 1/2))$ because of the structure of its shortest-support generating functions η_1, η_2, η_3, and η_4 (Figure 2.5). Indeed, it yields the direct interpolation formula

$$\tilde{f}(x) = \sum_{k\in\mathbb{Z}} f(k+1/2)\eta_1(x-k) + f(k)\eta_2(x-k+1) \tag{2.63}$$

$$+f'(k+1/2)\eta_3(x-k) + f'(k)\eta_4(x-k+1). \tag{2.64}$$

The sampling step is $1/2$, but the spline knots are still located at the integers. Note that the sampling step can be tuned at will by dilation of the generating functions. More generally, we conjecture that there exist basis functions with the interpolatory property for any space of the form $S_2 + \cdots + S_{2p+1}$ and the sampling step $1/p$. This conjecture was verified for $p = 1$ (bicubic Hermite splines), $p = 2$ (Figure 2.5) and $p \in \{3, 4\}$.

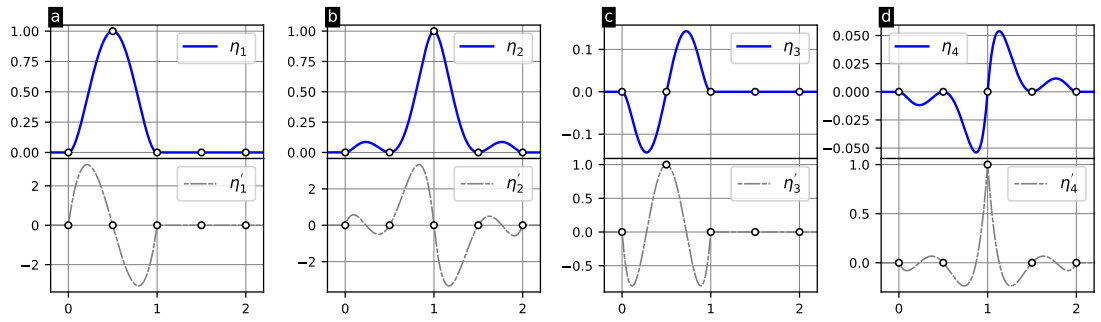

Figure 2.5: Shortest basis of $S_2 + S_3 + S_4 + S_5$ for direct derivative sampling.

2.6.3 Classical Interpolation

The classical interpolation problem reads for $f \in \mathcal{H}$

$$\text{find } \tilde{f} \in S_{\mathbf{n}} : f(k) = \tilde{f}(k), k \in \mathbb{Z}. \tag{2.65}$$

When the number N of generating functions is greater than 1, we have two equivalent options:

(i) to sample the function f with the sampling step $1/N$;

(ii) to dilate the generators by a factor of N, keeping a unit sampling step.

We present the result in accordance with Option (i).

Modified Lagrange Polynomials in $S_1 + \cdots + S_N$

Classical interpolation is well solved by B-splines but, starting from degree 2, the filter is neither FIR nor causal. Exact operations such as local interpolation or interpolation with a finite delay are therefore not possible. Some workarounds exist [151]; we present now one that is based on modified Lagrange polynomials. Let $\boldsymbol{l} = (l_1, \ldots, l_N)$ be a collection of N generating function such that, for $x \in [0, 1]$, $l_q(x) = \prod_{\substack{p=0 \\ p \neq q}}^{N} \frac{Nx-p}{q-p}$. In this way, when $q = 1, \ldots, (N-1)$, l_q is zero at $x = 0$ and $x = 1$ so it can be set to zero for $x \notin [0, 1]$ and $l_q \in S_1$. Noting that $l_N(1) = 1$, to make sure that $l_N \in S_1$, we extend its support to $[1, 2]$ and set, $\forall x \in [1, 2]$, $l_N(x) = l_N(2 - x)$ (see Figure 2.6). These functions constitute a shortest-support basis of $S_1 + \cdots + S_N$ and give a direct interpolation formula. Interestingly, those basis functions are sometimes used for finite-element methods [152].

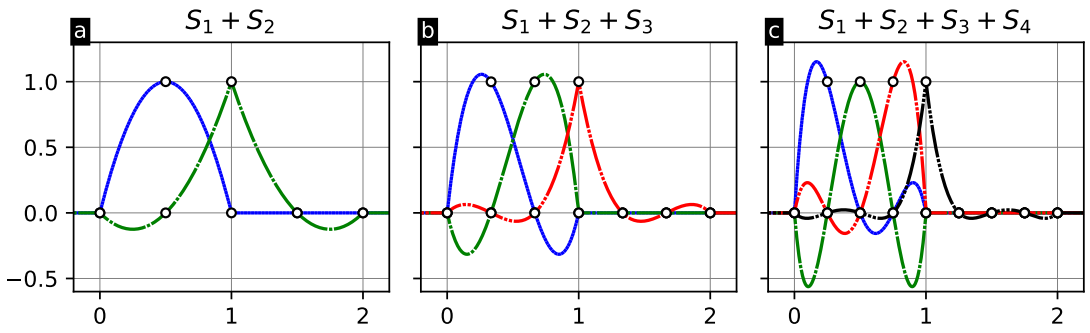

Figure 2.6: Shortest-support basis for $S_1 + \cdots + S_N$. The basis functions are continuous and able to reproduce any polynomial of degree up to N.

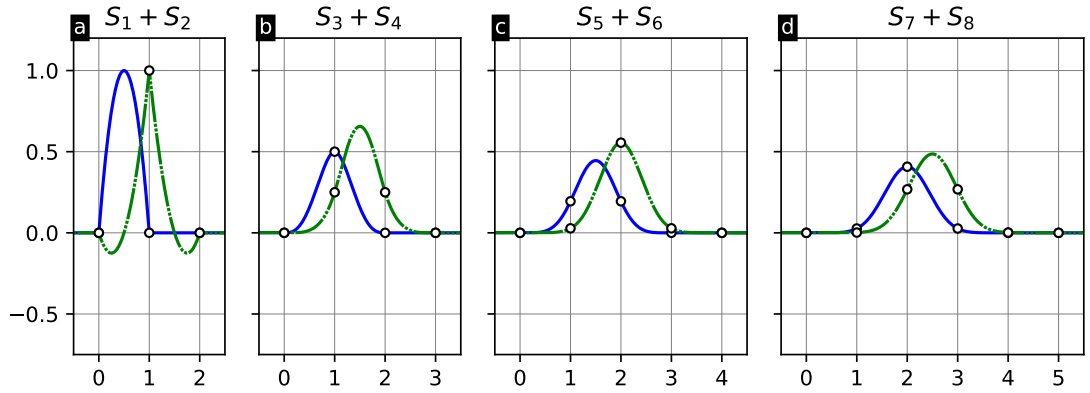

Figure 2.7: Shortest bi-spline bases for classical interpolation with a half-integer sampling step. (a) In $S_1 + S_2$, the functions presented give a direct interpolation formula. (b) (c) (d) The functions are piecewise polynomials of degree 4, 6, 8 with continuity of the derivatives of order 2, 4, 6 respectively. To perform interpolation, a filter with 2, 4, 6 roots respectively has to be inverted.

Bi-Spline Classical Interpolation in $S_{2p+1} + S_{2p+2}$

A bi-spline is the sum of two splines of different degrees, and it can be used to perform classical interpolation. In particular, interpolation in the reconstruction space $S_{\mathbf{n}} = S_{2p+1} + S_{2p+2}$ leads to a filter with p pairs of reciprocal roots. In terms of filtering, it has therefore the same complexity as for the interpolation inverse filter associated with the single space S_{2p+1}. Shortest-support basis functions for such spaces are plotted in Figure 2.7. We now detail how this interpolation is performed for $S_3 + S_4$, keeping in mind that the other cases are similar. The z-transform of the filter $\hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}(z)$ reads

$$\hat{\boldsymbol{A}}_{\boldsymbol{\Phi\Psi}}(z) = \begin{bmatrix} \frac{z^{-1}}{2} & \frac{z^{-1}+z^{-2}}{4} \\ \frac{5(z^{-1}+z^{-2})}{32} & \frac{5(z^{-1}+z^{-3})+210z^{-2}}{320} \end{bmatrix}, \tag{2.66}$$

while the z-transform of the inverse filter can be decomposed as

$$\hat{\boldsymbol{Q}}(z) = \hat{p}(z)\hat{\boldsymbol{P}}(z), \tag{2.67}$$

where

$$\hat{\boldsymbol{P}}(z) = \begin{bmatrix} \frac{5(1+z^{-2})+210z^{-1}}{320} & -\frac{1+z^{-1}}{4} \\ -\frac{5(1+z^{-1})}{32} & \frac{1}{2} \end{bmatrix} \tag{2.68}$$

and

$$\hat{p}(z) = \frac{32}{(1-z_0 z^{-1})(1-z_0^{-1}z^{-1})} \tag{2.69}$$

with $z_0 = (4-\sqrt{15})$. The final steps are identical to the detailed case of derivative sampling (recursive filtering).

2.6.4 Bézier Curves and Computer Graphics in $S_1 + S_2 + S_3$ and $S_1 + S_2$

In this section, we use our multi-spline formulation to revisit some Bézier curves and, in particular, the cubic Bézier curves that are popular in computer graphics. Each portion of the curve is a cubic polynomial defined by four control points.

- Starting point and ending point of the portion.
- Two handles that control the tangent of the curve at each extremity of the portion.

Thus, the value of the function and its left and right derivatives are controlled on the knots. From a multi-spline perspective, any cubic Bézier curve lies in the space $S_1+S_2+S_3$. With the well chosen generating functions η_1, η_2, and η_3 plotted in Figure 2.8, the interpolation formula is explicit and reads

$$\tilde{f}(x) = \sum_{k\in\mathbb{Z}} f(k)\eta_1(x-k) + \sum_{k\in\mathbb{Z}} f'(k^-)\eta_2(x-k) + \sum_{k\in\mathbb{Z}} f'(k^+)\eta_3(x-k), \tag{2.70}$$

where $f'(k^-)$ and $f'(k^+)$ denote the left and right derivatives at k, respectively. Interestingly, η_2 and η_3 can be obtained from the bi-cubic Hermite splines, by splitting the antisymmetric function into two functions (see Figure 2.2 (a)). It gives a simple interpretation to cubic Bézier curves as illustrated in Figure 2.9. Similarly, quadratic Bézier curves are also multi-splines, this time associated to the space $S_1 + S_2$ (Figure 2.8).

2.6.5 Nonconsecutive Bi-spline Spaces

Nonconsecutive multi-spline spaces are relevant to represent signals that have components of different regularity [153]. For instance, the space $S_0 + S_p$, with $p > 0$, consists of

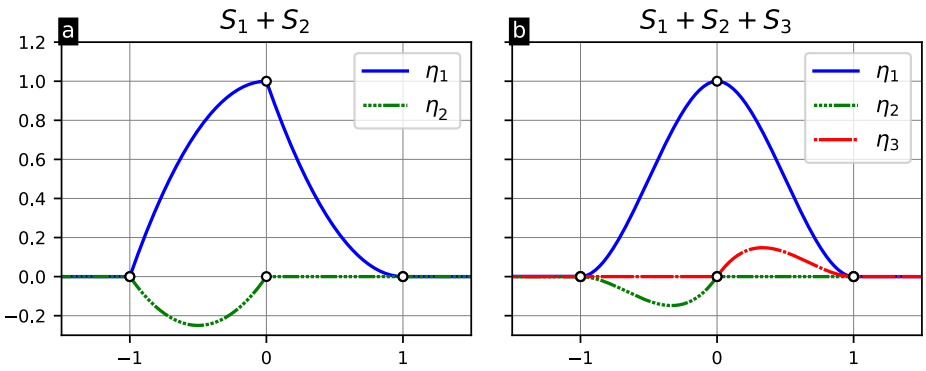

Figure 2.8: Shortest-support bases for application in classical computer-graphics. (a) Shortest basis for $S_1 + S_2$. The function η_1 controls the value of the function on the knots while η_2 controls the left derivative on the knots. These functions reproduce any quadratic Bézier curve. (b) Shortest basis for $S_1 + S_2 + S_3$. The function η_1 controls the value of the function on the knots while η_2 and η_3 control the left and right derivatives, respectively, on the knots. These functions can reproduce any cubic Bézier curve with the shortest support. They also give a simple interpretation of such curves.

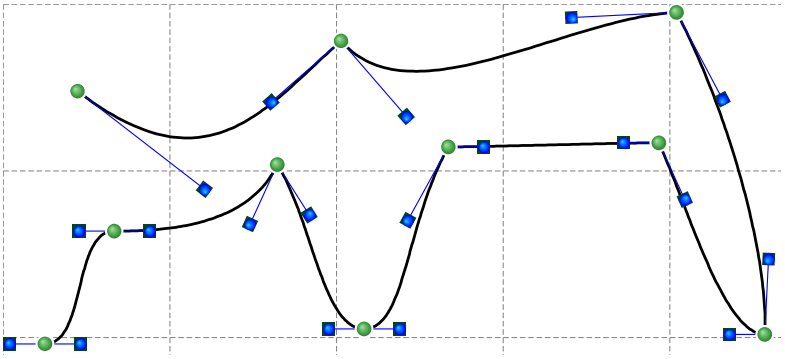

Figure 2.9: Screenshot from the online demo. The shortest basis of the space $S_1 + S_2 + S_3$ allows one to control the value of the function (green dots) and the left/right derivatives (handles). It yields the same curve as with standard vector-graphics editors relying on cubic Bézier curves. In this figure, the parametric curves are two-dimensional and the interpolation is performed component-wise.

smooth signals with sharp jumps. In Figure 2.10, we show shortest-support bases of $S_0 + S_p$, for $p \in \{2, 3, 4\}$, that were obtained with our construction algorithm.

2.7 Conclusion

In this chapter, we have introduced the notion of shortest-support bases of degree M. They are the shortest-support collections of functions that generate a reconstruction space with an approximation power of order $(M + 1)$. We proved that shortest-support bases necessarily generate Riesz bases, a minimal requirement for practical applications.

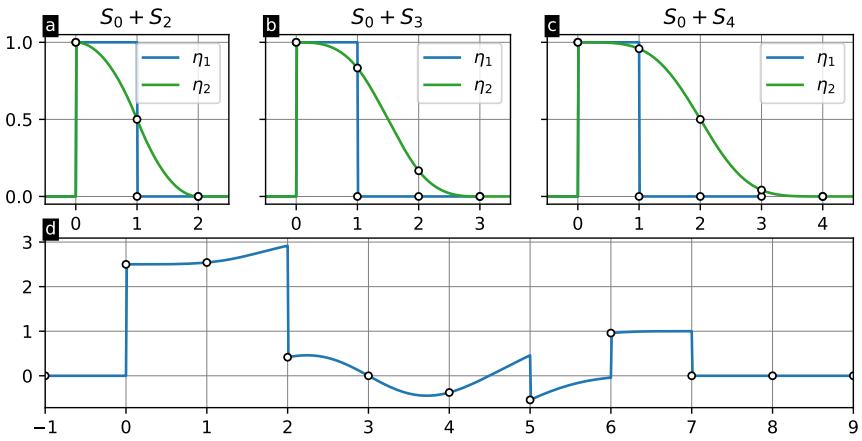

Figure 2.10: (a) (b) (c) Shortest-support bases for the spaces $S_0 + S_2$, $S_0 + S_3$ and $S_0 + S_4$. (d) An example of a hybrid bi-spline that lies in the space $S_0 + S_4$.

With a single generator, the unique shortest-support basis of degree M is the well-known B-spline of degree M. We extended this notion to multiple generators and proposed a recursive method that yields shortest bases for any multi-spline space. These new sets of functions helped us transpose the efficient reconstruction techniques developed for B-splines, and perform generalized sampling. In particular, we have provided a method to perform fast derivative sampling with any approximation power. Finally, we presented a new way to approach some Bézier curves.

3 Stable Parameterization of Continuous and Piecewise-Linear Functions

The text of this chapter is adapted from the published article

A. Goujon, J. Campos and M. Unser "Stable parameterization of continuous and piecewise-linear functions", *Applied and Computational Harmonic Analysis*, volume 67, paper 101581, November 2023.

Summary

Rectified-linear-unit (ReLU) neural networks, which play a prominent role in deep learning, generate continuous and piecewise-linear (CPWL) functions. While they provide a powerful parametric representation, the mapping between the parameter and function spaces lacks stability. In this chapter, we investigate an alternative representation of CPWL functions that relies on local hat basis functions and that is applicable to low-dimensional regression problems. It is predicated on the fact that any CPWL function can be specified by a triangulation and its values at the grid points. We give the necessary and sufficient condition on the triangulation (in any number of dimensions and with any number of vertices) for the hat functions to form a Riesz basis, which ensures that the link between the parameters and the corresponding CPWL function is stable and unique. In addition, we provide an estimate of the $\ell_2 \to L_2$ condition number of this local representation. As a special case of our framework, we focus on a systematic parameterization of $\mathbb{R}^d$ with control points placed on a uniform grid. In particular, we choose hat basis functions that are shifted replicas of a single linear box spline. In this setting, we prove that our general estimate of the condition number is exact. We also relate the local representation to a nonlocal one based on shifts of a causal ReLU-like function. Finally, we indicate how to efficiently estimate the Lipschitz constant of the CPWL mapping.

3.1 Introduction

3.1.1 Continuous and Piecewise-Linear Functions for Supervised Learning

The purpose of supervised learning is to reconstruct an unknown mapping from a set of samples [154]. Namely, given a collection of training data pairs $(\mathbf{v}_k, y_k) \in \mathbb{R}^d \times \mathbb{R}$ for $k = 1, \dots, K$, one wants to find $f\colon \mathbb{R}^d \to \mathbb{R}$ such that $f(\mathbf{v}_k) \approx y_k$ for $k = 1, \dots, K$, without overfitting. As such, the problem is ill-posed. To make it numerically tractable, a reconstruction space $\mathcal{H}$ is chosen as the image of a finite-dimensional parameter space Θ under a given synthesis operator $T\colon \Theta \to \mathcal{H}$. This operator maps a parameter $\boldsymbol{\theta} \in \Theta$ to its continuous representation $T\{\boldsymbol{\theta}\} \in \mathcal{H}$. A celebrated way to choose the synthesis operator is to pick a feedforward neural-network architecture. Given the multidimensional parameter $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{L+1}) \in \Theta$, we then have that

$$T\{\boldsymbol{\theta}\} = (\boldsymbol{f}_{\boldsymbol{\theta}_{L+1}} \circ \boldsymbol{\sigma}_L \circ \boldsymbol{f}_{\boldsymbol{\theta}_L} \circ \boldsymbol{\sigma}_{L-1} \circ \cdots \circ \boldsymbol{\sigma}_2 \circ \boldsymbol{f}_{\boldsymbol{\theta}_2} \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{f}_{\boldsymbol{\theta}_1}), \tag{3.1}$$

where L is the number of hidden layers of the neural network, $\boldsymbol{f}_{\boldsymbol{\theta}_k}\colon \mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}}$ is an affine function parameterized by $\boldsymbol{\theta}_k$, and $\boldsymbol{\sigma}_k$ is an activation function that is chosen *a priori*. For the model to be able to generate a function space with good approximation properties, the activation functions need to be nonaffine—otherwise, the generated function would remain trivially affine. The pointwise rectified-linear unit (ReLU) $x \mapsto \max(x, 0)$ is one of the most popular activation functions and it usually provides state-of-the-art performance [155, 156]. In this case, $T\{\boldsymbol{\theta}\}$ is the composition of continuous piecewise-linear (CPWL) functions, which turns out to be a CPWL function as well [70]. The reverse also holds true: any CPWL function $\mathbb{R}^d \to \mathbb{R}$ can be parameterized by a deep neural network with at most $\lceil \log_2(d+1) \rceil$ hidden layers [157]. The depth of the architecture is instrumental to improve the approximation power of the network [70, 158, 159] and its generalization ability [160], but it is an obstacle to the control of the model. For instance, to control the Lipschitz constant of a feedforward neural network, state-of-the-art techniques rely on theoretical upper bounds whose tightness degrades each time a new layer is added [48, 161]. The depth also makes it hard to determine the role of each parameter in the constructed mapping.

3.1.2 Linear Expansion of Continuous and Piecewise-Linear Functions

More interpretable representations of CPWL functions are provided by linear expansions. They boil down to two families: local and nonlocal representations (Figure 3.1).

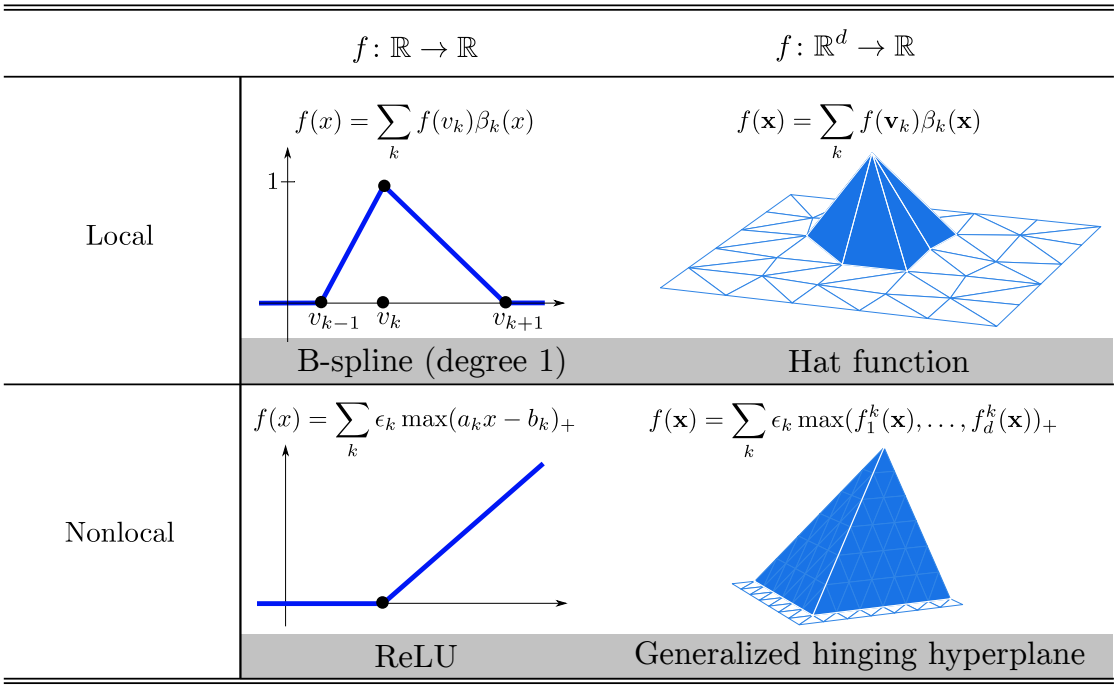

Figure 3.1: Local and nonlocal building bricks of CPWL functions, from dimension 1 to any dimension. Notice that the nonlocal basis functions have several equivalent variations; only the ReLU-like one is shown in this figure.

Local Representation

In dimension $d = 1$, any CPWL function f can be represented by the linear expansion $f = \sum_k f(v_k)\beta_k$, where the β_k are the underlying triangular B-spline functions [129](Figure 3.1, upper left) and the $\{v_k\}$ are the knots of f, where the slope of f changes. In dimension $d > 1$, the knots are replaced by a triangulation of the set $\{\mathbf{v}_k\}$ of vertices, which partitions the input domain into simplices. These simplices are the convex hull of a subset of $(d + 1)$ vertices and will serve to define the linear regions of f. Given an appropriate triangulation, any CPWL function can be represented by the linear expansion $f = \sum_k f(\mathbf{v}_k)\beta_k$, where the β_k now denote the hat functions—a.k.a. nodal basis functions or tent-shaped linear basis functions—that correspond to the triangulation [162] (Figure 3.1, upper right). Each basis function β_k is defined as the unique CPWL function that satisfies $\beta_k(\mathbf{v}_q) = 1$ for $k = q$ and $\beta_k(\mathbf{v}_q) = 0$ otherwise, and that is affine over each simplex of the triangulation. The β_k are locally supported, hence the attribute *local*. When the vertices are regularly spaced so that they coincide with the sites of a lattice, the hat functions can be chosen as translates of a unique function, namely, a linear box spline [163, 164].

There exist recent works that leverage the explicit representation offered by the local parameterization to learn a CPWL function f for regression tasks. Since the partition of f is known and typically not learned, it is possible to regularize the function during

training with an explicit regularizer. For instance, one can use the total roughness of f [165–167] or the Hessian total variation of f [168, 169], which is not possible with neural networks (NNs). With NNs, it is more common to regularize the parameters, which yields an implicit regularization on the learned mapping that is hard to understand. The global knowledge of the function offered by the local parameterization has, however, a cost. Since the linear regions are fixed, it suffers from the curse of dimensionality. To partially overcome this limitation, it was proposed in [166] to use the local parameterization on subspaces of the features along with ensemble techniques.

Nonlocal Representations

In dimension $d = 1$, a CPWL function f with control points v_k can also be represented by the nonlocal representation [170]

$$f(x) = a_0 + a_1(x - v_1) + \sum_{k=2}^{K-1} a_k \mathrm{ReLU}(x - v_k). \tag{3.2}$$

Note that (3.2) is one instance out of many nonlocal representations, another one being $f(x) = c_0 + c_1(x - v_1) + \sum_{k=2}^{K-1} c_k \, |x - v_k|$. The generalization to any dimension is due to Wang and Sun [171]. Their generalized hinging-hyperplanes (GHH) model can represent any CPWL function as $f(\mathbf{x}) = \sum_k \epsilon_k \max(f_1^k(\mathbf{x}), \ldots, f_{m_k}^k(\mathbf{x}))$, where $f_1^k, \ldots, f_{m_k}^k$ are affine functions, $\epsilon_k = \pm 1$, and $m_k \leq d + 1$. This expansion has also different variations and can be recast as $f(\mathbf{x}) = \sum_k \epsilon_k \max(g_1^k(\mathbf{x}), \ldots, g_{m_k-1}^k(\mathbf{x}))_+$, where $g_1^k, \ldots, g_{m_k-1}^k$ are affine functions, $\epsilon_k = \pm 1$, $m_k \leq d + 1$, and for any $x \in \mathbb{R}$, $(x)_+ \coloneqq \max(x, 0) = \mathrm{ReLU}(x)$.

The basis functions of nonlocal representations are the building blocks of many feedforward neural networks. They play the role of activation functions, including ReLU, leaky ReLU, PReLU, CReLU, and Maxout [155, 172–175].

3.1.3 Stability of the Parameterizations

The stability of a parameterization can be analyzed through various metrics, including the condition number and the Lipschitz constant of appropriate mappings (see Section 3.2 for the mathematical definitions).

Condition Number of the Interpolation Problem: Consider the problem of finding the parameter $\boldsymbol{\theta}$ such that $T\{\boldsymbol{\theta}\}$ interpolates the data $(\mathbf{v}_k, y_k)$. For linear models, this amounts to solving a system of linear equations with unknown $\boldsymbol{\theta}$ and input (y_k). The stability of this problem is usually quantified via its condition number, which is the ratio of the largest to the smallest singular value of the system matrix. For the local representation, the explicit solution is $f = \sum_k y_k \beta_k$ in any dimension. The associated condition number

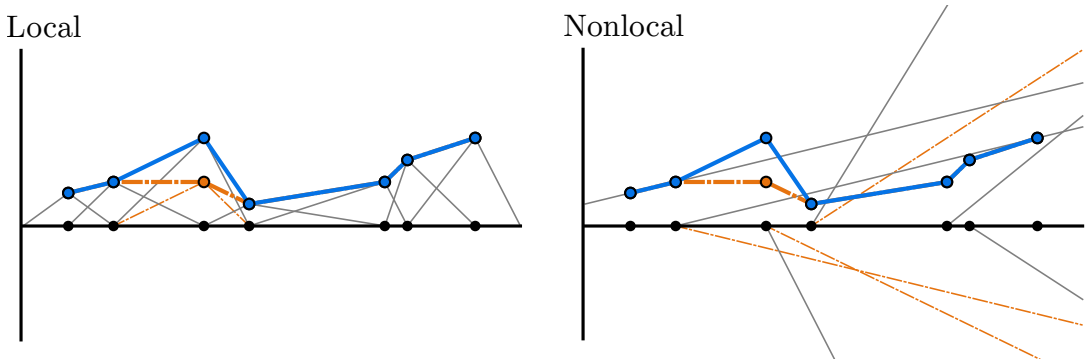

Figure 3.2: Linear expansions of two nearly identical CPWL one-dimensional functions (solid thick blue line and mixed thick orange line). The weighted basis functions are represented with thin lines (solid gray and solid orange respectively if different). Left: the local expansions of the two curves are very similar. Right: the small difference between the two curves induces a great difference in their nonlocal parameterization.

is unity and optimal. The situation is less favorable for nonlocal representations. In Figure 3.2, we illustrate a case where a small change in a single target value implies a significant change of the nonlocal parameter $\boldsymbol{\theta}$ for $T\{\boldsymbol{\theta}\}$ to remain interpolatory. In Appendix 3.8, we prove that the interpolation condition number given K data points is at least $\mathcal{O}(K^{3/2})$ for the nonlocal representation (3.2) in the one-dimensional case. Thus the problem is generally ill-conditioned.

Riesz Basis and Riesz Ratio: In the context of the local parameterization, we consider the parameter space $\ell_2(\mathbb{Z})$ of finite-energy sequences and equip the function space with the L_2 norm. Although the collection (β_k) of the local atoms is not an orthonormal basis, it forms a Riesz basis under weak conditions [176, 177]. This means that (β_k) is the image of an orthonormal basis under a bounded invertible linear operator. This guarantees that the synthesis operator T is a bounded linear bijection from $\ell_2(\mathbb{Z})$ to $\mathcal{H} \subset L_2(\mathbb{R}^d)$ and allows us to define the condition number of T, referred to as the Riesz ratio in the sequel. The Riesz basis is a standard requirement in many signal-processing theories and finite-elements methods [110, 121, 145, 178, 179] within a broad spectrum of applications. To the best of our knowledge, the exact Riesz bounds of linear box splines are not known in high dimensions and the case of arbitrary triangulations has not been addressed in full generality so far.

In the context of a the nonlocal representation, since the basis functions do not belong to $L_2(\mathbb{R}^d)$, the synthesis operator $T\colon \Theta \to \mathcal{H}$ is ill-conditioned, in the sense that a small change of $\boldsymbol{\theta}$ can lead to arbitrarily large changes in $\|T\{\boldsymbol{\theta}\}\|_{L_2}$. In other words, it is not possible to form a Riesz basis with nonlocal CPWL atoms. Consequently, the associated guarantees on stability disappear.

Since the condition number of the interpolation problem is always 1 for the local parame-

terization, we shall only concentrate on the Riesz ratio in the remainder of this chapter, as the latter requires a more involved analysis.

Lipschitz Constant: Beyond the stability of the synthesis operator, in many learning applications, it is also highly desirable to control the stability of the input-output mapping. For instance, neural networks with controlled Lipschitz constants tend to generalize better [180–182], to be more robust against adversarial attacks [183–186], and to be more interpretable [185, 187]. Despite many recent advances to control the Lipschitz constant of deep models [80–82, 188], it remains very challenging to learn compositional models under a Lipschitz constraint [189]. This results from the fact that the determination of the Lipschitz constant of a ReLU network is already NP-hard as soon as there is one hidden layer [48].

3.1.4 Contributions and Outline

In this chapter, we propose to investigate in any dimension the stability of the local parameterization of CPWL functions with the hope to bring detailed results against which other parameterizations could be compared. We define the Riesz ratio in Section 3.2 and establish a few preliminary results in Section 3.3. We then derive in Section 3.4 an upper bound on the Riesz ratio that is applicable to any triangulation and that considers the worst case scenario. We also propose a complementary stochastic framework that addresses better an average behavior instead. In Section 3.5, we show that our upper bound on the Riesz ratio is exact for linear box splines. Finally, in Section 3.6, we show that the local parameterization gives access to the Lipschitz constant of the map and we provide novel fast-to-evaluate upper and lower bounds on the Lipschitz constant.

3.2 Mathematical Preliminaries

3.2.1 Simplicial Continuous and Piecewise-Linear Functions

Definition 3.1. *A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is continuous and piecewise-linear (CPWL) if it is continuous and if there exist distinct affine functions $f_1, f_2, \ldots, f_M$ and subsets $R_1, R_2, \ldots, R_M$ of $\mathbb{R}^d$ such that*

(i) each R_m is closed with nonempty interior;

(ii) for $n \neq m$, R_n and R_m have disjoint interiors;

(iii) the space is partitioned as $\bigcup_{m=1}^{M} R_m = \mathbb{R}^d$;

(iv) the function f agrees with f_m on R_m.

In this chapter, we extend this definition to compact input domains of $\mathbb{R}^d$ and to any function $f\colon \mathbb{R}^d \to \mathbb{R}$ whose restriction to any compact set is CPWL (sometimes referred to as locally piecewise-affine functions [190]).

Definition 3.2. *A set $\mathcal{V} \subset \mathbb{R}^d$ is locally finite if its intersection with any compact set of $\mathbb{R}^d$ is finite*.*

In the sequel, $\mathcal{V}$ will always denote a locally finite set of $\mathbb{R}^d$ indexed by $I \subset \mathbb{N}$ that is not contained in any $(d-1)$-dimensional affine subspace of $\mathbb{R}^d$. To each point $\mathbf{v}_k \in \mathcal{V}$ we associate a target value $y_k \in \mathbb{R}$. Under Definition 3.1, it is not clear how to construct a CPWL function f that satisfies $f(\mathbf{v}_k) = y_k$. The local representation offers a more systematic way to address this problem. This requires first to form a triangulation of the set $\mathcal{V}$.

Partition of the Input Domain into Simplices

A polyhedron is the intersection of finitely many half spaces. A polytope is a bounded polyhedron. Simplices are the polytopes that have the fewest number of faces; in growing number of dimension $d = 0, \dots, 3$ they include points, segments, triangles, and tetrahedrons. Formally, a d-simplex s of $\mathbb{R}^d$ is the convex hull of $(d+1)$ affinely independent vertices

$$s = \operatorname{conv}(\mathbf{v}_1, \dots, \mathbf{v}_{d+1}) = \left\{ \sum_{k=1}^{d+1} \lambda_k \mathbf{v}_k \colon \lambda_k \geq 0, \sum_{k=1}^{d+1} \lambda_k = 1 \right\}. \tag{3.3}$$

A k-face of a simplex is the convex hull of $(k+1)$ of its vertices, which is a k-simplex embedded in $\mathbb{R}^d$. The volume of a simplex admits the explicit form

$$\operatorname{Vol}(s) = \frac{1}{d!} |\det(\mathbf{v}_2 - \mathbf{v}_1, \dots, \mathbf{v}_{d+1} - \mathbf{v}_1)|.$$

Definition 3.3 (adapted from [191])**.** *A triangulation of a locally finite set $\mathcal{V} \subset \mathbb{R}^d$ of points is a collection $\mathcal{S}$ of d-simplices whose vertices are points in $\mathcal{V}$ and such that*

(i) the union of all the simplices equals $\operatorname{conv}(\mathcal{V})$ *(union property);*

(ii) any pair of simplices intersects in a (possibly empty) common face (intersection property).

A triangulation of $\mathcal{V}$ is said to be *full* if all the points of $\mathcal{V}$ are vertices of it, a property that we shall always assume to hold true in the sequel. For a given locally finite set $\mathcal{V} \subset \mathbb{R}^d$ of vertices of dimension d, the existence of a triangulation is granted but in general not unique. In practice, most applications fall into one of the two.

*Note that the meaning of the term "locally finite" depends on the mathematical field.

- **Regular Sampling Locations**: The vertices coincide with the sites of a lattice, for which explicit triangulations are known (*e.g.*, the Kuhn triangulation [192, 193]).

- **Irregular (Random) Sampling Locations**: A Delaunay triangulation always exists in any number of dimensions for a finite set $\mathcal{V}$, and there exist algorithms to compute it [194, 195].

Definition 3.4. *Let $\mathcal{S}$ be a triangulation of a locally finite set $\mathcal{V} \subset \mathbb{R}^d$. The star* $\mathrm{St}(\mathbf{v})$ *of a vertex $\mathbf{v} \in \mathcal{V}$ is the set of those simplices of $\mathcal{S}$ that contain $\mathbf{v}$.*

We define the volume of the star of a vertex as $\mathrm{Vol}(\mathrm{St}(\mathbf{v})) = \sum_{s \in \mathrm{St}(\mathbf{v})} \mathrm{Vol}(s)$. Its cardinality is denoted by $|\mathrm{St}(\mathbf{v})|$.

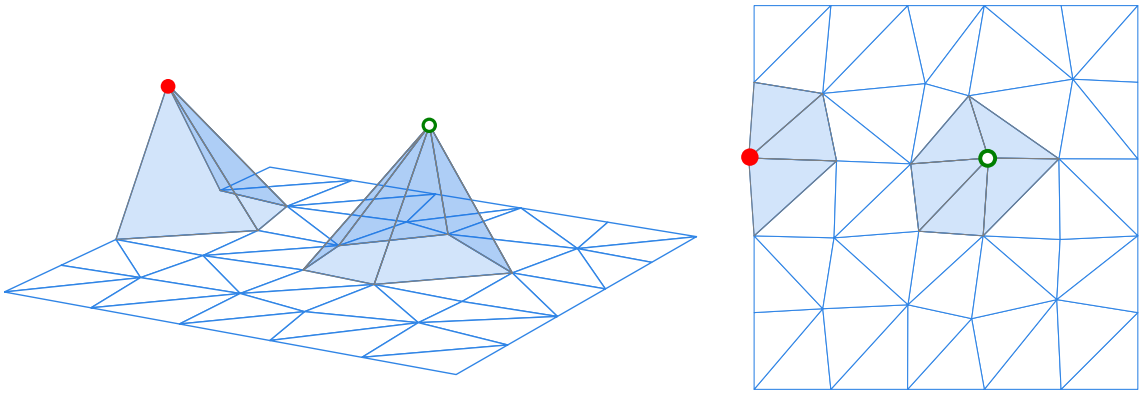

Figure 3.3: Left: 3D view of two hat functions on a finite triangulation. Note that, although the left hat function seems discontinuous, it is continuous on the triangulation. Right: star of two vertices of the triangulation (filled area).

Hat Basis Functions

The set of CPWL functions on a triangulation $\mathcal{S}$ with vertices $\mathcal{V}$ is defined as

$$\mathrm{CPWL}(\mathcal{S}) = \{f \in \mathbb{R}^{\mathrm{conv}(\mathcal{V})} \colon f \text{ is affine on any } s \in \mathcal{S} \text{ and continuous on } \mathrm{conv}(\mathcal{V})\}. \tag{3.4}$$

It is said to be the space of linear simplicial splines on $\mathcal{S}$. Note that the functions in $\mathrm{CPWL}(\mathcal{S})$ are only defined over the convex hull $\mathrm{conv}(\mathcal{V})$ of $\mathcal{V}$, which can range from a compact set to the whole space $\mathbb{R}^d$.

It is known that any polyhedron can be partitioned into simplices [196]. As a result, any CPWL function can be viewed as a linear simplicial spline. Two affine functions that coincide on the vertices of a d-simplex are equal, which means that any element of $\mathrm{CPWL}(\mathcal{S})$ is uniquely determined by the values it assumes at the vertices $\mathcal{V}$ of $\mathcal{S}$. This

leads to the local linear expansion

$$\forall f \in \mathrm{CPWL}(\mathcal{S})\colon f = \sum_{\mathbf{v}\in\mathcal{V}} f(\mathbf{v})\beta^{\mathcal{S}}_{\mathbf{v}}, \tag{3.5}$$

where the hat functions $\beta^{\mathcal{S}}_{\mathbf{v}} \in \mathbb{R}^{\mathrm{conv}(\mathcal{V})}$ (see Figure 3.3) are defined on every simplex $s \in \mathcal{S}$ by

$$\beta^{\mathcal{S}}_{\mathbf{v}\,|s} = \begin{cases} \lambda^s_{\mathbf{v}}, & s \in \mathrm{St}(\mathbf{v}) \\ 0, & \text{otherwise,} \end{cases} \tag{3.6}$$

where $\lambda^s_{\mathbf{v}}$ is the unique affine function that vanishes at all vertices of s but takes value 1 at vertex $\mathbf{v}$. In other words, $\lambda^s_{\mathbf{v}}$ outputs the barycentric coordinate of simplex s attached to vertex $\mathbf{v}$ for a given $\mathbf{x} \in s$. Note that depending on the set of vertices, the hat basis functions might not be defined over the whole $\mathbb{R}^d$ and, in the sequel, for any $f \in \mathbb{R}^{\mathrm{conv}(\mathcal{V})}$ we use the notation $\|f\|_{L_p} = (\int_{\mathbf{x}\in\mathrm{conv}(\mathcal{V})} |f(\mathbf{x})|^p \mathrm{d}\mathbf{x})^{1/p}$. The hat basis functions have many desirable properties such as

- for $\mathbf{u}, \mathbf{v} \in \mathcal{V}, \beta^{\mathcal{S}}_{\mathbf{v}}(\mathbf{u}) = \begin{cases} 1, & \mathbf{v} = \mathbf{u} \\ 0, & \text{otherwise;} \end{cases}$
- continuity;
- compact support $\mathrm{supp}(\beta^{\mathcal{S}}_{\mathbf{v}}) = \bigcup_{s\in\mathrm{St}(\mathbf{v})} s$;
- minimal support among all nonzero functions of $\mathrm{CPWL}(\mathcal{S})$;
- ability to reproduce polynomials of degree up to 1 on $\mathrm{conv}(\mathcal{V})$, so that

$$\forall (\mathbf{a}, b) \in \mathbb{R}^d \times \mathbb{R},\ \forall \mathbf{x} \in \mathrm{conv}(\mathcal{V})\colon \mathbf{a}^T\mathbf{x} + b = \sum_{v\in\mathcal{V}} (\mathbf{a}^T\mathbf{v} + b)\beta^{\mathcal{S}}_{\mathbf{v}}(\mathbf{x}), \tag{3.7}$$

  which includes the partition-of-unity condition $\sum_{v\in\mathcal{V}} \beta^{\mathcal{S}}_{\mathbf{v}} = 1$.

When $\mathrm{St}(\mathbf{v})$ is convex, the hat function simply reads [162]

$$\beta^{\mathcal{S}}_{\mathbf{v}} = \left(\min_{s\in\mathrm{St}(\mathbf{v})} \lambda^s_{\mathbf{v}} \right)_+ . \tag{3.8}$$

3.2.2 Riesz Basis and Riesz Ratio

For a set $I \subset \mathbb{N}$, we denote by $\ell_2(I)$ the set of complex-valued sequences indexed by I with finite energy.

Definition 3.5. *Let $\mathcal{H}$ be a separable Hilbert space over $\mathbb{C}$ and $I \subset \mathbb{N}$. A collection of functions $\{\varphi_k\}_{k\in I}$ in $\mathcal{H}$ is a Riesz basis if*

(i) $\overline{\text{Span}(\{\varphi_k\}_{k\in I})} = \mathcal{H}$ *(completeness),*

(ii) there exist $0 < A \leq B < +\infty$ *such that, for any* $c \in \ell_2(I)$*,*

$$A\|c\|_{\ell_2} \leq \|\sum_{k\in I} c_k\varphi_k\|_{L_2} \leq B\|c\|_{\ell_2} \text{ (Riesz-sequence property)}, \tag{3.9}$$

where $\|c\|_{\ell_2} = \left(\sum_{k\in I} |c_k|^2\right)^{1/2}$.

The tightest constants A and B that satisfy (3.9) are called the *Riesz bounds*.

Riesz Ratio: Consider the linear synthesis operator $T\colon c \mapsto \sum_{k\in I} c_k\varphi_k$, where $\{\varphi_k\}_{k\in I}$ is a Riesz basis. The synthesis operator T is a bounded linear bijection, which means that there is a unique and stable link between the parameters and the functions being generated. The condition number of the linear operator T is defined as

$$\kappa(T) = \sup_{\mathbf{c}_1,\mathbf{c}_2\neq\mathbf{0}} \frac{\|T\{\mathbf{c}_1\}\|_{L_2}/\|T\{\mathbf{c}_2\}\|_{L_2}}{\|\mathbf{c}_1\|_{\ell_2}/\|\mathbf{c}_2\|_{\ell_2}}. \tag{3.10}$$

It follows from (3.9) that $\kappa(T)$ is the ratio B/A of the Riesz bounds of $\{\varphi_k\}_{k\in I}$. Hence, we refer to this number as the Riesz ratio. The Riesz ratio is 1 if and only if the collection of functions $(\varphi_k)_{k\in I}$ forms an orthonormal basis (up to a scaling factor).

When the collection of functions is formed by the multiindex shifts of a single generating function ($\{\varphi_{\mathbf{k}}\} = \{\varphi(\cdot - \mathbf{k})\colon \mathbf{k} \in \mathbb{Z}^d\}$), the Riesz-sequence property is characterized via the discrete-time Fourier transform $\widehat{g}$ of the sampled autocorrelation of φ, as given by

$$\widehat{g}\colon \boldsymbol{\omega} \mapsto \sum_{\mathbf{k}\in\mathbb{Z}^d} \langle\varphi, \varphi(\cdot - \mathbf{k})\rangle \mathrm{e}^{-\mathrm{i}\mathbf{k}^T\boldsymbol{\omega}}. \tag{3.11}$$

In this uniform scenario, the Fourier equivalent of the Riesz-sequence condition is [121, 197]

$$0 < A^2 = \operatorname*{ess\,inf}_{\boldsymbol{\omega}\in[0,2\pi]^d} \widehat{g}(\boldsymbol{\omega}) \leq B^2 = \operatorname*{ess\,sup}_{\boldsymbol{\omega}\in[0,2\pi]^d} \widehat{g}(\boldsymbol{\omega}) < +\infty. \tag{3.12}$$

3.2.3 Lipschitz Constant

A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is L_p-Lipschitz, with $L_p \in \mathbb{R}_+$ and $p \geq 1$, if, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, it holds that

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L_p\|\mathbf{x} - \mathbf{y}\|_p, \tag{3.13}$$

and $\text{Lip}_p(f) \in \mathbb{R}_+$ is the smallest constant L_p for which f is L_p-Lipschitz. It is well-known that a CPWL function with finitely many affine pieces is Lipschitz continuous. In

particular, the Lipschitz constant of the CPWL function f as defined in Definition 3.1 is given by the maximum Lipschitz constant of its affine pieces, so that

$$\mathrm{Lip}_p(f) = \max_{m=1,\dots,M} \mathrm{Lip}_p(f_m). \tag{3.14}$$

3.3 Affine Functions on Simplices

Considerations on affine functions on simplices will help us lay the foundations of the analysis of the stability of the local parameterization on simplicial partitions (Sections 3.4 and 3.5).

Proposition 3.1. *Let $f : \mathbb{R}^d \to \mathbb{C}$ be an affine function and $s = \mathrm{conv}(\mathbf{v}_1, \dots, \mathbf{v}_{d+1})$ a d-simplex. Then*

$$\frac{\mathrm{Vol}(s)}{(d+2)(d+1)} \sum_{k=1}^{d+1} |f(\mathbf{v}_k)|^2 \leq \int_{\mathbf{x}\in s} |f(\mathbf{x})|^2 \mathrm{d}\mathbf{x} \leq \frac{\mathrm{Vol}(s)}{(d+1)} \sum_{k=1}^{d+1} |f(\mathbf{v}_k)|^2. \tag{3.15}$$

Prior to proving Proposition 3.1, we provide a series of useful results regarding the computation of some integrals of affine functions over simplices.

For a linear function $f\colon \mathbb{R}^d \to \mathbb{R}$, an integer $p \in \mathbb{N}$, and a d-simplex $s = \mathrm{conv}(\mathbf{v}_1, \dots, \mathbf{v}_{d+1})$, it is known that [198, 199]

$$\int_s f(\mathbf{x})^p \mathrm{d}\mathbf{x} = \mathrm{Vol}(s) \binom{p+d}{d}^{-1} \sum_{\mathbf{k}\in\mathbb{N}^{d+1}, |\mathbf{k}|=p} f(\mathbf{v}_1)^{k_1} \cdots f(\mathbf{v}_{d+1})^{k_{d+1}}, \tag{3.16}$$

where we use the notation $\mathbf{k} = (k_1, \dots, k_{d+1})$ and $|\mathbf{k}| := k_1 + \cdots + k_{d+1}$. We can extend this to affine functions.

Lemma 3.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an affine function and $s = \mathrm{conv}(\mathbf{v}_1, \dots, \mathbf{v}_{d+1})$ a d-simplex. For any $p \in \mathbb{N}$, we have that*

$$\int_s f(\mathbf{x})^p \mathrm{d}\mathbf{x} = \mathrm{Vol}(s) \binom{p+d}{d}^{-1} \sum_{\mathbf{k}\in\mathbb{N}^{d+1}, |\mathbf{k}|=p} f(\mathbf{v}_1)^{k_1} \cdots f(\mathbf{v}_{d+1})^{k_{d+1}}. \tag{3.17}$$

Proof. If the affine function f is not constant, then it can be written as $f(\mathbf{x}) = \mathbf{a}^T(\mathbf{x} - \mathbf{x}_0)$.

Equation (3.16) can be applied after a change of variable, as in

$$\int_s f(\mathbf{x})^p \mathrm{d}\mathbf{x} = \int_s (\mathbf{a}^T(\mathbf{x}-\mathbf{x}_0))^p \mathrm{d}\mathbf{x} = \int_{s-\mathbf{x}_0} (\mathbf{a}^T\mathbf{y})^p \mathrm{d}\mathbf{y} \tag{3.18}$$

$$= \mathrm{Vol}(s-\mathbf{x}_0)\binom{p+d}{d}^{-1} \sum_{\mathbf{k}\in\mathbb{N}^{d+1},|\mathbf{k}|=p} (\mathbf{a}^T(\mathbf{v}_1-\mathbf{x}_0))^{k_1}\cdots(\mathbf{a}^T(\mathbf{v}_{d+1}-\mathbf{x}_0))^{k_{d+1}} \tag{3.19}$$

$$= \mathrm{Vol}(s)\binom{p+d}{d}^{-1} \sum_{\mathbf{k}\in\mathbb{N}^{d+1},|\mathbf{k}|=p} f(\mathbf{v}_1)^{k_1}\cdots f(\mathbf{v}_{d+1})^{k_{d+1}}. \tag{3.20}$$

where $s-\mathbf{x}_0 \coloneqq \{\mathbf{x}-\mathbf{x}_0 \colon \mathbf{x}\in s\}$. If now the affine function f is constant with $f(\mathbf{x})=b$, then $\int_s f(\mathbf{x})^p \mathrm{d}\mathbf{x} = \mathrm{Vol}(s)b^p$. We also have that

$$\binom{p+d}{d}^{-1} \sum_{\mathbf{k}\in\mathbb{N}^{d+1},|\mathbf{k}|=p} f(\mathbf{v}_1)^{k_1}\cdots f(\mathbf{v}_{d+1})^{k_{d+1}} = \binom{p+d}{d}^{-1} \sum_{\mathbf{k}\in\mathbb{N}^{d+1},|\mathbf{k}|=p} b^p = b^p, \tag{3.21}$$

where we have used that $\sum_{\mathbf{k}\in\mathbb{N}^{d+1},|\mathbf{k}|=p} 1 = \binom{p+d}{d}$. This number is known in combinatorics as the combinations with replacement [200]. □

We can now deduce an important property of the hat functions.

Proposition 3.2. *The L_p norm of the hat function $\beta^{\mathcal{S}}_{\mathbf{v}}$ only depends on the dimension d and the volume of its support. It reads*

$$\|\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_p} = \left(\binom{p+d}{d}^{-1}\mathrm{Vol}(\mathrm{St}(\mathbf{v}))\right)^{1/p}. \tag{3.22}$$

Proof. We split the integral over the simplices of the support of $\beta^{\mathcal{S}}_{\mathbf{v}}$ and apply Lemma 3.1, which reads

$$\begin{aligned}
\|\beta^{\mathcal{S}}_{\mathbf{v}}\|^p_{L_p} &= \int_{\mathrm{conv}(\mathcal{V})} |\beta^{\mathcal{S}}_{\mathbf{v}}(\mathbf{x})|^p \mathrm{d}\mathbf{x} = \int_{\mathrm{conv}(\mathcal{V})} \beta^{\mathcal{S}}_{\mathbf{v}}(\mathbf{x})^p \mathrm{d}\mathbf{x} \\
&= \sum_{s\in\mathcal{S}} \int_s \beta^{\mathcal{S}}_{\mathbf{v}}(\mathbf{x})^p \mathrm{d}\mathbf{x} = \sum_{s\in\mathrm{St}(\mathbf{v})} \int_s \beta^{\mathcal{S}}_{\mathbf{v}}(\mathbf{x})^p \mathrm{d}\mathbf{x} \\
&= \sum_{s\in\mathrm{St}(\mathbf{v})} \binom{p+d}{d}^{-1} \mathrm{Vol}(s) = \binom{p+d}{d}^{-1} \mathrm{Vol}(\mathrm{St}(\mathbf{v})).
\end{aligned} \tag{3.23}$$

□

In the sequel, we shall make use of Proposition 3.2 through the two following relations:

- the L_2 norm

$$\|\beta_{\mathbf{v}}^{\mathcal{S}}\|_{L_2}^2 = \frac{2\text{Vol}(\text{St}(\mathbf{v}))}{(d+1)(d+2)}; \tag{3.24}$$

- the inner-product relation

$$\langle \beta_{\mathbf{v}}^{\mathcal{S}}, \sum_{\mathbf{u}\in\mathcal{V}} \beta_{\mathbf{u}}^{\mathcal{S}} \rangle_{\text{conv}(\mathcal{V})} = \langle \beta_{\mathbf{v}}^{\mathcal{S}}, 1 \rangle_{\text{conv}(\mathcal{V})} = \|\beta_{\mathbf{v}}^{\mathcal{S}}\|_{L_1} = \frac{\text{Vol}(\text{St}(\mathbf{v}))}{(d+1)}, \tag{3.25}$$

  where $\langle f, h \rangle_{\text{conv}(\mathcal{V})} = \int_{\mathbf{x}\in\text{conv}(\mathcal{V})} \overline{f(\mathbf{x})} h(\mathbf{x}) \mathrm{d}\mathbf{x}$ and where the first equality results from the partition of unity of the hat functions.

When $p = 2$, the integral in Lemma 3.1 is a quadratic form of the value of the function on the vertices and admits the matrix form shown in Lemma 3.2.

Lemma 3.2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an affine function, $s = \text{conv}(\mathbf{v}_1, \ldots, \mathbf{v}_{d+1})$ a d-simplex, and $\mathbf{f}_s = (f(\mathbf{v}_1), \cdots, f(\mathbf{v}_{d+1})) \in \mathbb{R}^{d+1}$. Then,*

$$\int_s f(\mathbf{x})^2 \mathrm{d}\mathbf{x} = \frac{\text{Vol}(s)}{(d+1)(d+2)} \mathbf{f}_s^T \mathbf{P}_{d+1} \mathbf{f}_s, \tag{3.26}$$

where

$$\mathbf{P}_{d+1} = \mathbf{1}_{d+1} + \mathbf{I}_{d+1} = \begin{bmatrix} 2 & 1 & \dots & 1 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 2 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}, \tag{3.27}$$

and where $\mathbf{1}_{d+1} \in \mathbb{R}^{(d+1)\times(d+1)}$ is the matrix of ones and $\mathbf{I}_{d+1} \in \mathbb{R}^{(d+1)\times(d+1)}$ is the identity matrix.

Proof. Following Lemma 3.1, on one hand we have that

$$\begin{aligned} \int_s f(\mathbf{x})^2 \mathrm{d}\mathbf{x} &= \text{Vol}(s) \binom{2+d}{d}^{-1} \sum_{\mathbf{k}\in\mathbb{N}^{d+1}, |\mathbf{k}|=2} f(\mathbf{v}_1)^{k_1} \cdots f(\mathbf{v}_{d+1})^{k_{d+1}} \\ &= \frac{2\text{Vol}(s)}{(d+1)(d+2)} 1/2 \left(\sum_{p,q=1}^{d+1} f(\mathbf{v}_p) f(\mathbf{v}_q) + \sum_{p=1}^{d+1} f(\mathbf{v}_p)^2 \right) \\ &= \frac{\text{Vol}(s)}{(d+1)(d+2)} \left(\left(\sum_{p=1}^{d+1} f(\mathbf{v}_q) \right)^2 + \sum_{p=1}^{d+1} f(\mathbf{v}_p)^2 \right). \end{aligned} \tag{3.28}$$

On the other hand, we have that

$$\mathbf{f}_s^T \mathbf{P}_{d+1} \mathbf{f}_s = \sum_{p=1}^{d+1} f(\mathbf{v}_p) \left(\sum_{q=1}^{d+1} f(\mathbf{v}_q) + f(\mathbf{v}_p) \right) \tag{3.29}$$

$$= \left(\sum_{p=1}^{d+1} f(\mathbf{v}_p) \right)^2 + \sum_{p=1}^{d+1} f(\mathbf{v}_p)^2. \tag{3.30}$$

□

Lemma 3.2 can be extended to pairs of complex-valued functions.

Lemma 3.3. *Let $f, g : \mathbb{R}^d \to \mathbb{C}$ be affine functions, $s = \mathrm{conv}(\mathbf{v}_1, \ldots, \mathbf{v}_{d+1})$ a d-simplex, $\mathbf{f}_s = (f(\mathbf{v}_1), \cdots, f(\mathbf{v}_{d+1})) \in \mathbb{R}^{d+1}$, and $\mathbf{g}_s = (g(\mathbf{v}_1), \cdots, g(\mathbf{v}_{d+1})) \in \mathbb{R}^{d+1}$. It holds that*

$$\int_s \overline{f(\mathbf{x})} g(\mathbf{x}) \mathrm{d}\mathbf{x} = \frac{\mathrm{Vol}(s)}{(d+1)(d+2)} \mathbf{f}_s^H \mathbf{P}_{d+1} \mathbf{g}_s, \tag{3.31}$$

where

$$\mathbf{P}_{d+1} = \mathbf{1}_{d+1} + \mathbf{I}_{d+1} \in \mathbb{R}^{(d+1)\times(d+1)}. \tag{3.32}$$

To prove Lemma 3.3, we first consider real-valued functions f and g and apply Lemma 3.2 to the left-hand side of the equality $2fg = \big((f+g)^2 - f^2 - g^2\big)$. The announced result is then reached because $\mathbf{P}_{d+1}$ is a symmetric matrix. The generalization to complex-valued functions is directly obtained via the decomposition of f and g into their real and imaginary parts.

Proof of Proposition 3.1. The matrix $\mathbf{P}_{d+1}$ defined in Lemma 3.2 is a symmetric circulant matrix generated by the vector $(2, 1, \ldots, 1)$. Its eigenvalues are known to be [201]

$$\lambda_m = 2 + \sum_{n=1}^{d} \zeta_{d+1}^{mn} \quad \text{with } m = 1, \ldots, d+1 \text{ and where } \zeta_{d+1} = \mathrm{e}^{\mathrm{i}\frac{2\pi}{d+1}}. \tag{3.33}$$

These expressions are further simplified to

$$\lambda_m = 1 + \sum_{n=1}^{d+1} \zeta_{d+1}^{mn} = \begin{cases} d+2, & m = d+1 \\ 1, & \text{otherwise,} \end{cases} \tag{3.34}$$

which shows that $\min_{m\in\{1,\ldots,d+1\}}(\lambda_m) = 1$ and $\max_{m\in\{1,\ldots,d+1\}}(\lambda_m) = (d+2)$. As a result, for any $\mathbf{c} \in \mathbb{C}^{d+1}$

$$\|\mathbf{c}\|_2^2 \leq \mathbf{c}^H \mathbf{P}_{d+1} \mathbf{c} \leq (d+2)\|\mathbf{c}\|_2^2. \tag{3.35}$$

We now conclude by applying Lemma 3.3. □

The condition number $\sqrt{d+2}$ given by the inequalities in Proposition 3.1 depends on the dimension d. However, the eigenvalues of $\mathbf{P}_{d+1}$ are all 1 except for the largest one, given by $(d+2)$. This fact is used to derive Lemma 3.4, which offers a stochastic view on the problem.

Lemma 3.4. *Let $s = \mathrm{conv}(\mathbf{v}_1, \ldots, \mathbf{v}_{d+1})$ be a d-simplex, $\boldsymbol{C} = (C_1, \ldots, C_{d+1})$ a random vector of $\mathbb{R}^{d+1}$ with independent zero-mean components and with $\mathbb{E}(\|\boldsymbol{C}\|_2^2) \leq +\infty$, and $f_{\boldsymbol{C}} : \mathbb{R}^d \to \mathbb{R}$ the unique affine function such that $f_{\boldsymbol{C}}(\mathbf{v}_k) = C_k$ for $k = 1, \ldots, d+1$. Then,*

$$\mathbb{E}\left(\int_s f(\mathbf{x})^2 \mathrm{d}\mathbf{x}\right) = \frac{2\mathrm{Vol}(s)}{(d+1)(d+2)} \mathbb{E}(\|\boldsymbol{C}\|_2^2). \tag{3.36}$$

Proof. Given the matrix $\mathbf{P}_{d+1}$ defined in Lemma 3.2, it holds that $\boldsymbol{C}^T \mathbf{P}_{d+1} \boldsymbol{C} = \sum_{1 \leq k,l \leq d+1} (P_{d+1})_{kl} C_l C_k$. Using the independence of the zero-mean entries of $\boldsymbol{C}$, it follows that $\mathbb{E}(\boldsymbol{C}^T \mathbf{P}_{d+1} \boldsymbol{C}) = \sum_{k=1}^{d+1} (P_{d+1})_{kk} \mathbb{E}(C_k^2) = 2\mathbb{E}(\|\boldsymbol{C}\|_2^2)$. The conclusion follows from the formula given in Lemma 3.2. □

3.4 Riesz Ratio on Arbitrary Triangulations

3.4.1 Triangulations with Any Number of Vertices

Triangulations in high dimensions have complex combinatorial structures that can induce a wide range of behaviors with the usual descriptors, for instance, shape of the simplices, degree of the vertices, number of simplices shared by $2, \ldots, d$ vertices. Fortunately, the necessary and sufficient condition that a triangulation must satisfy for the hat functions to form a Riesz basis only relies on the volume of the star of the vertices, as outlined in Theorem 3.1. This result can be seen as an extension of [202, Theorem 3.1], which gives a similar bound for the condition number of the mass matrix for the hat functions on triangulations with finitely many vertices. The present result applies to any triangulation, including those with infinitely many vertices. This will be used to determine the exact Riesz bounds for linear box splines in Section 3.5.

Theorem 3.1. *Let $\mathcal{S}$ be a triangulation of a locally finite set $\mathcal{V} = \{\mathbf{v}_k\}_{k \in I}$ of vertices in $\mathbb{R}^d$ and let $(\beta_{\mathbf{v}}^{\mathcal{S}})_{\mathbf{v} \in \mathcal{V}}$ be the corresponding hat functions. Then, the following statements are equivalent:*

(i) the collection of functions $(\beta_{\mathbf{v}}^{\mathcal{S}})_{\mathbf{v} \in \mathcal{V}}$ forms a Riesz basis of the space $\mathrm{CPWL}(\mathcal{S}) \cap L_2(\mathbb{R}^d)$;

(ii) $\begin{cases} V_{\mathrm{inf}}^{\mathrm{St}} = \inf_{\mathbf{v} \in \mathcal{V}} \mathrm{Vol}(\mathrm{St}(\mathbf{v})) > 0 \\ V_{\mathrm{sup}}^{\mathrm{St}} = \sup_{\mathbf{v} \in \mathcal{V}} \mathrm{Vol}(\mathrm{St}(\mathbf{v})) < +\infty; \end{cases}$

(iii) $\begin{cases} \inf_{\mathbf{v}\in\mathcal{V}} \|\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_2} > 0 \\ \sup_{\mathbf{v}\in\mathcal{V}} \|\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_2} < +\infty. \end{cases}$

When these statements hold, for any $c \in \ell_2(I)$ we have that

$$\sqrt{\frac{V^{\mathrm{St}}_{\mathrm{inf}}}{(d+1)(d+2)}}\|c\|_{\ell_2} \leq \|\sum_{v\in\mathcal{V}} c_{\mathbf{v}}\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_2} \leq \sqrt{\frac{V^{\mathrm{St}}_{\mathrm{sup}}}{(d+1)}}\|c\|_{\ell_2} \tag{3.37}$$

or, equivalently, that

$$\frac{1}{\sqrt{2}} \inf_{\mathbf{v}\in\mathcal{V}} \|\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_2}\|c\|_{\ell_2} \leq \|\sum_{v\in\mathcal{V}} c_{\mathbf{v}}\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_2} \leq \sqrt{\frac{d+2}{2}} \sup_{\mathbf{v}\in\mathcal{V}} \|\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_2}\|c\|_{\ell_2}. \tag{3.38}$$

The Riesz ratio r satisfies

$$r \leq \sqrt{d+2}\sqrt{\frac{V^{\mathrm{St}}_{\mathrm{sup}}}{V^{\mathrm{St}}_{\mathrm{inf}}}} = \sqrt{d+2} \times \frac{\sup_{\mathbf{v}\in\mathcal{V}} \|\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_2}}{\inf_{\mathbf{v}\in\mathcal{V}} \|\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_2}}. \tag{3.39}$$

Proof. The equivalence $(ii) \Leftrightarrow (iii)$ is a direct consequence of (3.24).

We now show that $(i) \Rightarrow (iii)$. We consider sequences that are zero everywhere, but at one location, and use the Riesz-sequence property to deduce that, for any $\mathbf{v} \in \mathcal{V}$,

$$A \leq \|\beta^{\mathcal{S}}_{\mathbf{v}}\|_{L_2} \leq B, \tag{3.40}$$

where $0 \leq A \leq B \leq +\infty$ are the Riesz bounds.

We now prove $(ii) \Rightarrow (i)$. Let $c \in \ell_2(I)$, $f = \sum_{\mathbf{v}\in\mathcal{V}} c_{\mathbf{v}}\beta^{\mathcal{S}}_{\mathbf{v}}$. We have that

$$\begin{aligned}
\|f\|^2_{L_2} &= \int_{\mathrm{conv}(\mathcal{V})} |f(\mathbf{x})|^2 \mathrm{d}\mathbf{x} = \sum_{s\in\mathcal{S}} \int_s |f(\mathbf{x})|^2 \mathrm{d}\mathbf{x} \\
&\leq \frac{1}{(d+1)} \sum_{s\in\mathcal{S}} \sum_{\mathbf{v}\in s\cap\mathcal{V}} \mathrm{Vol}(s)|f(\mathbf{v})|^2 \\
&= \frac{1}{(d+1)} \sum_{\mathbf{v}\in\mathcal{V}} \sum_{s\in\mathrm{St}(\mathbf{v})} \mathrm{Vol}(s)|f(\mathbf{v})|^2 \\
&= \frac{1}{(d+1)} \sum_{\mathbf{v}\in\mathcal{V}} |f(\mathbf{v})|^2 \sum_{S\in\mathrm{St}(\mathbf{v})} \mathrm{Vol}(s) \\
&= \frac{1}{(d+1)} \sum_{\mathbf{v}\in\mathcal{V}} \mathrm{Vol}(\mathrm{St}(\mathbf{v}))|f(\mathbf{v})|^2 \\
&\leq \frac{\sup_{\mathbf{v}\in\mathcal{V}}(\mathrm{Vol}(\mathrm{St}(\mathbf{v})))}{(d+1)} \|c\|^2_{\ell_2},
\end{aligned} \tag{3.41}$$

where we have applied Proposition 3.1 and interchanged the order of the double summation with positive arguments (special case of Tonelli's theorem). Similarly,

$$\begin{aligned}
\|f\|_{L_2}^2 &= \int_{\mathrm{conv}(\mathcal{V})} |f(\mathbf{x})|^2 \mathrm{d}\mathbf{x} = \sum_{s\in\mathcal{S}} \int_s |f(\mathbf{x})|^2 \mathrm{d}\mathbf{x} \\
&\geq \frac{1}{(d+1)(d+2)} \sum_{s\in\mathcal{S}} \sum_{\mathbf{v}\in s\cap\mathcal{V}} \mathrm{Vol}(s)|f(\mathbf{v})|^2 \\
&= \frac{1}{(d+1)(d+2)} \sum_{\mathbf{v}\in\mathcal{V}} \sum_{s\in\mathrm{St}(\mathbf{v})} \mathrm{Vol}(s)|f(\mathbf{v})|^2 \\
&= \frac{1}{(d+1)(d+2)} \sum_{\mathbf{v}\in\mathcal{V}} \mathrm{Vol}(\mathrm{St}(\mathbf{v}))|f(\mathbf{v})|^2 \\
&\geq \frac{\inf_{\mathbf{v}\in\mathcal{V}}(\mathrm{Vol}(\mathrm{St}(\mathbf{v})))}{(d+1)(d+2)} \|c\|_{\ell_2}^2. \qquad (3.42)
\end{aligned}$$

□

Theorem 3.1 provides a quantitative way to compare the stability of triangulations. In particular, a triangulation is good when the ratio $\frac{V_{\mathrm{sup}}^{\mathrm{St}}}{V_{\mathrm{inf}}^{\mathrm{St}}}$ is close to 1, which is an indicator of how uniform the triangulation is.

From Theorem 3.1, we deduce a stronger condition that is sufficient for the Riesz property to hold and that gives further insight into the problem. Since $\mathrm{Vol}(\mathrm{St}(\mathbf{v})) = \sum_{s\in\mathrm{St}(\mathbf{v})} \mathrm{Vol}(s)$, we have that

$$\inf_{s\in\mathcal{S}} \mathrm{Vol}(s) \times \inf_{\mathbf{v}\in\mathcal{V}} |\mathrm{St}(\mathbf{v})| \leq V_{\mathrm{inf}}^{\mathrm{St}} \leq V_{\mathrm{sup}}^{\mathrm{St}} \leq \sup_{s\in\mathcal{S}} \mathrm{Vol}(s) \times \sup_{\mathbf{v}\in\mathcal{V}} |\mathrm{St}(\mathbf{v})|. \qquad (3.43)$$

This means that the hat functions form a Riesz basis whenever the degree of the vertices is upper bounded and the volume of the simplices is upper and lower bounded. This condition, however, is not necessary.

Theorem 3.1 does not contain direct information on the tightness of the bounds. Yet, we shall prove in Section 3.5 that, when the hat functions are shifts of a linear box spline, the given bounds are optimal.

The upper bound on the Riesz ratio given in Theorem 3.1 behaves as $\mathcal{O}(\sqrt{d})$. In Corollary 3.1, we propose a stochastic perspective that conveys the intuition that the upper Riesz bound B only accounts for very rare behaviors, all the more in high dimensions.

Corollary 3.1. *Let $\mathcal{S}$ be a triangulation of a locally finite set $\mathcal{V} = \{\mathbf{v}_k\}_{k\in I}$ of vertices in $\mathbb{R}^d$ and let $(\beta_{\mathbf{v}}^{\mathcal{S}})_{\mathbf{v}\in\mathcal{V}}$ be the corresponding hat functions. In addition, suppose that*

$$\begin{cases} V_{\mathrm{inf}}^{\mathrm{St}} = \inf_{\mathbf{v}\in\mathcal{V}} \mathrm{Vol}(\mathrm{St}(\mathbf{v})) > 0 \\ V_{\mathrm{sup}}^{\mathrm{St}} = \sup_{\mathbf{v}\in\mathcal{V}} \mathrm{Vol}(\mathrm{St}(\mathbf{v})) < +\infty. \end{cases} \qquad (3.44)$$

Let $C \in \ell_2(I)$ be a random sequence with independent zero-mean entries and such that $\mathbb{E}(\|C\|_{\ell_2}^2) \leq +\infty$. Then,

$$A^2\mathbb{E}(\|C\|_{\ell_2}^2) \leq \mathbb{E}\left(\|\sum_{\mathbf{v}\in\mathcal{V}} C_{\mathbf{v}}\beta_{\mathbf{v}}^{\mathcal{S}}\|_{L_2}^2\right) \leq 2A^2\left(\frac{V_{\text{sup}}^{\text{St}}}{V_{\text{inf}}^{\text{St}}}\right)\mathbb{E}(\|C\|_{\ell_2}^2), \tag{3.45}$$

where $A = \sqrt{\frac{V_{\text{inf}}^{\text{St}}}{(d+1)(d+2)}}$ is the lower Riesz bound of $(\beta_{\mathbf{v}}^{\mathcal{S}})_{\mathbf{v}\in\mathcal{V}}$ given in Theorem 3.1.

Proof. The inequalities in (3.45) follow from Lemma 3.4 combined with the proof of Theorem 3.1. □

Within the setting of Corollary 3.1, it can be inferred from (3.45) that the standard deviation of the random variable $\|\sum_{\mathbf{v}\in\mathcal{V}} C_{\mathbf{v}}\beta_{\mathbf{v}}^{\mathcal{S}}\|_{L_2}$ is upper bounded by a quantity that varies with d in a similar way as the lower Riesz bound A given in Theorem 3.1. This yields the intuition that, in this random setting, the typical behavior is better represented by the deterministic lower bound, and that the deterministic upper Riesz bound is only rarely approached in high-dimensional input spaces.

3.4.2 Triangulations with Finitely Many Vertices

For a triangulation $\mathcal{S}$ of a finite set $\mathcal{V}$ of vertices in d dimensions, the hat functions always form a Riesz basis. Indeed, since we assumed that $\mathcal{V}$ cannot be contained in any $(d-1)$-dimensional affine subspace of $\mathbb{R}^d$, all simplices of $\mathcal{S}$ are not degenerated and the conditions of Theorem 3.1 are fulfilled. We now apply our formalism to recover a classical result from the finite-element literature, for example [202, Theorem 3.1].

Theorem 3.2. *[see also [202, Theorem 3.1.]] Let $\mathcal{S}$ be a triangulation of a finite set $\mathcal{V}$ of dimension d of vertices. The corresponding hat functions form a Riesz basis of* $\text{CPWL}(\mathcal{S})$ *with Riesz bounds*

$$A = \sqrt{\frac{\lambda_{\min}(\mathbf{M})}{(d+1)(d+2)}} \quad \text{and } B = \sqrt{\frac{\lambda_{\max}(\mathbf{M})}{(d+1)(d+2)}}\,, \tag{3.46}$$

where the matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is symmetric and defined as

$$[\mathbf{M}]_{pq} = \begin{cases} 2\text{Vol}(\text{St}(\mathbf{v}_p)), & p = q \\ \text{Vol}(\text{St}(\mathbf{v}_p) \cap \text{St}(\mathbf{v}_q)), & \textit{otherwise}, \end{cases} \tag{3.47}$$

and where $\lambda_{\min}(\mathbf{M})$ and $\lambda_{\max}(\mathbf{M})$ are the smallest and largest eigenvalues of $\mathbf{M}$.

Proof. Let $f = \sum_{\mathbf{v}\in\mathcal{V}} c_{\mathbf{v}}\beta_{\mathbf{v}}^{\mathcal{S}}$. The coefficients $c_{\mathbf{v}}$ are ordered to form a vector $\mathbf{c} \in \mathbb{R}^{|\mathcal{V}|}$.

To each $s \in \mathcal{S}$ we associate a matrix $\mathbf{L}_s \in \mathbb{R}^{(d+1)\times|\mathcal{V}|}$ such that $\mathbf{L}_s\mathbf{c} \in \mathbb{R}^{d+1}$ contains the coefficients associated with the vertices of s. By invoking Lemma 3.2, we obtain

$$\begin{aligned}\|f\|_{L_2}^2 &= \int_{\mathrm{conv}(\mathcal{V})} |f(\mathbf{x})|^2 \mathrm{d}\mathbf{x} = \sum_{s\in\mathcal{S}} \int_s |f(\mathbf{x})|^2 \mathrm{d}\mathbf{x} \\ &= \frac{1}{(d+1)(d+2)} \sum_{s\in\mathcal{S}} \mathrm{Vol}(s) (\mathbf{L}_s\mathbf{c})^H \mathbf{P}_{d+1} (\mathbf{L}_s\mathbf{c}) \\ &= \frac{1}{(d+1)(d+2)} \mathbf{c}^H \left(\sum_{s\in\mathcal{S}} \mathrm{Vol}(s) \mathbf{L}_s^T \mathbf{P}_{d+1} \mathbf{L}_s \right) \mathbf{c} \\ &= \frac{1}{(d+1)(d+2)} \mathbf{c}^H \mathbf{M} \mathbf{c}, \end{aligned} \tag{3.48}$$

where $\mathbf{M} = \sum_{s\in\mathcal{S}} \mathrm{Vol}(s) \mathbf{L}_s^T \mathbf{P}_{d+1} \mathbf{L}_s$. Let $1 \le p, q \le |\mathcal{V}|$. For $p \neq q$, each entry (p, q) of $\mathrm{Vol}(s)\mathbf{L}_s^T \mathbf{P}_{d+1} \mathbf{L}_s$ is given by

- $\mathrm{Vol}(s)$, if and only if $\mathbf{v}_p$ and $\mathbf{v}_q$ are in s or, equivalently, $s \in \mathrm{St}(\mathbf{v}_p) \cap \mathrm{St}(\mathbf{v}_q)$;
- 0, otherwise.

Now, for $p = q$, each term $\mathrm{Vol}(s)\mathbf{L}_s^T \mathbf{P}_{d+1} \mathbf{L}_s$ of the sum has the entry (p, p) be

- $2\mathrm{Vol}(s)$, if and only if $\mathbf{v}_p$ is in s or, equivalently $s \in \mathrm{St}(\mathbf{v}_p)$;
- 0, otherwise.

This shows that $\mathbf{M}$ is the matrix given in Theorem 3.2. □

In the previously cited work [202, Theorem 3.1], the basis functions are attached to interior vertices, which corresponds to zero boundary conditions in our framework. For finite elements, the matrix $\mathbf{M}$ is usually referred to as the mass matrix: it is the Gram matrix of the set of basis functions. For hat basis functions, this matrix has the direct geometric interpretation given in Theorem 3.2.

For a finite set $\mathcal{V}$ of vertices, one may wonder which triangulation yields the most stable CPWL model in the L_2 sense (*i.e.*, the smallest Riesz ratio). The Delaunay triangulation is known to be optimal in several ways (*e.g.*, in the plane it maximizes the minimum angle of all the angles of the triangles in the triangulation), but it does not necessarily give the smallest Riesz ratio. When the Delaunay triangulation is not unique, the choice can be guided by the related Riesz ratio, as detailed in Figure 3.4.

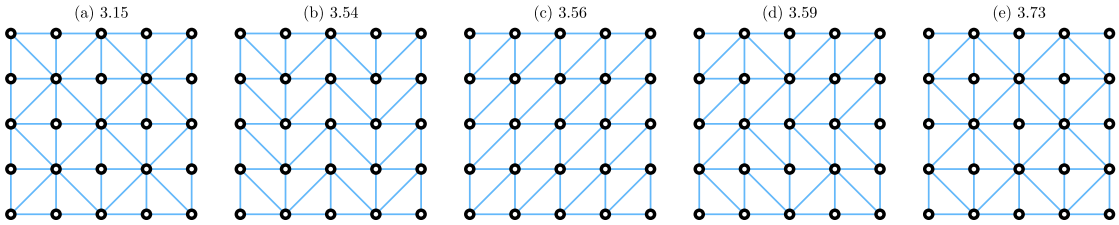

Figure 3.4: Riesz ratio of the local parameterization of CPWL functions on various Delaunay triangulations for the same set of vertices. The results stem from Theorem 3.2 and were obtained numerically. In a) and e), the pattern is similar but the Riesz ratios differ significantly. This comes mainly from the behavior on the border since the smallest star has two simplices for a) while it has a single one for e).

3.5 Exact Riesz Bounds for Linear Box Splines

Throughout this section, we assume that the vertices coincide with the sites of a lattice Λ. This is relevant in some applications, such as image processing [203], but can also be taken advantage of in low-dimensional learning problems [168]. While a uniform grid constrains the model and thereby reduces its expressivity, it significantly improves the computational performance. In this setting, the parameterization of CPWL functions involves shifts of a single linear box spline B for the hat basis functions. The generated space $\mathrm{CPWL}(\mathcal{S}) = \{\sum_{\mathbf{k}\in\Lambda} c_{\mathbf{k}} B(\cdot - \mathbf{k}) \colon c_{\mathbf{k}} \in \mathbb{C}\}$ is now integer-shift-invariant and lends itself to the tools of Fourier analysis [123]. Note that, for any $\mathbf{x} \in \mathbb{R}^d$, the sum $\sum_{\mathbf{k}\in\Lambda} c_{\mathbf{k}} B(\mathbf{x} - \mathbf{k})$ has at most $(d+1)$ nonzero terms. This follows from the short support of linear box splines.

3.5.1 Linear Box Splines

Box splines of any degree have been extensively studied. We refer the reader to the book by de Boor *et al.* [163] for a general theory and a more comprehensive account. In this chapter, we shall concentrate solely on linear box splines, which will in return allow us to derive exact Riesz bounds, which is one of our contributions.

Consider a matrix $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1 \cdots \boldsymbol{\xi}_d] \in \mathbb{R}^{d\times d}$, where $(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d)$ is a collection of linearly independent vectors of $\mathbb{R}^d$. The matrix $\boldsymbol{\Xi}$ generates a lattice of $\mathbb{R}^d$ whose sites are $\boldsymbol{\Xi}\mathbb{Z}^d = \{\boldsymbol{\Xi}\mathbf{k} \colon \mathbf{k} \in \mathbb{Z}^d\}$. Moreover, let $\boldsymbol{\Xi}_{d+1} = [\boldsymbol{\xi}_1 \cdots \boldsymbol{\xi}_{d+1}] \in \mathbb{R}^{d\times(d+1)}$ with $\boldsymbol{\xi}_{d+1} = \sum_{k=1}^{d} \boldsymbol{\xi}_k$. The linear box spline $B_{\boldsymbol{\Xi}_{d+1}} \colon \mathbb{R}^d \to \mathbb{R}$ generated by the collection of vectors $(\boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_{d+1})$ can be defined via its Fourier transform

$$\widehat{B}_{\boldsymbol{\Xi}_{d+1}}(\boldsymbol{\omega}) = |\det \boldsymbol{\Xi}| \prod_{k=1}^{d+1} \frac{1 - \mathrm{e}^{-\mathrm{i}\boldsymbol{\xi}_k^T \boldsymbol{\omega}}}{\mathrm{i}\boldsymbol{\xi}_k^T \boldsymbol{\omega}}. \tag{3.49}$$

The normalization factor $|\det \boldsymbol{\Xi}|$ ensures consistency with our definition of a hat function, but is often not included in the literature. The fact that $B_{\boldsymbol{\Xi}_{d+1}}$ is a CPWL function is made explicit with Proposition 3.3.

Proposition 3.3.

$$B_{\mathbf{\Xi}_{d+1}}(\mathbf{x}) = \sum_{\boldsymbol{\epsilon}\in\{0,1\}^{d+1}} (-1)^{|\boldsymbol{\epsilon}|} \min\left(\mathbf{\Xi}^{-1}(\mathbf{x} - \mathbf{\Xi}_{d+1}\boldsymbol{\epsilon})\right)_+, \tag{3.50}$$

where $|\boldsymbol{\epsilon}| = \sum_{k=1}^{d+1} \epsilon_k$.

Proof. The product in (3.49) can be expanded as

$$\widehat{B}_{\mathbf{\Xi}_{d+1}}(\boldsymbol{\omega}) = |\det \mathbf{\Xi}| \left(\prod_{k=1}^{d+1} \frac{1}{\mathrm{i}\boldsymbol{\xi}_k^T\boldsymbol{\omega}}\right) \times \sum_{\boldsymbol{\epsilon}\in\{0,1\}^{d+1}} (-1)^{|\boldsymbol{\epsilon}|} \mathrm{e}^{-\mathrm{i}(\mathbf{\Xi}_{d+1}\boldsymbol{\epsilon})^T\boldsymbol{\omega}}. \tag{3.51}$$

By invoking Lemma 3.7 (Appendix 3.9) and making use of the general Fourier-stretch theorem, we get that

$$\min(\mathbf{\Xi}^{-1}\mathbf{x})_+ \overset{\mathcal{F}}{\mapsto} |\det \mathbf{\Xi}| \prod_{k=1}^{d+1} \left(\frac{1}{\mathrm{i}\boldsymbol{\xi}_k^T\boldsymbol{\omega}} + \pi\delta(\boldsymbol{\xi}_k^T\boldsymbol{\omega})\right), \tag{3.52}$$

where $\mathbf{x}$ is the space variable, $\boldsymbol{\omega}$ is the pulsation variable and δ is the Dirac distribution. Knowing that $(1 - \mathrm{e}^{-\mathrm{i}\boldsymbol{\xi}_k^T\boldsymbol{\omega}})\delta(\boldsymbol{\xi}_k^T\boldsymbol{\omega}) = 0$, we observe that (3.49) has the equivalent form

$$\widehat{B}_{\mathbf{\Xi}_{d+1}}(\boldsymbol{\omega}) = |\det \mathbf{\Xi}| \left(\prod_{k=1}^{d+1} \frac{1}{\mathrm{i}\boldsymbol{\xi}_k^T\boldsymbol{\omega}} + \pi\delta(\boldsymbol{\xi}_k^T\boldsymbol{\omega})\right) \sum_{\boldsymbol{\epsilon}\in\{0,1\}^{d+1}} (-1)^{|\boldsymbol{\epsilon}|} \mathrm{e}^{-\mathrm{i}\boldsymbol{\epsilon}^T\mathbf{\Xi}_{d+1}^T\boldsymbol{\omega}}. \tag{3.53}$$

We conclude by taking the inverse Fourier transform on both sides of (3.53). □

Proposition 3.3 is illustrated in dimension $d = 2$ in Figure 3.5. This expansion gives a way to prove that the GHH model [171] can represent any linear box spline. In addition, it provides a relation between the local and nonlocal representations, with explicit generalized hinging hyperplanes (namely, the shifts of a single generalized hinging hyperplane). Our result is close to [204], where a similar formula is proven for three directional box splines of any degree, but only in dimension $d = 2$. In any dimension and for box splines of any degree, a comparable decomposition is given in [205]. Nonetheless, when applied to linear box splines, the expansion in [205] is made of discontinuous Green functions and contains more terms.

The key properties of the linear box spline $B_{\mathbf{\Xi}_{d+1}}$ are

- continuity and piecewise linearity (obvious from Proposition 3.3 since $\mathbf{x} \mapsto \min(\mathbf{x})_+$ is CPWL);
- compact support;

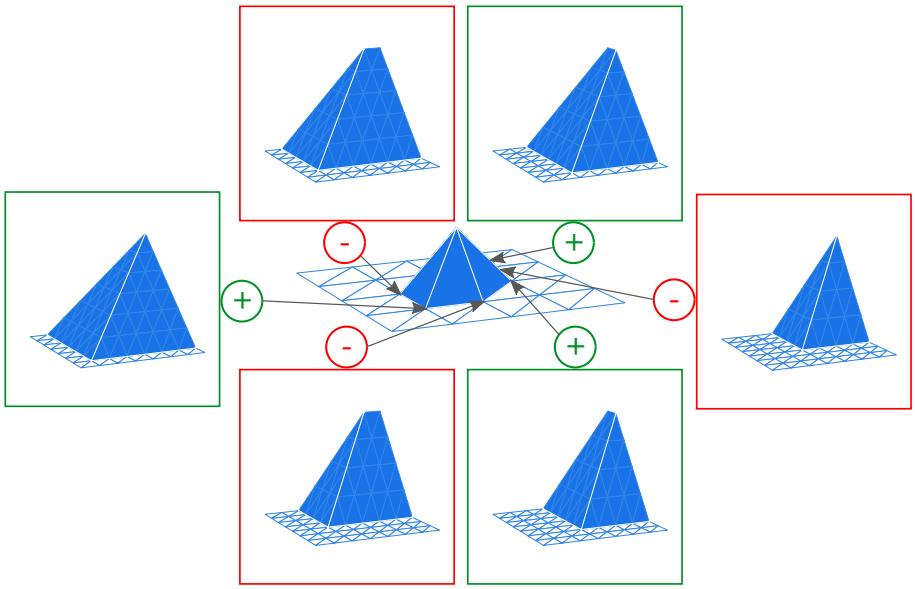

Figure 3.5: Linear decomposition of the 2D linear box spline with translates of the nonlocal function $(x, y) \mapsto \min(x, y)_+$. The six basis functions are all translates of $(x, y) \mapsto \min(x, y)_+$. They are only represented here over a finite-size grid, but they are not compactly supported. The location of the corner of each hinge in the central scene is given by the gray arrow.

- approximation power of order 2 [163], which means that the reconstruction error of sufficiently smooth decaying functions decreases with the square of the grid size.

3.5.2 Derivation of the Exact Riesz-Basis Bounds

Prior to giving the general Riesz-basis bounds of linear box splines (Theorem 3.3), we show an invariance result: the Riesz ratio associated to linear box splines does not depend on the directions $(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d)$.

Proposition 3.4. *Let $(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d)$ be a family of linearly independent vectors of $\mathbb{R}^d$, $\boldsymbol{\xi}_{d+1} = \sum_{k=1}^{d} \boldsymbol{\xi}_k$, $\boldsymbol{\Xi}_{d+1} = [\boldsymbol{\xi}_1 \cdots \boldsymbol{\xi}_{d+1}]$ and $\mathbf{U} \in \mathbb{R}^{d \times d}$ an invertible matrix. Then, the Riesz bounds of $B_{\mathbf{U}\boldsymbol{\Xi}_{d+1}}$ are the ones of $B_{\boldsymbol{\Xi}_{d+1}}$ scaled by $\sqrt{|\det(\mathbf{U})|}$.*

Proof. From Proposition 3.3, we infer that $B_{\mathbf{U}\boldsymbol{\Xi}_{d+1}} = B_{\boldsymbol{\Xi}_{d+1}} \circ \mathbf{U}^{-1}$. (The generalized version of this relation $M_{\mathbf{U}\boldsymbol{\Xi}} = \left|\det \mathbf{U}^{-1}\right| M_{\boldsymbol{\Xi}} \circ \mathbf{U}^{-1}$ is given in [163] and holds true for unnormalized box splines of any degree.) In addition, for any $f \in L_2(\mathbb{R}^d)$, a change of variable allows one to show that $\|f \circ \mathbf{U}^{-1}\|_{L_2}^2 = |\det(\mathbf{U})| \, \|f\|_{L_2}^2$, which means that, for

$c \in \ell_2(\mathbb{Z})$, it holds that

$$\| \sum_{\mathbf{k}\in\mathbb{Z}^d} c_{\mathbf{k}} B_{\mathbf{U}\boldsymbol{\Xi}_{d+1}}(\cdot - \mathbf{U}\boldsymbol{\Xi}\mathbf{k})\|_{L_2}^2 = \| \sum_{\mathbf{k}\in\mathbb{Z}^d} c_{\mathbf{k}} B_{\boldsymbol{\Xi}_{d+1}} \circ \mathbf{U}^{-1}(\cdot - \mathbf{U}\boldsymbol{\Xi}\mathbf{k})\|_{L_2}^2 \tag{3.54}$$

$$= \| \sum_{\mathbf{k}\in\mathbb{Z}^d} c_{\mathbf{k}} B_{\boldsymbol{\Xi}_{d+1}}(\mathbf{U}^{-1}\cdot - \boldsymbol{\Xi}\mathbf{k})\|_{L_2}^2 = \|(\sum_{\mathbf{k}\in\mathbb{Z}^d} c_{\mathbf{k}} B_{\boldsymbol{\Xi}_{d+1}}(\cdot - \boldsymbol{\Xi}\mathbf{k})) \circ \mathbf{U}^{-1}\|_{L_2}^2 \tag{3.55}$$

$$= |\det(\mathbf{U})| \, \| \sum_{\mathbf{k}\in\mathbb{Z}^d} c_{\mathbf{k}} B_{\boldsymbol{\Xi}_{d+1}}(\cdot - \boldsymbol{\Xi}\mathbf{k})\|_{L_2}^2, \tag{3.56}$$

which allows us to conclude. □

The fact that $\{B_{\boldsymbol{\Xi}_{d+1}}(\cdot - \boldsymbol{\Xi}\mathbf{k})\colon \mathbf{k} \in \mathbb{Z}^d\}$ forms a Riesz basis can be inferred directly from established criterion in the literature [176, 177, 206, 207]. Here, we go further and provide the upper and lower Riesz bounds in any number of dimensions (Theorem 3.3), which is a new result to the best of our knowledge.

Theorem 3.3. *Let $(\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_d)$ be a basis of $\mathbb{R}^d$, $\boldsymbol{\xi}_{d+1} = \sum_{k=1}^d \boldsymbol{\xi}_k$, and $\boldsymbol{\Xi}_{d+1} = [\boldsymbol{\xi}_1 \cdots \boldsymbol{\xi}_{d+1}]$. Then, the collection $\{B_{\boldsymbol{\Xi}_{d+1}}(\cdot - \boldsymbol{\Xi}\mathbf{k})\colon \mathbf{k} \in \mathbb{Z}^d\}$ of linear box splines forms a Riesz basis with Riesz bounds*

$$A = \sqrt{\frac{|\det \boldsymbol{\Xi}|}{(d+2)}} \quad \textit{and} \quad B = \sqrt{|\det \boldsymbol{\Xi}|}. \tag{3.57}$$

The associated Riesz ratio is $\sqrt{d+2}$.

Proof. Following Proposition 3.4, we focus solely on the Cartesian lattice $\mathbb{Z}^d$ and denote the corresponding linear box spline by $D = B_{\boldsymbol{\Xi}_{d+1}}$, where $\boldsymbol{\Xi}_d$ is the identity matrix of $\mathbb{R}^d$. In this case, it is known that the simplices form a Kuhn/Freudenthal triangulation [164]. The vertices of any simplex $s = \mathrm{conv}(\mathbf{v}_0, \dots, \mathbf{v}_d)$ of the triangulation take the form

$$\mathbf{v}_k = \mathbf{v}_0 + \sum_{p=1}^{k} \mathbf{e}_{\boldsymbol{\sigma}(p)}, \tag{3.58}$$

where $\boldsymbol{\sigma}$ is a permutation of the set $\{0, \dots, d\}$, $k = 1, \dots, d$ and $\mathbf{e}_i$ denotes the ith element of the canonical basis of $\mathbb{R}^d$. Correspondingly, for $p = 0, \dots, d$, there is a unique vertex $\mathbf{v}_{k_p}$ in each simplex such that $\left|\mathbf{v}_{k_p}\right| \equiv p \mod (d+1)$, where for any $\mathbf{x} = (x_1, \dots, x_{d+1}) \in \mathbb{R}^{d+1}$, we use the notation $|\mathbf{x}| = \sum_{k=1}^{d+1} x_k$. Indeed, (3.58) yields that $|\mathbf{v}_k| = |\mathbf{v}_0| + k$. Then, we use the Fourier characterization (3.12) to find the Riesz bounds. Specifically, we have to find the essential extrema of

$$\widehat{g}\colon \boldsymbol{\omega} \mapsto \sum_{\mathbf{k}\in\mathbb{Z}^d} \langle D, D(\cdot - \mathbf{k})\rangle \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}^T\mathbf{k}}. \tag{3.59}$$

Since the basis function D is compactly supported, the sum is finite and the function $\widehat{g}\colon \mathbb{R}^d \to \mathbb{C}$ is continuous and 2π-periodic with respect to each coordinate. We can

therefore simply look for the maximum and the minimum of $\widehat{g}$ in the hypercube $[0, 2\pi)^d$. We apply the triangular inequality, use the nonnegativity of D and invoke the partition of unity of the linear box spline to conclude that the maximum of $\widehat{g}$ is attained for $\boldsymbol{\omega} = \mathbf{0}$. With (3.25) we find the upper bound $B^2 = \frac{\mathrm{Vol}(\mathrm{St}(D))}{d+1}$, where $\mathrm{St}(D)$ is the support of D or, equivalently, the star of the vertex located at $\mathbf{1}$. Now, for the minimum, we have $\min_{\boldsymbol{\omega}\in[0,2\pi]^d} \widehat{g}(\boldsymbol{\omega}) \leq \widehat{g}(\boldsymbol{\omega}^*)$ for $\boldsymbol{\omega}_0 = (-\frac{2\pi}{d+1}\mathbf{1})$ and get

$$\begin{aligned}\sum_{\mathbf{k}\in\mathbb{Z}^d} \langle D, D(\cdot-\mathbf{k})\rangle \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega_0}^T\mathbf{k}} &= \langle D, \sum_{\mathbf{k}\in\mathbb{Z}^d} \zeta_{d+1}^{|\mathbf{k}|} D(\cdot-\mathbf{k})\rangle \\ &= \sum_{s\in\mathrm{St}(D)} \langle D, \sum_{\mathbf{k}\in\mathbb{Z}^d} \zeta_{d+1}^{|\mathbf{k}|} D(\cdot-\mathbf{k})\rangle_s \\ &= \sum_{s\in\mathrm{St}(D)} \langle D, \sum_{(\mathbf{k}-\mathbf{1})\in\mathbb{Z}^d\cap s} \zeta_{d+1}^{|\mathbf{k}|} D(\cdot-\mathbf{k})\rangle_s, \end{aligned} \tag{3.60}$$

where $\langle f, h\rangle_s = \int_{\mathbf{x}\in s} \overline{f(\mathbf{x})} h(\mathbf{x})\mathrm{d}\mathbf{x}$. On one hand, we apply Lemma 3.3 and use the inherent structure of the Kuhn triangulation displayed in (3.58) to deduce that

$$\begin{aligned}\widehat{g}(\boldsymbol{\omega}^*) &= \sum_{s\in\mathrm{St}(D)} \frac{1}{(d+1)(d+2)} \mathrm{Vol}(s) \begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix} \begin{bmatrix}2 & 1 & \cdots & 1\\ 1 & \ddots & \ddots & \vdots\\ \vdots & \ddots & \ddots & 1\\ 1 & \cdots & 1 & 2\end{bmatrix} \begin{bmatrix}1\\ \zeta_{d+1}\\ \vdots\\ \zeta_{d+1}^d\end{bmatrix} \\ &= \frac{\mathrm{Vol}(\mathrm{St}(D))}{(d+1)(d+2)}. \end{aligned} \tag{3.61}$$

On the other hand, Theorem 3.1 implies that $\min_{\boldsymbol{\omega}\in[0,2\pi]^d} \widehat{g}(\boldsymbol{\omega}) \geq \frac{\mathrm{Vol}(\mathrm{St}(D))}{(d+1)(d+2)}$, from which we infer that $A^2 = \frac{\mathrm{Vol}(\mathrm{St}(D))}{(d+1)(d+2)}$. Moreover, we have that $\int_{\mathbb{R}^d} D(\mathbf{x})\mathrm{d}\mathbf{x} = \widehat{D}(\mathbf{0}) = \det(\mathbf{I}) = 1$ (from (3.49)) and, since the box spline D is nonnegative, $\int_{\mathbb{R}^d} D(\mathbf{x})\mathrm{d}\mathbf{x} = \|D\|_{L_1} = \frac{\mathrm{Vol}(\mathrm{St}(D))}{(d+1)}$ (from (3.25)). In short, $\mathrm{Vol}(\mathrm{St}(D)) = (d+1)$, which allows us to derive the result for the Cartesian lattice. The extension to any lattice then follows from Proposition 3.4.

□

Theorem 3.1 and 3.3 yield the same bounds for linear box splines, which shows the relevance of the bounds provided for arbitrary triangulations.

In the proof of Theorem 3.3 we showed, on one hand, that the volume of the star of a vertex in the Kuhn triangulation is $(d+1)$. On the other hand, the volume of the simplices of this triangulation can be readily computed and amounts to $\frac{1}{d!}$. We deduce that the linear box spline is made of $(d+1)!$ nonzero affine pieces.

From Theorem 3.3, the Riesz ratio of the linear box-spline parameterization is $\sqrt{d+2}$, which grows with the dimension. However, this metric only reflects extreme cases. A

good estimate of the average behavior of the parameterization is given by the mean of the function $\widehat{g}$ (defined in the proof of Theorem 3.3) over $[0, 2\pi]^d$, computed as

$$\frac{1}{(2\pi)^d} \int_{\boldsymbol{\omega}\in[0,2\pi]^d} \sum_{\mathbf{k}\in\mathbb{Z}^d} \langle D, D(\cdot - \mathbf{k})\rangle \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}^T\mathbf{k}} \mathrm{d}\boldsymbol{\omega} = \langle D, D\rangle = \|D\|_{L_2}^2 \tag{3.62}$$

$$= 2\frac{\mathrm{Vol}((\mathrm{St}(D))}{(d+2)(d+1)} = 2 \min_{\boldsymbol{\omega}\in[0,2\pi]^d} \widehat{g}(\boldsymbol{\omega}). \tag{3.63}$$

We infer that $\widehat{g}$ rarely takes values close to its upper bound, especially in high dimensions, which was observed for affine functions on a simplex in Lemma 3.4. In short, although the Riesz ratio scales badly with the dimension, most linear combinations of box splines will behave in a similar way to the lower bound and the dimension should not significantly degrade the stability of the parameterization.

3.6 Lipschitz Bounds

While the determination of the Lipschitz constant of ReLU networks is NP-hard, the situation is much more favorable for the local parameterization.

3.6.1 Lipschitz Properties of ReLU Networks

Consider a one-hidden-layer ReLU network mapping $\mathbb{R}^d$ to $\mathbb{R}$ and defined by

$$h\colon \mathbf{x} \mapsto \mathbf{u}^T \mathrm{ReLU}(\mathbf{W}^T\mathbf{x} + \mathbf{b}) = \sum_{k=1}^{m} u_k \mathrm{ReLU}(\mathbf{w}_k^T\mathbf{x} + b_k), \tag{3.64}$$

where $\mathbf{W} \in \mathbb{R}^{d\times m}$, $\mathbf{u}, \mathbf{b} \in \mathbb{R}^m$ are the parameters, and $\mathbf{w}_k$ are the columns of $\mathbf{W}$. This network parameterizes a CPWL function; hence, its Lipschitz constant is the maximal Lipschitz constant of its affine pieces. However, the knowledge of these affine pieces is not straightforward as it requires one to determine the linear regions of h. This amounts to finding the nonempty regions of the form

$$\bigcap_{k=1}^{m} \{\mathbf{x} \in \mathbb{R}^d\colon \epsilon_k(\mathbf{w}_k^T\mathbf{x} + b_k) \geq 0\}, \tag{3.65}$$

where $\epsilon_k \in \{0, 1\}$. This geometrical problem is known as "finding the chambers of a hyperplane arrangement", and a sharp upper bound on the maximal number of these regions for given (d, m) can be found in [208]. The combinatorial aspect of the problem is such that the computation of $\mathrm{Lip}_2(h)$ is NP-hard [48]. In addition, under the exponential-time hypothesis, there exists no polynomial-time approximation algorithm with ratio $\Omega(d^{1-c})$ for the computation of $\mathrm{Lip}_1(h)$ and $\mathrm{Lip}_\infty(h)$ for any constant $c > 0$ [209]. In practice, the strict control of the Lipschitz constant can be achieved by the use of fast-to-

compute but loose bounds. The most common approach relies on the sub-multiplicativity of the Lipschitz constant for compositions [80, 161, 210], which yields that

$$\mathrm{Lip}_p(h) \leq \|\mathbf{W}^T\|_{p,p}\|\mathbf{u}\|_q, \tag{3.66}$$

where the matrix norm $\|\cdot\|_{\alpha,\beta}$ is defined as

$$\|\mathbf{M}\|_{\alpha,\beta} = \sup_{\|\mathbf{x}\|_\alpha=1} \|\mathbf{M}\mathbf{x}\|_\beta. \tag{3.67}$$

Remark 3.1. *The (p,p) norm of the row matrix $\mathbf{u}^T \in \mathbb{R}^{1\times d}$ is the q vector norm $\|\mathbf{u}\|_q$, where $1/p + 1/q = 1$.*

3.6.2 Lipschitz Properties of the Local Parameterization

Lipschitz Constant

In comparison to ReLU neural networks, the local parameterization $f = \sum_{\mathbf{v}\in\mathcal{V}} c_\mathbf{v}\beta^\mathcal{S}_\mathbf{v}$ generates CPWL functions with known linear regions, namely, the simplices of $\mathcal{S}$. The Lipschitz constant of f therefore reads

$$\mathrm{Lip}_p(f) = \sup_{s\in\mathcal{S}}(\|\boldsymbol{\nabla} f|_{\mathring{s}}\|_q), \tag{3.68}$$

where $1/p + 1/q = 1$, and where $\boldsymbol{\nabla} f|_{\mathring{s}}$ is the gradient of f in the interior of the simplex $\mathring{s}$, hence, a constant vector. Note that when (3.68) is not finite, then f is not a Lipschitz function. For $d > 1$, the discrete estimate

$$\sup_{\mathbf{v},\mathbf{u}\in\mathcal{V}} \frac{|c_\mathbf{v} - c_\mathbf{u}|}{\|\mathbf{v}-\mathbf{u}\|_p} \leq \mathrm{Lip}_p(f), \tag{3.69}$$

is inadequate, as illustrated in our previous work [211, Figure 2], and is generally only a lower bound. The determination of $\mathrm{Lip}_p(f)$ requires some more involved computations. Let $s = \mathrm{conv}(\mathbf{v}_1,\ldots,\mathbf{v}_{d+1})$ be a d-simplex, and let $h\colon \mathbf{x} \mapsto \mathbf{g}_s^T\mathbf{x} + b_s$ be the affine function which satisfies $h(\mathbf{v}_k) = c_k$ for $k = 1,\ldots,d+1$. The gradient $\mathbf{g}_s$ can then be computed as

$$\mathbf{g}_s = (\Delta\mathbf{V}_s^{-T})\Delta\mathbf{c}, \tag{3.70}$$

where $\Delta\mathbf{V}_s = [\mathbf{v}_2 - \mathbf{v}_1 \quad \cdots \quad \mathbf{v}_{d+1} - \mathbf{v}_1]$ and $\Delta\mathbf{c} = (c_2 - c_1, \ldots, c_{d+1} - c_1)$. The matrix $\Delta\mathbf{V}_s$ is indeed invertible since we have assumed that $\mathrm{Vol}(s) = \frac{1}{d!}|\det(\Delta\mathbf{V}_s)| > 0$. It follows from (3.68) and (3.70) that $\mathrm{Lip}_p(f)$ can always be computed in polynomial time for finite partitions.

Lipschitz Constant of the Hat Function

We propose a geometrical interpretation of the Lipschitz constant of the hat basis function, which will then lead us to a practical Lipschitz bound in Proposition 3.5.

Lemma 3.5. *Let $\beta_{\mathbf{v}}^{\mathcal{S}}$ be the hat basis function attached to the vertex $\mathbf{v}$. For $s \in \mathrm{St}(\mathbf{v})$, let $f_{s,\mathbf{v}}$ be the facet of s opposed to $\mathbf{v}$, let $\mathbf{n}_{s,\mathbf{v}}$ be the normal vector of $f_{s,\mathbf{v}}$ pointing toward the interior of s, and let $h_{s,\mathbf{v},p}$ be the p-norm length of the height of s with apex $\mathbf{v}$. Then,*

$$\boldsymbol{\nabla}\beta_{\mathbf{v}}^{\mathcal{S}}|_{\mathring{s}} = (1/h_{s,\mathbf{v},2})\mathbf{n}_{s,\mathbf{v}}, \text{ and } \mathrm{Lip}_p(\beta_{\mathbf{v}}^{\mathcal{S}}) = \max_{s\in\mathrm{St}(\mathbf{v})}(1/h_{s,\mathbf{v},q}), \tag{3.71}$$

where $1/p + 1/q = 1$.

Proof. Let $s \in \mathrm{St}(\mathbf{v})$. To simplify the proof, we write $s = \mathrm{conv}(\mathbf{v}_1, \ldots, \mathbf{v}_{d+1})$, where $\mathbf{v}_1 = \mathbf{v}$. Following (3.70), we have that

$$\boldsymbol{\nabla}\beta_{\mathbf{v}}^{\mathcal{S}}|_{\mathring{s}} = -(\Delta\mathbf{V}_s^{-T})\mathbf{1}. \tag{3.72}$$

In addition, we have that

$$\Delta\mathbf{V}_s\mathbf{e}_i = \mathbf{v}_{i+1}, \tag{3.73}$$

where $\mathbf{e}_i$ is the ith element of the canonical basis. Finally, we obtain for any $1 \leq i < j \leq d$

$$\boldsymbol{\nabla}\beta_{\mathbf{v}}^{\mathcal{S}}|_{\mathring{s}}^{T}(\mathbf{v}_{i+1} - \mathbf{v}_{j+1}) = -\mathbf{1}^T\Delta\mathbf{V}_s^{-1}\Delta\mathbf{V}_s(\mathbf{e}_i - \mathbf{e}_j) = 0, \tag{3.74}$$

which means that $\boldsymbol{\nabla}\beta_{\mathbf{v}}^{\mathcal{S}}|_{\mathring{s}}$ is orthogonal to the facet of s opposed to $\mathbf{v}$. Let $\mathbf{r}$ be the orthogonal projection of $\mathbf{v}$ onto the facet of s opposed to $\mathbf{v}$. On the one hand, $\beta_{\mathbf{v}}^{\mathcal{S}}|_s(\mathbf{v}) = 1$ and $\beta_{\mathbf{v}}^{\mathcal{S}}|_s(\mathbf{r}) = 0$. On the other hand, $\mathbf{v} - \mathbf{r}$ is parallel to $\boldsymbol{\nabla}\beta_{\mathbf{v}}^{\mathcal{S}}|_{\mathring{s}}$, which yields

$$\boldsymbol{\nabla}\beta_{\mathbf{v}}^{\mathcal{S}}|_{\mathring{s}} = \mathbf{v} - \mathbf{r}, \tag{3.75}$$

and allows us to conclude the proof. □

A Lipschitz Bound with Finite Differences and Geometrical Descriptors

To compute the Lipschitz constant of a CPWL function expressed with the local representation, one has to compute the gradients of the function over each simplex. This requires solving a set of linear problems that are specific to both the geometry (triangulation) and the sequence of coefficients ($\mathbf{c}$), as shown in (3.70). We now propose simpler-to-compute upper and lower bounds. They require solving linear problems that depend only on the geometry and, hence, can be solved offline for a given triangulation and then used for any sequence of coefficients $\mathbf{c}$ (Proposition 3.5). The bounds highlight the role of the finite differences of the coefficients and the geometry of the simplices.

Proposition 3.5. *Consider the same setting as in Theorem 3.1. Then,*

$$\sup_{\mathbf{v},\mathbf{u}\in\mathcal{V}} \frac{|c_\mathbf{v} - c_\mathbf{u}|}{\|\mathbf{v}-\mathbf{u}\|_2} \leq \mathrm{Lip}_2\left(\sum_{\mathbf{v}\in\mathcal{V}} c_\mathbf{v}\beta^\mathcal{S}_\mathbf{v}\right) \leq \sup_{s\in\mathcal{S}} \sum_{\mathbf{v}\in s\cap\mathcal{V}} \frac{|c_\mathbf{v}-\alpha_s|}{h_{s,\mathbf{v},2}}, \tag{3.76}$$

for any $\alpha_s \in \mathbb{R}$ for $s \in \mathcal{S}$, and where $h_{s,\mathbf{v},2}$ is the height of simplex s with apex $\mathbf{v}$.

Proof. Let $f = \sum_{\mathbf{v}\in\mathcal{V}} c_\mathbf{v}\beta^\mathcal{S}_\mathbf{v}$. For any $s \in \mathcal{S}$, it holds that

$$\boldsymbol{\nabla} f|_{\mathring{s}} = \sum_{\mathbf{v}\in s\cap\mathcal{V}} c_\mathbf{v}\boldsymbol{\nabla}\beta^\mathcal{S}_\mathbf{v}|_{\mathring{s}}. \tag{3.77}$$

Hence,

$$\|\boldsymbol{\nabla} f|_{\mathring{s}}\|_2 \leq \sum_{\mathbf{v}\in s\cap\mathcal{V}} |c_\mathbf{v}| \|\boldsymbol{\nabla}\beta^\mathcal{S}_\mathbf{v}|_{\mathring{s}}\|_2 = \sum_{\mathbf{v}\in s\cap\mathcal{V}} \frac{|c_\mathbf{v}|}{h_{s,\mathbf{v},2}}. \tag{3.78}$$

We now note that

$$\boldsymbol{\nabla}\left(\sum_{\mathbf{v}\in s\cap\mathcal{V}} \beta^\mathcal{S}_\mathbf{v}\right)|_{\mathring{s}} = \boldsymbol{\nabla}(1)|_{\mathring{s}} = \mathbf{0}. \tag{3.79}$$

Hence,

$$\boldsymbol{\nabla} f|_{\mathring{s}} = \sum_{\mathbf{v}\in s\cap\mathcal{V}} c_\mathbf{v}\boldsymbol{\nabla}\beta^\mathcal{S}_\mathbf{v}|_{\mathring{s}} = \sum_{\mathbf{v}\in s\cap\mathcal{V}} (c_\mathbf{v}-\alpha_s)\boldsymbol{\nabla}\beta^\mathcal{S}_\mathbf{v}|_{\mathring{s}}, \tag{3.80}$$

for any $\alpha_s \in \mathbb{R}$, which allows us to conclude. □

The bound given in Proposition 3.5 requires one to choose a reference value α_s for each simplex s against which to compare the value of the function at the vertices. In the case where $h_{s,\mathbf{v},2}$ does not depend on $\mathbf{v}$, then the choice of α_s as the median of $\{c_\mathbf{v}\}$ for $\mathbf{v} \in s \cap \mathcal{V}$ leads to the tightest bound. More generally, a suitable choice is to pick $\alpha_s \in [\min_{\mathbf{v}\in s\cap\mathcal{V}} c_\mathbf{v}, \max_{\mathbf{v}\in s\cap\mathcal{V}}, c_\mathbf{v}]$, which ensures an upper bound of 0 for constant functions. Note that this choice yields an exact bound when $d = 1$, but this is not necessarily true for $d > 1$.

3.7 Conclusion

We have provided a measure of the stability of the parameterization of CPWL functions with hat basis functions on any triangulation. First, we have estimated the $\ell_2 \to L_2$ condition number of the parameterization in full generality. We have found that it is mainly determined by the relative volume of the star of the vertices of the triangulation, namely, the relative size of the support of the hat functions. This result is in accordance with the literature on finite-element methods, which however only focus on the case of finitely many vertices. When the vertices lie at the sites of a lattice, we parameterize the CPWL functions with linear box splines. We have provided a formula to relate these

local basis functions to nonlocal ReLU-like functions. In this uniform setting, we have proved that the exact condition number only depends on the dimension d and is $\sqrt{d+2}$. Although it increases with the dimension, we noticed that, from a stochastic point of view, the dimension might only rarely affect the stability of the local representation. Eventually, we have shown that the Lipschitz constant of a CPWL function can be easily determined when its local parameterization is known. This highly contrasts with CPWL functions defined by ReLU neural networks.

3.8 Appendix – Interpolation Condition Number of the Non-local Parameterization

Consider the nonlocal parameterization of one-dimensional CPWL functions with knots $v_1 < \cdots < v_K \in \mathbb{R}$

$$T\{\boldsymbol{\theta}\}(x) = \theta_1 + \theta_2(x - v_1) + \sum_{k=2}^{K-1} \theta_{k+1}(x - v_k)_+. \tag{3.81}$$

Given target values $y_1, \ldots, y_K \in \mathbb{R}$, the parameters $\theta_1, \ldots, \theta_K \in \mathbb{R}$ have to satisfy for $p = 1, \ldots, K$ that

$$y_p = T\{\boldsymbol{\theta}\}(v_p) = \theta_1 + \theta_2(v_p - v_1) + \sum_{k=2}^{K-1} \theta_{k+1}(v_p - v_k)_+, \tag{3.82}$$

which, assuming a constant step size $h = (v_{k+1} - v_k)$, further simplifies in $y_p = \theta_1 + h\sum_{k=1}^{p-1} \theta_{k+1}(p-k)$. This yields the matrix equation

$$\boldsymbol{\theta} = \frac{1}{h}\begin{bmatrix} 1/h & 0 & \cdots & \cdots & 0 \\ 1/h & 1 & 0 & \cdots & 0 \\ \vdots & 2 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 1/h & (K-1) & \cdots & 2 & 1 \end{bmatrix}^{-1} \quad \mathbf{y} = \mathbf{M}^{-1}\mathbf{y}. \tag{3.83}$$

To give a lower bound to the condition number of the problem, we remark that $\mathbf{M}\mathbf{e}_2 = h\sum_{k=1}^{K-1} k\mathbf{e}_{k+1}$ and $\mathbf{M}\mathbf{e}_K = h\mathbf{e}_K$, where $\mathbf{e}_k$ are the canonical vectors of $\mathbb{R}^d$. We infer that $\|\mathbf{M}^{-1}\sum_{k=1}^{K-1} k\mathbf{e}_k\|_2 / \|\sum_{k=1}^{K-1} k\mathbf{e}_k)\|_2 = (K(K-1)(2K-1)/6)^{-1/2}/h$ and that $\frac{\|\mathbf{M}^{-1}\mathbf{e}_K\|_2}{\|\mathbf{e}_K\|_2} = 1/h$. It implies that the ℓ_2 condition number r of the problem satisfies

$$r = \max_{\mathbf{a},\mathbf{b}\in\mathbb{R}^d\setminus\{\mathbf{0}\}} \left\{ \frac{\left\|\mathbf{M}^{-1}\mathbf{a}\right\|}{\|\mathbf{a}\|} \frac{\|\mathbf{b}\|}{\|\mathbf{M}^{-1}\mathbf{b}\|} \right\} \geq \sqrt{\frac{K(K-1)(2K-1)}{6}}. \tag{3.84}$$

3.9 Appendix – The Generalized Hinging-Hyperplane Generating Function of Linear Box Splines

The Fourier transform $\widehat{h}$ of the Heaviside function $h\colon x \mapsto \begin{cases} 1, & x \geq 0 \\ 0, & \text{otherwise,} \end{cases}$ is given by

$$\widehat{h}\colon \omega \mapsto \frac{1}{\mathrm{i}\omega} + \pi\delta(\omega). \tag{3.85}$$

From this we infer $\widehat{H}$, the Fourier transform of $H\colon \mathbf{x} \mapsto \prod_{k=1}^{d} h(x_k)$, the separable version of h in d dimensions,

$$\widehat{H}\colon \boldsymbol{\omega} \mapsto \prod_{k=1}^{d} \left(\frac{1}{\mathrm{i}\omega_k} + \pi\delta(\omega_k) \right). \tag{3.86}$$

We define a directional Heaviside wall function as $W\colon \mathbf{x} \mapsto h(x_d) \prod_{k=1}^{d-1} \delta(x_k)$ and the matrix $\mathbf{A}_d = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \\ 1 & \cdots & \cdots & 1 \end{bmatrix}$.

Lemma 3.6.

$$\forall \mathbf{x} \in \mathbb{R}^d, (H * (W \circ \mathbf{A}_d))(\mathbf{x}) = \min(\mathbf{x})_+. \tag{3.87}$$

Proof. We have that

$$(H * (W \circ \mathbf{A}_d))(\mathbf{x}) = \int_{\mathbb{R}^d} H(\mathbf{x} - \mathbf{t}) W(\mathbf{A}_d \mathbf{t}) \mathrm{d}\mathbf{t}. \tag{3.88}$$

In the sequel, we take advantage of two properties.

- We prove that $\det(\mathbf{A}_d) = d$ by induction using the Laplace/cofactor expansion along the last column.
- We observe that $\mathbf{A}_d \mathbf{1} = d\mathbf{e}_d$, which in turn implies that $\mathbf{A}_d^{-1}\mathbf{e}_d = d^{-1}\mathbf{1}$.

We now use the change of variable $\mathbf{y} = \mathbf{A}_d \mathbf{t}$. It yields that

$$\begin{aligned}
(H * (W \circ \mathbf{A}_d))(\mathbf{x}) &= (1/d) \int_{\mathbb{R}^d} H(\mathbf{x} - \mathbf{A}_d^{-1}\mathbf{y}) h(y_d) \prod_{k=1}^{d-1} \delta(y_k) \mathrm{d}\mathbf{y} \\
&= (1/d) \int_{\mathbb{R}^d} H(\mathbf{x} - y_d \mathbf{A}_d^{-1}\mathbf{e}_d) h(y_d) \prod_{k=1}^{d-1} \delta(y_k) \mathrm{d}\mathbf{y} \\
&= (1/d) \int_{\mathbb{R}^d} H(\mathbf{x} - \frac{y_d}{d}\mathbf{1}) h(y_d) \prod_{k=1}^{d-1} \delta(y_k) \mathrm{d}\mathbf{y} \\
&= (1/d) \int_{\mathbb{R}^d} h(y_d) \left(\prod_{k=1}^{d} h(x_k - y_d/d) \right) \left(\prod_{k=1}^{d-1} \delta(y_k) \right) \mathrm{d}\mathbf{y} \\
&= (1/d) \int_{\mathbb{R}} h(y_d) \left(\prod_{k=1}^{d} h(x_k - y_d/d) \right) \mathrm{d}y_d.
\end{aligned} \tag{3.89}$$

The quantity $\prod_{k=1}^{d} h(x_k - y_d/d) h(y_d)$ is nonzero when $y_d > 0$ and $y_d < dx_k$ for $k =$

$1, \ldots, d$, which is equivalent to $y_d > 0$ and $y_d < d \min(x_k)$. We can now conclude that $(H * (L \circ \mathbf{A}_d))(\mathbf{x}) = \min(\mathbf{x})_+$. □

Lemma 3.7.

$$\min(\mathbf{x})_+ \overset{\mathcal{F}}{\mapsto} \prod_{k=1}^{d+1} \left(\frac{1}{\mathrm{i}\omega_k} + \pi\delta(\omega_k) \right), \tag{3.90}$$

where $\omega_{d+1} = \sum_{k=1}^{d} \omega_k$.

Proof. From Lemma 3.6, we have that $(H * (W \circ \mathbf{A}_d))(\mathbf{x}) = \min(\mathbf{x})_+$. The function W is separable and its Fourier transform reads $\widehat{W}(\boldsymbol{\omega}) = \frac{1}{\mathrm{i}\omega_d} + \pi\delta(\omega_d)$. In addition, the general stretch theorem implies that

$$(W \circ \mathbf{A}_d) \overset{\mathcal{F}}{\mapsto} \frac{1}{d}\widehat{W}(\mathbf{A}_d^{-T}\boldsymbol{\omega}) = \frac{1}{d}\left(\frac{1}{\mathrm{i}\mathbf{e}_d^T \mathbf{A}_d^{-T}\boldsymbol{\omega}} + \pi\delta(\mathbf{e}_d^T \mathbf{A}_d^{-T}\boldsymbol{\omega}) \right). \tag{3.91}$$

Now, we use that $\mathbf{A}_d^{-1}\mathbf{e}_d = d^{-1}\mathbf{e}_d$ and the effect of a dilation on the Dirac distribution to conclude that

$$(W \circ \mathbf{A}_d) \overset{\mathcal{F}}{\mapsto} \frac{1}{d}\widehat{W}(\mathbf{A}_d^{-T}\boldsymbol{\omega}) = \left(\frac{1}{\mathrm{i}\omega_{d+1}} + \pi\delta(\omega_{d+1}) \right). \tag{3.92}$$

We reach the conclusion by using the Fourier transform of H and by transforming the convolution into a product in the Fourier domain. □

Part II Going Deeper with Stability Guarantees

The second part of this thesis is dedicated to the design of stable DNNs to build trustworthy plug-and-play (PnP) image reconstruction methods. We propose to build Lipschitz-constrained deep neural network (DNN) denoisers to obtain provably convergent and robust PnP methods. The main challenge lies in the fact that for most DNNs, the Lipschitz constant is not efficiently computable, and the existing methods to impose a hard Lipschitz constraint usually come with a significant performance drop. For these reasons, many popular PnP methods rely on relaxed or soft Lipschitz constraints, at the possible expense of trustworthiness. In this part, we do not relax the constraint and pursue the goal of restoring the performance of Lipschitz-constrained DNN denoisers.

Our first chapter serves as a transition from shallow parameterizations, as studied in Part I, to deep ones. The game changer in DNNs is the composition operation. To shed some light on its role in DNNs, we first propose a geometrical analysis in Chapter 4 and focus on the family of continuous and piecewise-linear (CPWL) DNNs, which covers many popular architectures. Beyond the role of depth and width, which have been abundantly studied, we characterize the role of generic nonlinear modules on the linear regions of DNNs and generalize some well-established results on the expressivity of ReLU NNs. In the context of this thesis, the outcome is twofold. First, a DNN produces such complicated mappings that the computation of the Lipschitz constant, which would be trivial if the set of linear regions were known, is not feasible in practice. Second, the use of expressive activation functions is expected to boost the expressivity of DNNs with few additional parameters.

To bypass the exact computation of the Lipschitz constant, we thus propose to build stable DNNs with the popular layer-wise approach. The Lipschitz constant of each layer is constrained, resulting in a hard constraint on the Lipschitz constant of the whole DNN. This strategy is justified by the sub-multiplicativity of the Lipschitz constant for the composition operation, but it may lead to overly constrained DNNs, all the more for deep NNs. Hence, one needs to craft carefully the layers of the DNN to preserve

expressivity under the constraint. To do so, we provide a theoretical study in Chapter 5 and reveal the benefits of using learnable linear spline (LLS) activations with sufficiently many breakpoints to build Lipschitz-constrained DNNs. In Chapter 6, we then aim to leverage these findings to improve the performance of trustworthy PnP methods. We begin with an overview of PnP methods and detail our theoretical motivations for using Lipschitz-constrained denoisers. We then propose an algorithm to efficiently deploy Lipschitz-constrained LLS within DNNs. Finally, we empirically show that Lipschitz-constrained LLS activations yield improvements over other competing activations in denoising, CT, and MRI image reconstruction.

4 On the Number of Regions of Continuous and Piecewise-Linear Neural Networks

The text of this chapter is adapted from the in press paper

A. Goujon, A. Etemadi and M. Unser "On the number of regions of piecewise linear neural-networks", accepted in *Journal of Computational and Applied Mathematics*, 2023.

Summary

Many feedforward neural networks (NNs) generate continuous and piecewise-linear (CPWL) mappings. Specifically, they partition the input domain into regions on which the mapping is affine. The number of these so-called linear regions offers a natural metric to characterize the expressiveness of CPWL NNs. The precise determination of this quantity is often out of reach in practice, and bounds have been proposed for specific architectures, including for ReLU and Maxout NNs. In this chapter, we generalize these bounds to NNs with arbitrary CPWL activation functions. We first provide upper and lower bounds on the maximal number of linear regions of a CPWL NN given its depth, width, and the number of linear regions of its activation functions. Our results rely on the combinatorial structure of convex partitions and confirm the distinctive role of depth which, on its own, is able to exponentially increase the number of regions. We then introduce a complementary stochastic framework to estimate the average number of linear regions produced by a CPWL NN. Under reasonable assumptions, the expected density of linear regions along any 1D path is bounded by the product of depth, width, and a measure of activation complexity (up to a scaling factor). This yields an identical role to the three sources of expressiveness: no exponential growth with depth is observed anymore.

4.1 Introduction

The ability to train deep parametric models has enabled dramatic advances in a wide variety of fields, ranging from computer vision to natural-language processing [212, 213]. Many popular deep models belong to the family of feedforward neural networks (NNs), for which the input-output mapping takes the form

$$\mathbf{x} \mapsto (\boldsymbol{\sigma}_L \circ \boldsymbol{f}_{\boldsymbol{\theta}_L} \circ \boldsymbol{\sigma}_{L-1} \circ \cdots \circ \boldsymbol{\sigma}_2 \circ \boldsymbol{f}_{\boldsymbol{\theta}_2} \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{f}_{\boldsymbol{\theta}_1})(\mathbf{x}), \tag{4.1}$$

where L is the number of layers of the NN (referred to as the depth of the NN), $\boldsymbol{f}_{\boldsymbol{\theta}_k} \colon \mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}}$ is an affine function parameterized by $\boldsymbol{\theta}_k$, and $\boldsymbol{\sigma}_k$ is a non-affine activation function. One of the most widespread activation functions in deep learning is the rectified linear unit $\mathrm{ReLU}(x) = \max(x, 0)$ [155, 156, 172]. With this choice, the mapping is a composition of continuous and piecewise-linear (CPWL) functions, which yields a map that is CPWL too [214]. Remarkably, the reverse also holds true: any CPWL function $\mathbb{R}^d \to \mathbb{R}$ can be parameterized by a ReLU NN with at most $\lceil \log_2(d+1) \rceil$ hidden layers [157]. The family of NNs generating CPWL functions (referred to as CPWL NNs in the sequel) is broad. It benefits from a large choice of effective activation functions that includes ReLU [155], leaky ReLU [172], PReLU [173], CReLU [174], Maxout [175], linear splines [215, 216], GroupSort [82], Householder [217] as well as other components such as convolutional layers, max- and average-pooling, skip connections [218], and batch normalization [219](once the model is trained). While the depth of the architecture is instrumental to overcome the curse of dimensionality [158–160], it concurrently deters our understanding of the parameterization when compared to simpler models [220].

The observation that a ReLU NN produces a CPWL function sheds light on its behavior. In effect, a ReLU NN partitions the input domain into affine regions [71, 221]. The characteristics of the regions are therefore fundamental to grasp the structure of the learned mapping and there exist different approaches to define them [222, 223]. The regions can be described as polyhedrons or union of polyhedrons, which results from the continuity and the piecewise-affine property of the mapping. In the case of ReLU NNs, it is common to define activation regions, which are sets of points that fire the same group of neurons. On each activation region, the mapping is affine and these sets are convex [224]. Unfortunately, the linear regions in deep NNs are only indirectly specified. While they can be locally described [225] their global delimitation becomes computationally less and less tractable as the dimension increases, which compromises the interpretability of deep NNs. Yet, it is entangled with their ability to overcome the curse of dimensionality.

The successive compositions inherent in deep models prevent us from attributing a specific role to each parameter. The size and the expressiveness of the function space $\mathcal{H}_{\mathcal{N}}$ generated by a given architecture $\mathcal{N}$ is consequently remotely connected to the number of trainable parameters. With their remarkable structure, CPWL NNs benefit from

another meaningful descriptor: the distribution of counts of regions of all the mappings that the architecture can produce. Two approaches have been proposed to give a better understanding of this descriptor.

- *Upper and lower bound the maximum number of regions of the CPWL mappings generated by a given architecture.* The first bounds, given in [214], showed that the maximum number of regions that can be produced by ReLU NNs increases exponentially with their depth. This revealed that deep models have the ability to generate much more complex functions than shallow ones do. The bounds for ReLU NNs have since been refined, for example in [73] and then in [226], and also extended to other NN architectures. For instance, [227] specifies bounds for the maximum number of regions of convolutional NNs (CNNs). It is shown that CNNs produce more regions per parameter than fully connected NNs do. For Maxout NNs, bounds can be derived directly from the ones on ReLU NNs [73, 214]. However, this approach usually yields loose bounds, as recently shown in [228]. The derivation of sharp bounds for Maxout NNs, as proposed in [228], requires to take into account the specificities of the Maxout unit, and it was handled via the use of tropical geometry.

  The available bounds show that the maximum number of regions in ReLU and Maxout NNs increases exponentially with their depth. It suggests that deep models have the ability to generate more complex functions than shallow ones do [73, 157, 214, 226, 228].

- *Upper bound the average number of regions of the mappings generated by ReLU and Maxout NNs.* The available bound for ReLU NNs depends on the number of neurons, regardless of whether the NN is deep or wide, and depth does not produce exponentially more regions on average [224, 229]. In other words, this behavior drastically differs from the maximum number of regions. This new perspective was then recently extended to Maxout NNs, with a similar qualitative conclusion [230].

The existing toolbox of CPWL NNs is broad and likely not complete yet, as hinted by recent works on the MaxMin or more generally GroupSort activation function, in the field of Lipschitz-constrained NNs [82, 231], or with the Piecewise Linear Unit (PWLU) [232]. Previous studies on the count of linear regions have provided insights on some specific CPWL NNs only, mostly ReLU and Maxout NNs. Their qualitative outcomes turn out to hold true for CPWL NNs in general. We intend to prove this claim with quantitative results in this chapter. We want to improve the understanding of the role of the three main ways to increase the expressiveness of CPWL NNs (Figure 4.1), namely,

- **depth**, which is the number of composed CPWL functions;
- **width**, which relates the input and output dimensions of the composed CPWL layers;

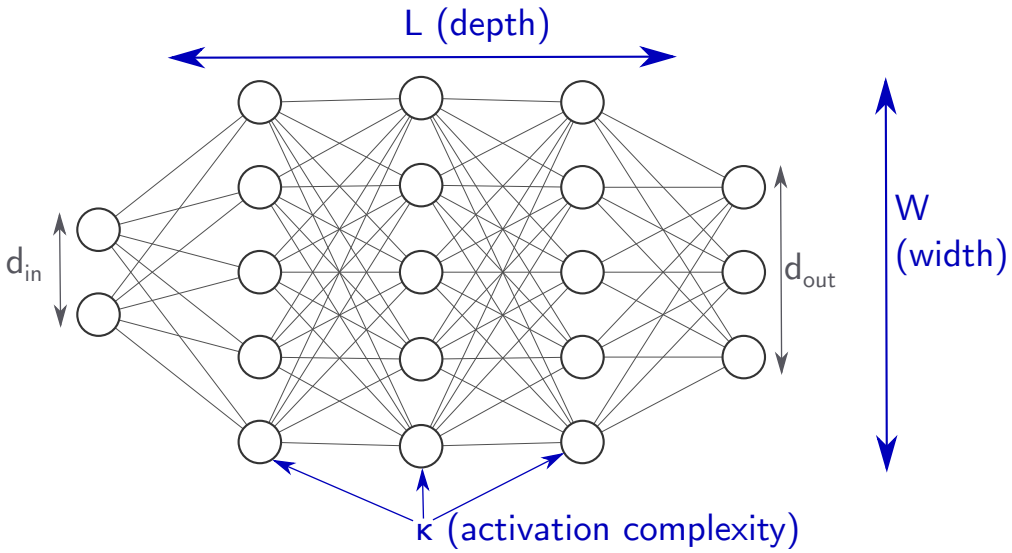

Figure 4.1: The three sources of complexity of CPWL NNs.

- **activation complexity**, the rationale there being that the expressiveness of CPWL NNs can be heightened by increasing the complexity of the composed functions. This strategy is used in Maxout, GroupSort, PWLU, and deepspline NNs for example. In the remainder of this chapter, the complexity of an activation will refer to its number of linear regions. For example, a rank-k Maxout unit has a complexity of k (see Figure 4.6 for visual examples).

Our contributions are as follows.

(i) Generalization of the notion of arrangement of hyperplanes to arrangement of convex partitions with analogous tight bounds on the number of regions.

(ii) Determination of precise bounds on the maximal number of linear convex regions generated by the primary operations of the space of CPWL functions (sum, vectorization, and composition). The compositional upper and lower bound grow exponentially with depth and polynomially with the width and the activation complexity.

(iii) Demonstration that, under reasonable assumptions, the expected number of regions along a 1D path for random CPWL NNs is at most linear with the product of the depth, the width, and the activation complexity (up to an independent factor), which yields equivalent roles to the three descriptors in terms of expressiveness.

The chapter is organized as follows: In Section 4.2, we present the relevant mathematical concepts. In Section 4.3, we bound from below and from above the maximal number of

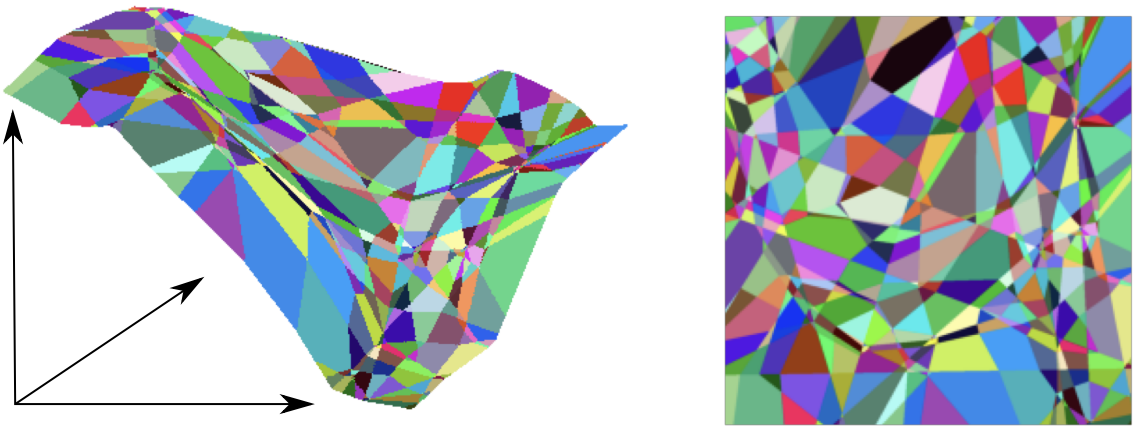

Figure 4.2: An $\mathbb{R}^2 \to \mathbb{R}$ CPWL function and its corresponding partition of the input space.

regions produced by CPWL NNs and, in Section 4.4, we present a stochastic framework to quantify the average expressiveness of CPWL NNs with random parameters.

4.2 Mathematical Preliminaries

4.2.1 CPWL Functions

Definition 4.1. *A function $\boldsymbol{f}\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ is continuous and piecewise-linear (CPWL) if it is continuous and if there exists a set $\{\boldsymbol{f}^k\colon k \in \{1,\dots,K\}\}$ of affine functions and closed subsets $(\Omega_k)_{k=1}^K$ of $\mathbb{R}^d$ with nonempty and pairwise disjoint interiors such that $\cup_{k=1}^K \Omega_k = \mathbb{R}^d$ and $\boldsymbol{f}_{|\Omega_k} = \boldsymbol{f}^k$ on Ω_k. The $\boldsymbol{f}^k$ are called the affine pieces of $\boldsymbol{f}$, and the Ω_k the corresponding projection regions.*

An example of a CPWL function and of its partition is given in Figure 4.2. The kth component of a vector-valued CPWL function $\boldsymbol{f}_\ell$, which is necessarily CPWL as well, will be denoted by $f_{\ell,k}$. The space of CPWL functions has the following remarkable properties:

- it is closed under compatible compositions;
- it is closed under compatible linear combinations;
- it is closed under compatible vectorization.

Since the function $\mathbf{x} \mapsto \max(\mathbf{x}) = \max(x_1,\dots,x_d)$ is CPWL (with d regions), the space of CPWL functions is also closed under max-pooling.

4.2.2 Regions of CPWL Functions and Convex Partitions

The term *linear region* is frequently used in an ambiguous way and may refer to different mathematical definitions. In the sequel, we shortly present some relevant definitions and discuss them in the context of CPWL NNs.

Projection Regions

We recall that a polyhedron is the intersection of finitely many half-spaces, and that a polytope is a bounded polyhedron. The subsets Ω_k in Definition 4.1 are commonly referred to as projection regions [222, 223]. The affine pieces of different projection regions are distinct and, since the overall function is continuous, the common points of two neighboring regions lie in a hyperplane. This implies that the Ω_k are polyhedrons or unions of polyhedrons. These projection regions might, however, not be connected (Figure 4.3).

Convex Regions

It is usually preferred to work with (connected) convex regions because of their simpler geometrical structure. We now precisely define convex linear regions of CPWL functions.

Definition 4.2 (Convex partitions of $\mathbb{R}^d$, adapted from [233])**.** *Let n and d be two positive integers. A convex partition of $\mathbb{R}^d$ is a collection $\Pi = (P_1, P_2, ..., P_n)$ of convex and closed subsets of $\mathbb{R}^d$ with nonempty and pairwise-disjoint interiors so that the union $\bigcup_{k=1}^{n} P_k = \mathbb{R}^d$. Each of the sets P_k is called a region of Π. Convex partitions with n regions are called n-partitions.*

Definition 4.3 (Linear convex partition)**.** *A convex partition Π of $\mathbb{R}^d$ is said to be a linear convex partition of a CPWL function $\boldsymbol{f}\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ if $\boldsymbol{f}$ is affine on each region of Π.*

The existence of a linear convex partition is guaranteed for any CPWL function but not its unicity. This motivates Definition 4.4, which gives a precise meaning to the *number of convex linear regions* for CPWL functions.

Definition 4.4 (Number of convex linear regions)**.** *The number $\kappa_{\boldsymbol{f}}$ of convex linear regions of $\boldsymbol{f}$ is defined as the minimal cardinality of all linear convex partitions of $\boldsymbol{f}$.*

A special instance of the linear convex regions for scalar-valued CPWL functions are the uniquely-ordered regions. Each of these regions has the same ordering of the values of the affine pieces f^k of f in all its points [223]. Uniquely-ordered regions are used to build the lattice representation of a CPWL function [222] and are tightly connected to the GroupSort activation function [82].

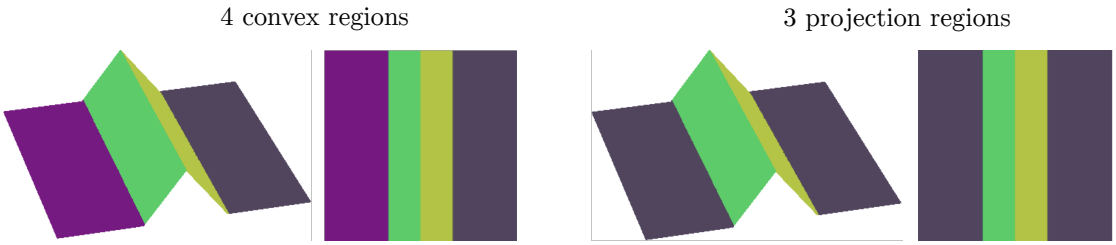

Figure 4.3: Convex and projection regions of the CPWL function $(x, y) \mapsto \mathrm{ReLU}(\min(x + 1, -x + 1))$.

Projection vs Convex Linear Regions

In the remainder of this chapter we shall keep in mind the following connections between projection and convex linear regions.

- Projection regions can always be partitioned into convex regions so that any upper bound on the number of convex regions also applies to the number of projection regions. Conversely, the number of convex regions can also be upper bounded by the number of projection regions (Proposition 4.1).

- The majority of commonly used parameterizations have typically convex projection regions. The local parameterization with hat basis functions produces simplicial linear splines whose natural regions are simplices [220] and, therefore, are convex. Other known linear expansions, such as the generalized hinging-hyperplanes model [171], use nonlocal CPWL basis functions that partition the input domain into convex regions. The generated function will produce projection regions that are convex for all sets of parameters except for some specific values that are usually encountered with zero probability in a learning framework. The convex regions are also naturally adapted to compositional models such as ReLU and Maxout NNs as explained it [229].

Proposition 4.1. *Let $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^{d'}$ be a CPWL function with ρ projection regions. The number κ of linear convex regions of $\boldsymbol{f}$ is no larger than the number of convex regions formed by the arrangement of $\rho(\rho-1)/2$ hyperplanes*

$$\rho \leq \kappa \leq \begin{cases} 2^{\rho(\rho-1)/2}, & \rho(\rho-1)/2 \leq d \\ \sum_{k=0}^{d} \binom{\rho(\rho-1)/2}{k}, & \textit{otherwise.} \end{cases} \tag{4.2}$$

The proof of Proposition 4.1 is given in 4.6.

Useful Properties of Convex Partitions

We now give a series of lemmas on convex partitions that are used in the proofs of Section 4.3. The proofs are given in 4.6.

For convenience, we extend the definition of convex partitions of $\mathbb{R}^d$ to convex partitions of affine subspaces of $\mathbb{R}^d$. In particular, a convex partition of an affine subspace E of $\mathbb{R}^d$ consists of convex and closed subsets of E of dimension $\dim(E)$ whose pairwise intersection is of dimension smaller than $\dim(E)$ and whose union is E.

Lemma 4.1 (Projection of a convex partition)**.** *Let E be an affine subspace of $\mathbb{R}^d$ and Π an n-partition of $\mathbb{R}^d$. Then, there exists a convex partition Π_E of E in $\mathbb{R}^d$ with no more than n regions such that, for $P_E \in \Pi_E$, there is $P \in \Pi$ with $P_E \subset P$.*

Lemma 4.2 (Preimage of a convex partition under affine maps)**.** *Let $\boldsymbol{f}\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ be an affine function and Π be an N-partition of the affine space $\boldsymbol{f}(\mathbb{R}^d)$ in $\mathbb{R}^{d'}$. Then, $\boldsymbol{f}^{-1}(\Pi) = \{\boldsymbol{f}^{-1}(P)\colon P \in \Pi\}$ is an N-partition of $\mathbb{R}^d$.*

Lemma 4.3. *Let $(\boldsymbol{f}_\ell)_{\ell\in[L]}$ be a collection of affine functions with $\boldsymbol{f}_\ell : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell+1}}$ .Then,*

$$\dim((\boldsymbol{f}_L \circ \cdots \circ \boldsymbol{f}_1)(\mathbb{R}^{d_1})) \leq \min(d_1, \ldots, d_{L+1}). \tag{4.3}$$

4.2.3 Arrangement of Convex Partitions

The known results on the number of convex regions of ReLU NNs are built upon the theory of hyperplane arrangements. In combinatorial geometry, an arrangement of hyperplanes refers to a set of hyperplanes. It is known that the number of connected regions formed by an arrangement of N hyperplanes in $\mathbb{R}^d$ is at most $\sum_{k=0}^{\min(d,N)} \binom{N}{k}$ [208]. This bound is reached when the hyperplanes are in general position: any collection of k of them intersect in a $(d-k)$-dimensional plane for $1 \leq k \leq d$ and have an empty intersection for $k > d$. Although this positioning seems very specific, it is qualified as "general" because it almost surely happens when the hyperplanes are randomly generated (with a "reasonable" notion of randomness). When it comes to the study of generic CPWL NNs, the concept of arrangement of hyperplanes lacks precision since only a small fraction of all convex partitions can be seen as arrangement of hyperplanes. We thus introduce the notion of arrangement of convex partitions (Definition 4.5 and Figure 4.4) as a generalization, which will prove to be necessary to find the precise bounds given in Section 4.3. Note that, in the case of an arrangement of N hyperplanes, our terminology differs. Instead of considering the hyperplanes, we rather consider the N 2-partitions they form, which consist of pairs of closed half-spaces separated by the hyperplanes.

Definition 4.5 (Arrangement of convex partitions)**.** *Let $(\Pi_k)_{k\in[N]}$ be a collection of N convex partitions, with $\Pi_k = (P_1^k, \ldots, P_{n_k}^k)$, for $k \in [N]$. The arrangement $\mathcal{A}(\Pi_1, \ldots, \Pi_N)$ of these partitions is the convex partition whose regions are the $A_{m_1,\ldots,m_N}$ that have*

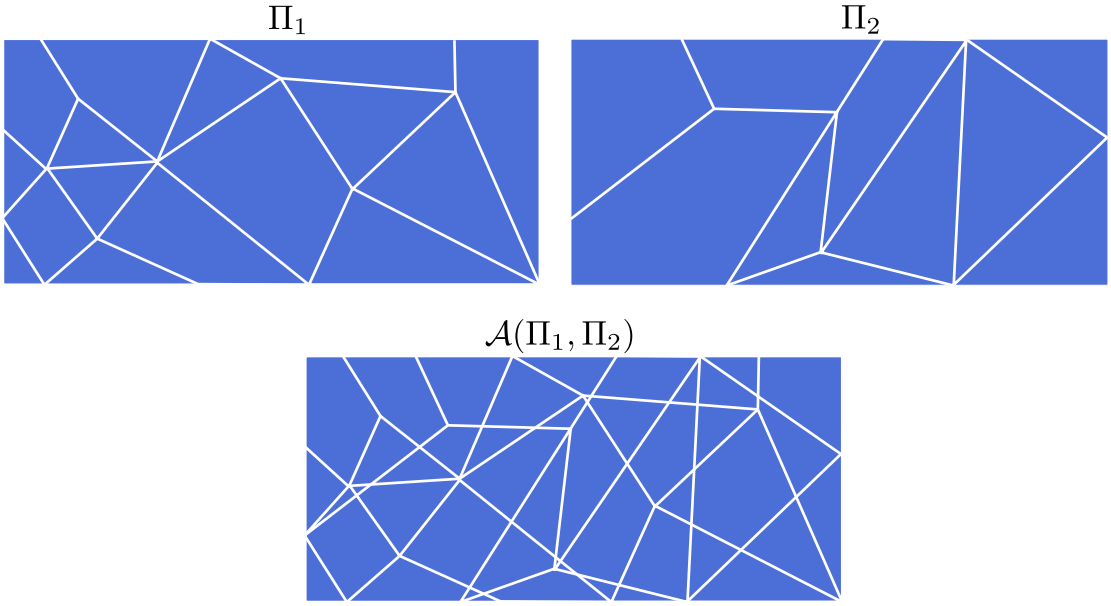

Figure 4.4: Arrangement of two convex partitions of $\mathbb{R}^2$.

nonempty interiors, where

$$A_{m_1,\dots,m_N} = \bigcap_{k=1}^{N} P_{m_k}^k,$$

for $(m_1, \dots, m_N) \in \{1, \dots, n_1\} \times \cdots \times \{1, \dots, n_N\}$.

4.3 Maximum Number of Regions Produced by CPWL NNs

In this section, we characterize the largest number of regions that can be generated by simple operations with CPWL functions, including sums, vectorizations, and compositions. In particular, we strictly generalize the known upper and lower bounds on the number of regions of ReLU NNs [73] and Maxout NNs [228] to NNs activated by generic CPWL activation functions.

4.3.1 Upper Bound on the Number of Regions of Arrangements

Operations with CPWL functions imply arrangement of convex partitions, either explicitly, for sums and vectorizations, or implicitly, for compositions. It is straightforward to see that an arrangement $\mathcal{A}(\Pi_1, \dots, \Pi_N)$ of N convex partitions $\Pi_1, \dots, \Pi_N$ of $\mathbb{R}^d$ with $n_1, \dots, n_N$ regions cannot yield more than $n_1 n_2 \cdots n_N$ regions. This naive bound is a polynomial of degree N in $n_1, \dots, n_N$. In dimension $d = 1$ one can, however, check that the bound is not sharp: the number of regions is no more than $1 + (n_1 - 1) + \cdots + (n_N - 1)$. More generally,

the number of regions of the arrangement is bounded by a polynomial in the cardinality of the partitions $n_1, \dots, n_N$ of degree $\min(d, N)$ (Theorem 4.1), which highlights the role played by the dimension of the ambient space.

Theorem 4.1 (Arrangements' upper bound)**.** *The maximum cardinality $\beta^d(n_1, \dots, n_N)$ of the arrangement $\mathcal{A}(\Pi_1, \dots, \Pi_N)$ of N convex partitions $\Pi_1, \dots, \Pi_N$ of $\mathbb{R}^d$ with cardinality $n_1, \dots, n_N$ is a polynomial in $n_1, \dots, n_N$ of degree $\min(d, N)$. It is given by*

$$\beta^d(n_1, \dots, n_N) = 1 + \sum_{k=1}^{\min(d,N)} \sum_{1 \le \ell_1 < \dots < \ell_k \le N} \prod_{q=1}^{k} (n_{\ell_q} - 1). \tag{4.4}$$

Moreover, this bound satisfies

$$\begin{array}{ll} \beta^d(n_1, \dots, n_N) & = \prod_{k=1}^{N} n_k, \qquad\qquad\qquad\qquad\qquad\quad \text{if } N \le d \\ \beta^d(n_1, \dots, n_N) & \le \left(1 + \sum_{k=1}^{N} (n_k - 1)\right)^d \le \left(\sum_{k=1}^{N} n_k\right)^d, \quad \text{otherwise.} \end{array} \tag{4.5}$$

The expression of the bound in Theorem 4.1 is based on a broad result of discrete geometry [234]. We then relied on Zaslavsky's Theorem [208] and Whitney's formula to construct a specific arrangement of convex partitions for any set of parameters $d, N, n_1, \dots, n_N \in \mathbb{N}\backslash\{0\}$ that achieves the bound. The proof of Theorem 4.1 is given in 4.6. We now discuss the result and its implications.

- Theorem 4.1 is a generalization of the hyperplane-arrangement bound. Indeed, let us consider the number of regions generated by an arrangement of N hyperplanes: each hyperplane defines a 2-partition of $\mathbb{R}^d$ and the bound yields $\beta^d(2, \dots, 2) = 1 + \sum_{k=1}^{\min(d,N)} \sum_{1 \le \ell_1 < \dots < \ell_k \le N} 1 = 1 + \sum_{k=1}^{\min(d,N)} \binom{N}{k} = \sum_{k=0}^{\min(d,N)} \binom{N}{k}$, which is known to be exactly the number of convex regions generated by an arrangement of N hyperplanes in general position [208].

- The naive upper bound can be rewritten as $\prod_{k=1}^{N} n_k = \prod_{k=1}^{N} ((n_k - 1) + 1) = 1 + \sum_{k=1}^{N} \sum_{1 \le \ell_1 < \dots < \ell_k \le N} (n_{\ell_1} - 1) \cdots (n_{\ell_k} - 1)$. This shows that when $N \le d$, the naive bound is optimal. By contrast, when $N > d$, the dimension enforces the existence of one or more empty intersections between regions of different partitions. This is illustrated in Figure 4.5 with a simple example.

- For partitions with the same number n of regions, we introduce the simpler notation $\beta_N^d(n) \coloneqq \beta^d(n, \dots, n) = 1 + \sum_{k=1}^{\min(d,N)} \sum_{1 \le \ell_1 < \dots < \ell_k \le N} (n-1)^k = \sum_{k=1}^{\min(d,N)} \binom{N}{k} (n-1)^k \le \min(n^N, (1 + N(n-1))^d)$.

- The bound is reached when the partitions Π_k are made of the regions of the arrangement of $(n_k - 1)$ distinct parallel hyperplanes, where the hyperplanes are in general position when only one per partition is selected (more detailed in the proof in 4.6).

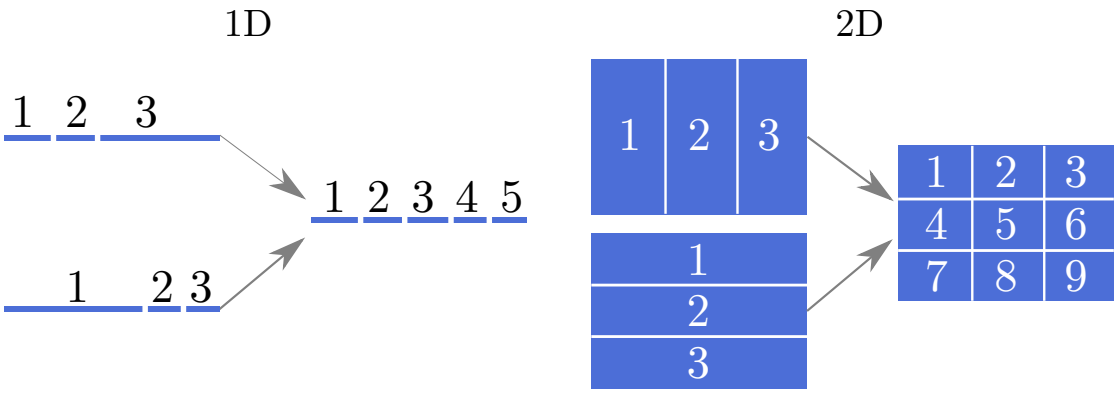

Figure 4.5: Arrangement of two convex partitions with 3 regions each. While in 2D the maximal number of regions is $3 \times 3 = 9$, this cannot be reached in 1D, for which the maximum is 5.

Remark 4.1. *After the disclosure of our work on arXiv, we became aware of [228], which contains highly relevant results on the complexity of Maxout NNs. From Theorem 4.1, we can directly recover the sharp bound on the number of regions of a shallow Maxout NN recently given in [228, Theorem 3.7]. Regarding the converse, i.e. inferring Theorem 4.1 from [228, Theorem 3.7], we believe that it could perhaps be done but it is not immediate. Indeed, [228, Theorem 3.7] is specific to convex partitions that are the linear partitions of the maximum of affine functions, i.e. only specific CPWL functions. Note that the proofs in [228] rely on tropical geometry, which gives an overall perspective very different from ours.*

4.3.2 Single Hidden-Layer: Bound for the Sum and Vectorization Operations

The sum and vectorization of CPWL functions both yield the same bound on the number of linear convex regions. We now give a novel optimal bound in Proposition 4.2. The proof is given in 4.6.

Proposition 4.2. *Let $f_1, \dots, f_N \colon \mathbb{R}^d \to \mathbb{R}$ be CPWL functions with $\kappa_1, \dots, \kappa_N$ convex linear regions. The number of convex linear regions of the sum $(f_1 + \cdots + f_N)$ and of the vector-valued function $(f_1, \dots, f_N)$ can be bounded by a polynomial in $\kappa_1, \dots, \kappa_N$ of degree $\min(d, N)$, namely*

$$\kappa_{f_1+\cdots+f_N} \leq \beta^d(\kappa_1, \dots, \kappa_N), \tag{4.6}$$

$$\kappa_{(f_1,\dots,f_N)} \leq \beta^d(\kappa_1, \dots, \kappa_N), \tag{4.7}$$

and these bounds are sharp.

Remark 4.2. *Bounds similar to the ones given in Proposition 4.2 have recently been derived for one hidden-layer Maxout NNs [228]. The latter work is a specific instance of our setting, in which the CPWL functions considered are the maximum of a finite set of*

affine functions.

As an illustration of Proposition 4.2 and Theorem 4.1, we give some direct implications on the number of regions of some building blocks of CPWL NNs before going deeper.

Ridge Functions Consider the ridge expansion $f_R\colon \mathbf{x} \mapsto \sum_{k=1}^{N} \lambda_k \mathrm{ReLU}(\mathbf{w}_k^T\mathbf{x} + b_k)$, where $\mathbf{w}_k \in \mathbb{R}^d$ and $b_k \in \mathbb{R}$. The number κ_{Ridge} of linear convex regions of f_R is upper-bounded as

$$\kappa_{\mathrm{Ridge}} \leq \beta_N^d(2) = \sum_{k=0}^{\min(d,N)} \binom{N}{k} \leq \min(2^N, (N+1)^d), \tag{4.8}$$

and the bound is tight.

Max-Pooling The kth component of the max-pooling operation $\boldsymbol{f}_{\mathrm{mp}}\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ takes the form $\boldsymbol{f}_{\mathrm{mp}}^k(x_1, \ldots, x_d) = \max_{p \in I_k}(x_p)$, where I_k is a set of chosen cardinality N of "neighboring" coordinate indices. The number κ_{mp} of convex linear regions of the max-pooling operation is upper-bounded as

$$\kappa_{\mathrm{mp}} \leq \beta_{d'}^d(N) = \sum_{k=0}^{\min(d,d')} \binom{d'}{k} (N-1)^k. \tag{4.9}$$

Generalized Hinging Hyperplanes (GHH) Consider the GHH expansion $\mathbb{R}^d \to \mathbb{R}$ $f_G = \sum_{k=1}^{N} \epsilon_k \max(f_1^k, \ldots, f_{d+1}^k)$, where f_p^k are affine functions and $\epsilon_k = \pm 1$ [171]. The number κ_{GHH} of convex linear regions of f_G is upper-bounded as

$$\kappa_{\mathrm{GHH}} \leq \beta_N^d(d+1) = \sum_{k=0}^{\min(d,N)} \binom{N}{k} d^k \leq \min\left((d+1)^N, (Nd+1)^d\right). \tag{4.10}$$

GroupSort Layer The sort operation takes as input a vector $\mathbf{x} \in \mathbb{R}^d$ and simply sorts its components. For any permutation σ of the set $\{1, \ldots, d\}$, we define the uniquely-ordered region $P_\sigma = \{\mathbf{x} \in \mathbb{R}^d\colon x_{\sigma(1)} \leq \cdots \leq x_{\sigma(d)}\}$, where x_k is the kth component of $\mathbf{x}$. These regions are convex as intersections of half-spaces and the sort operation agrees on them with distinct affine functions, namely, permutations. We infer that the sort operation has exactly $d!$ linear convex regions and the same number of projection regions.

The GroupSort activation was recently introduced and shown to be beneficial in the context of Lipschitz-constrained learning [82]. It generalizes the minmax and sort activations: it splits the pre-activation into a chosen number n_g of groups of size g_s (with $n_g g_s = d$),

sorts each pre-activation of each group in ascending order, and outputs the combined sorted groups. Each group produces $g_s!$ linear convex regions which are invariant along the coordinates that are not in the group. We infer the number of linear convex regions of the GroupSort activation to be $\kappa_{\text{GS}} = (g_s!)^{n_g}$, which can be bounded as

$$(g_s/2)^{d/2} \leq \kappa_{\text{GS}} \leq (g_s^{g_s})^{n_g} = g_s^d, \tag{4.11}$$

where we have used the known inequalities $(n/2)^{n/2} \leq n! \leq n^n$. The bounds support the intuition that larger group sizes generate more regions than smaller ones. Note, however, that they simultaneously increase the computational complexity of the layer.

PWLU The PWLU [232] is a learnable CPWL activation function with control points placed on a grid and with fixed linear regions (namely simplices whose vertices are control points). In its 2D version, a PWLU $\varphi_{\text{PWLU}} \colon \mathbb{R}^2 \to \mathbb{R}$ with M^2 control points has $2(M-1)^2$ linear regions that are triangles, see Figure 4.6 for an illustration with $M = 4$, and see [232, Figure 5] for a more generic representation of PWLUs. Consider the one-hidden layer $\mathbb{R}^d \to \mathbb{R}$ PWLU NN $f_{\text{PWLU}}(\mathbf{x}) = \sum_{k=1}^{N} \varphi_{\text{PWLU}}^k(\mathbf{W}_k\mathbf{x})$ with 2D PWLU activations φ_{PWLU}^k with M^2 control points and corresponding weight matrices $\mathbf{W}_k \in \mathbb{R}^{2\times d}$. The number κ_{PWLU} of convex linear regions of this PWLU NN is upper-bounded as

$$\begin{aligned}
\kappa_{\text{PWLU}} \leq \beta_N^d\left(2(M-1)^2\right) &= \sum_{k=1}^{\min(d,N)} \binom{N}{k} (2(M-1)^2-1)^k && (4.12)\\
&\leq \min((2(M-1)^2)^N, (1+N(2(M-1)^2-1))^d) && (4.13)\\
&\leq \min((2M^2)^N, (1+N(2M^2))^d). && (4.14)
\end{aligned}$$

Our framework also allows one to derive bounds for NNs activated with higher dimensional PWLUs, but we are not aware of their use in practice.

4.3.3 Multiple Hidden-Layers: Compositional Bounds

The architecture of a CPWL NN $\mathbb{R}^{d_1} \to \mathbb{R}^{d_{L+1}}$ is specified by its depth L, its layer dimensions $(d_1, \ldots, d_{L+1})$, and its activation complexity $\kappa_{\ell,k}$ at each node (ℓ, k), which is naturally depicted by the number of linear convex regions of the kth component of the ℓth composed function (Figure 4.6). Theorem 4.2 below yields precise bounds on the maximal number of convex linear regions of any CPWL NN. It is complemented by Corollary 4.1 which tackles the following question: given a CPWL NN with fixed input and output dimensions, how is the maximal number of regions related to depth, width, and activation complexities? Our results confirm and generalize the following qualitative intuitions:

- (i) depth can exponentially increase the complexity of the generated function;

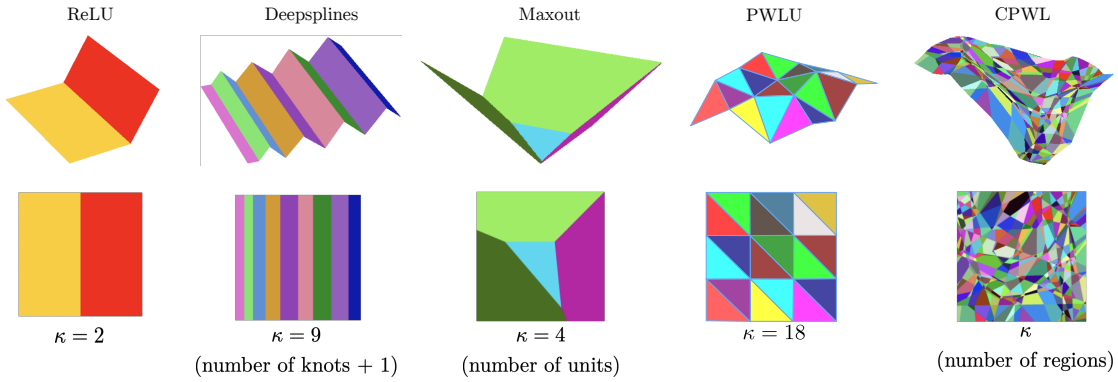

Figure 4.6: Partition and complexity of some CPWL components.

- (ii) width and activation complexity, on the contrary, can only increase the number of linear convex regions of the generated function polynomially;
- (iii) layers with small dimensions reduce the maximal number of regions produced by the NN, especially if they are located toward the input of the NN. This bottleneck effect stems from the upper bound given in Theorem 4.1.

Note that (i) is well known and was first proven in [214], (ii) is in agreement with the recent results in [228] obtained for the particular instance of Maxout NNs, and (iii) was observed for ReLU NNs in [73, 235].

Theorem 4.2. *The maximal number $\kappa_{\max}$ of convex linear regions of a CPWL NN with depth L, layer dimensions $(d_1, \dots, d_{L+1})$, and activation complexities $\kappa_{\ell,k}$ for $k = 1, \dots, d_{\ell+1}$ and $\ell = 1, \dots, L$, is bounded as*

$$\prod_{\ell=1}^{L} \alpha^{\min(d_1,\dots,d_{L+1})}(\kappa_{\ell,1}, \dots, \kappa_{\ell,d_{\ell+1}}) \leq \kappa_{\max} \leq \prod_{\ell=1}^{L} \beta^{\min(d_1,\dots,d_{\ell})}(\kappa_{\ell,1}, \dots, \kappa_{\ell,d_{\ell+1}}), \quad (4.15)$$

where $\beta^{\cdot}(\cdot)$ is the upper bound on the number of regions of an arrangement of convex partitions (Theorem 4.1) and where

$$\alpha^{\min(d_1,\dots,d_{L+1})}(\kappa_{\ell,1}, \dots, \kappa_{\ell,d_{\ell+1}}) = \max_{\tau \in \mathcal{T}_{d_\ell}} \prod_{r=1}^{\min(d_1,\dots,d_{L+1})} \sum_{k \in \tau^{-1}(\{r\})} \kappa_{\ell,k}. \quad (4.16)$$

There, $\mathcal{T}_{d_\ell}$ denotes the set of all mappings from $\{k \in \mathbb{N} \colon 1 \leq k \leq d_{\ell+1}\}$ to $\{k \in \mathbb{N} \colon 1 \leq k \leq \min(d_1, \dots, d_{L+1})\}$, and $\tau^{-1}(\{r\})$ denotes the preimage of $\{r\}$ under τ.

Corollary 4.1. *The maximal number $\kappa_{\max}$ of convex linear regions of a CPWL NN with L layers, layer dimensions $(d_{\text{in}}, W, \dots, W, d_{\text{out}})$, with $d_{\text{in}}, W, d_{\text{out}} \in \mathbb{N} \backslash \{0\}$ and $W \geq d_{\text{in}}$, where each component of the composed functions has κ linear convex regions is bounded as*

$$(\kappa \lfloor W/d^* \rfloor)^{Ld^*} \leq \kappa_{\max} \leq (\kappa W)^{Ld_{\text{in}}}, \quad (4.17)$$

where $d^* = \min(d_{\text{in}}, d_{\text{out}})$.

Corollary 4.2. *The bounds given in Theorem 4.2 and Corollary 4.1 also apply to the maximal number of projection regions of a CPWL NN and, equivalently, to its maximal number of distinct affine pieces.*

The proofs of Theorem 4.2 and of its corollaries can be found in 4.6.4.

4.3.4 Application to Some Popular CPWL NNs

In the sequel, we consider the CPWL NN $\boldsymbol{f}_L \circ \cdots \circ \boldsymbol{f}_1$ where $\boldsymbol{f}_\ell \colon \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell+1}}$. We now apply Theorem 4.2 to bound the maximal number of convex linear regions produced by the most popular architectures. Note that the lower bound given in Theorem 4.2 only applies to CPWL NNs with pointwise activation functions. This includes ReLU and, more generally, deepspline NNs. The reason is that the lower bound of Theorem 4.2 was found by building a deepspline NN.

ReLU/PReLU/leaky ReLU NNs In a ReLU NN, the kth component $f_{\ell,k}$ of $\boldsymbol{f}_\ell$ takes the form $f_{\ell,k} \colon \mathbf{x} \mapsto \text{ReLU}(\mathbf{w}_{\ell,k}\mathbf{x} + b_{\ell,k})$ and has two convex linear regions (half-spaces). Theorem 4.2 then yields

$$\kappa_{\text{ReLU}} \leq \prod_{\ell=1}^{L} \sum_{k=0}^{\min(d_1,\dots,d_\ell)} \binom{d_{\ell+1}}{k}, \tag{4.18}$$

which is the bound proposed in [214]. However, it is not the tightest upper bound known [73]. The reason is that the ReLU function is only a very specific instance of 1D CPWL functions with 2 linear regions: the image of the half real line $(-\infty, 0]$ by the ReLU function is only the singleton $\{0\}$. This reduces the apparent dimension of the problem for any region that would not fire all neurons. This observation was exploited in [73] to get a better estimate. In that sense, (4.18) is better tailored to PReLU and Leaky ReLU NNs, which have activations with two nonzero slope regions.

Deepspline NN Deepspline NNs have learnable pointwise 1D CPWL activation functions [215, 216, 236]. Given activation functions with $(\kappa - 1)$ knots (at most κ linear convex regions), the number of linear convex regions of the NN is bounded as

$$\kappa_{\text{Deepspline}} \leq \prod_{\ell=1}^{L} \sum_{k=0}^{\min(d_1,\dots,d_\ell)} \binom{d_{\ell+1}}{k} (\kappa - 1)^k. \tag{4.19}$$

Maxout NN In a Maxout NN with κ units, the kth component f_ℓ^k of $\boldsymbol{f}_\ell$ takes the form $f_\ell^k : \mathbf{x} \mapsto \max(h_{\ell,k}^1, \dots, h_{\ell,k}^\kappa)$, where $h_{\ell,k}^1, \dots, h_{\ell,k}^\kappa$ are learnable affine functions [175]. Theorem 4.2 yields that

$$\kappa_{\text{Maxout}} \leq \prod_{\ell=1}^{L} \sum_{k=0}^{\min(d_1,\dots,d_\ell)} \binom{d_{\ell+1}}{k} (\kappa - 1)^k. \tag{4.20}$$

This bound is an improvement over [73]. In their work they plug $d_\ell = d$ for $\ell = 1, \dots, L$ and obtain the bound $2^{\frac{\kappa(\kappa-1)}{2} dL}$, to be compared to κ^{dL} for (4.20).

GroupSort NNs To bound the number of linear convex regions of a GroupSort NN [82] with the same group size g_s in each layer, we consider for each composition the arrangement of $d_{\ell+1}/g_s$ convex partitions (one per group) with $g_s!$ regions each and obtain that

$$\kappa_{\text{GroupSort}} \leq \prod_{\ell=1}^{L} \sum_{k=0}^{\min(d_1,\dots,d_\ell)} \binom{d/g_s}{k} (g_s! - 1)^k. \tag{4.21}$$

These bounds provide an intuition of the role of the hyperparameters of CPWL NNs in terms of expressiveness. For instance, the number of units in Maxout NNs plays a role in the bound that is analogous to that of the number of knots of the activation functions in deepspline NNs. However, these two architectures do not induce the same implementation complexity. To increase the activation complexity by one unit, Maxout requires the inclusion of an additional learnable multidimensional affine function, whereas deepspline simply requires the insertion of one more knot to a 1D CPWL function.

While it is tempting to compare architectures on the sole basis of their expressiveness, it can be very delicate to draw generic practical conclusions from this comparison. The final choice of an architecture is guided by a tradeoff between expressiveness, computation complexity, memory usage, and ability to learn over the functional space. For instance, an increase in the group size of a GroupSort activation function increases the expressiveness with no additional parameters, but usually small group sizes are favored to keep the computational impact limited.

4.4 Expected Number of Regions Produced by CPWL NNs Along 1D Paths

In Section 4.3, we found that depth increases the expressiveness of the model exponentially when the corresponding metric is the maximal number of regions. However, the compositions that achieve the lower bound of Theorem 4.2 could be very specific and hard to reach in practice.

The composition $(\boldsymbol{f}_2 \circ \boldsymbol{f}_1)$ of two CPWL functions results in the partitioning of each linear region of $\boldsymbol{f}_1$ into smaller linear pieces. The successive compositions $(\boldsymbol{f}_\ell \circ \cdots \circ \boldsymbol{f}_1)$ have regions that are obtained from splitting of the regions of the previous compositions (Figure 4.7). As such, we expect the image of each region of the composition to shrink when depth increases, at least for compositions with reasonable gradients magnitude (~ 1). The extent of the split should therefore depend on the depth of the composition. The more there are regions produced by the first compositions, the fewer splits each region will undergo after the next compositions. This intuition rules out an exponential growth of the average number of regions with ℓ. This effect has already been revealed for ReLU NNs in [224] and recently extended to Maxout NNs in [230]. We now aim to prove that it is universal to NNs with any type of CPWL activations under reasonable assumptions.

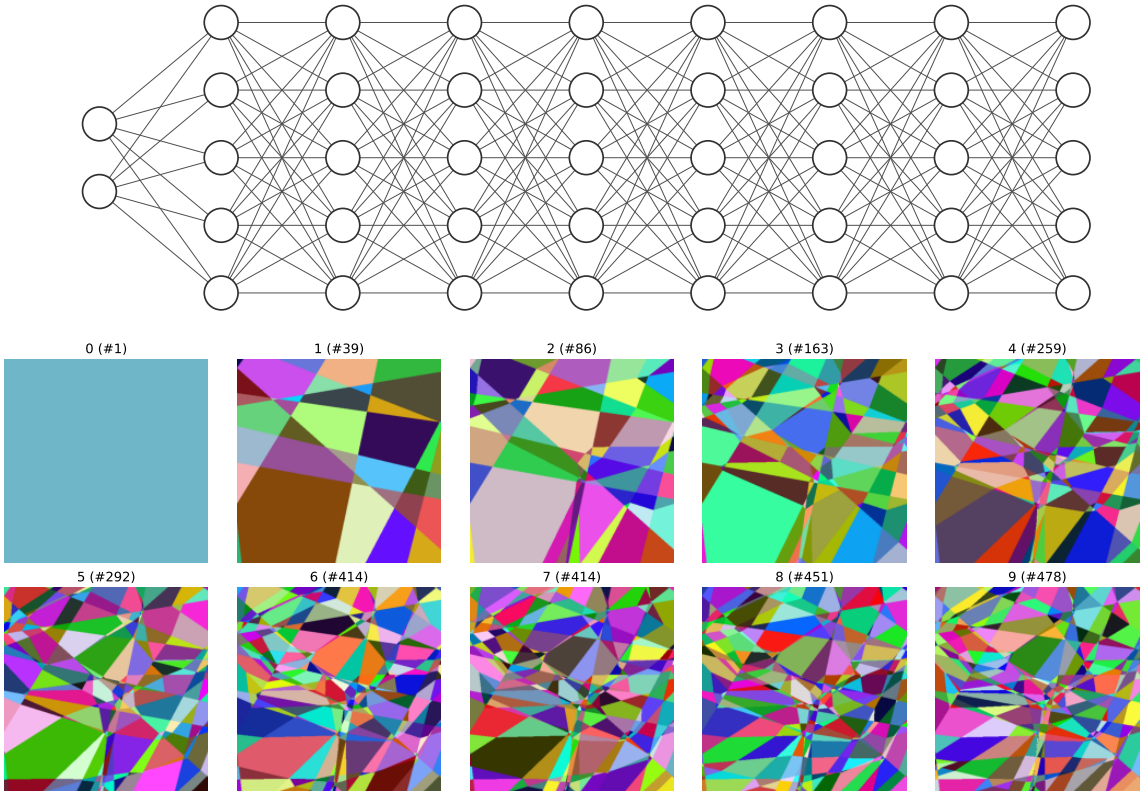

Figure 4.7: Linear region-splitting process for a CPWL NN with absolute-value activation function and randomly generated parameters. The figure shows the linear regions of the mapping after k activation layers, for $k = 0, \ldots, 9$. From one layer to the next, the regions are partitioned into smaller pieces. The number of linear regions is indicated in parentheses and suggests that the splitting process saturates with depth. The regions were numerically identified by evaluating the Jacobian of the mapping on a very fine grid.

Throughout this section, we consider CPWL functions $\boldsymbol{f}_{\boldsymbol{\theta}}$ parameterized by random parameters $\boldsymbol{\theta}$. We shall specify the parameterization and characteristics of the underlying stochastic model whenever needed. The natural extension of Section 4.3 is to estimate

the expected number of regions of compositions of randomly generated CPWL functions. This task seems unfortunately very complex as it mixes stochasticity and combinatorial geometry. It would involve an overly heavy framework with the risk to lose focus on the high-level intuition. Instead, we propose a simpler but closely related metric: the expected density of regions along 1D paths. This quantity is valuable in practice since it gives the expected number of linear regions that are found in-between two locations of the input space that are 1 unit distance apart. In addition, the inverse of the density gives a rough measure of the average size of a linear region along one direction.

4.4.1 Knot Density

The characterization of the density of linear regions of CPWL NNs along one-dimensional paths requires the introduction of some mathematical concepts.

1D CPWL Path A 1D CPWL path denotes in the sequel any CPWL mapping $\boldsymbol{\gamma}\colon S \to \mathbb{R}^d$ on a closed segment $S = [a, b]$ $(a, b \in \mathbb{R})$ with finitely many knots. This path will serve to "navigate" within the input domain of CPWL NNs for counting the linear regions. The length of $\boldsymbol{\gamma}$ is computed as $\text{Len}(\boldsymbol{\gamma}) \coloneqq \int_{t\in S} \|\frac{\mathrm{d}\boldsymbol{\gamma}}{\mathrm{d}t}\|_2 \mathrm{d}t$. Note that $\boldsymbol{\gamma}$ is a parameterization of what is often referred to as a polygonal chain. In this Section we only study the density of linear regions along CPWL paths because of their simplicity and connections with CPWL NNs, e.g. the composition of a CPWL path and a CPWL NN is again a CPWL path. This choice is, however, not very restrictive since CPWL paths can approximate any continuous path arbitrarily close.

Knot Density Along a Path Given a 1D CPWL path, the goal is to characterize the complexity of a CPWL NN along it. Informally, the number of knots of a CPWL NN along the path is the number of times the path crosses regions. This intuitive definition is unfortunately not sufficiently precise since it does not specify how to count knots when some nonzero-length portion of the path $\boldsymbol{\gamma}$ is contained in a face of a linear region, see in Figure 4.8 for an example. To avoid any ambiguity, we introduce the characteristic function

$$\boldsymbol{\varphi}_{\boldsymbol{f}}^{\boldsymbol{\gamma}}\colon \mathbb{R} \to \mathbb{R}^K \tag{4.22}$$

$$t \mapsto (\mathbb{1}_{\Omega_1}(\boldsymbol{\gamma}(t)), \ldots, \mathbb{1}_{\Omega_K}(\boldsymbol{\gamma}(t))) \tag{4.23}$$

of a CPWL $\boldsymbol{f}$ along $\boldsymbol{\gamma}$, where the sets Ω_k are the projection regions of $\boldsymbol{f}$ and $\mathbb{1}_{\Omega_k}(\gamma(t)) = 1$ if $\boldsymbol{\gamma}(t) \in \Omega_k$ and $\mathbb{1}_{\Omega_k}(\gamma(t)) = 0$ otherwise. Since $\boldsymbol{\gamma}$ is continuous with finitely many knots, and since the projection regions are unions of polyhedrons, $\varphi_{\boldsymbol{f}}^{\boldsymbol{\gamma}}$ is a binary function with finitely many jumps, see Figure 4.8. Note that $\boldsymbol{\varphi}_{\boldsymbol{f}}^{\boldsymbol{\gamma}}$ uniquely identifies the supporting affine function active at location $\boldsymbol{\gamma}(t)$. Hence, in practice, the knowledge of $\boldsymbol{f}(\boldsymbol{\gamma}(t))$ and

$\nabla \boldsymbol{f}(\boldsymbol{\gamma}(t))$, which is computable in any deep-learning library, suffice to identify $\varphi^{\boldsymbol{\gamma}}_{\boldsymbol{f}}(t)$.

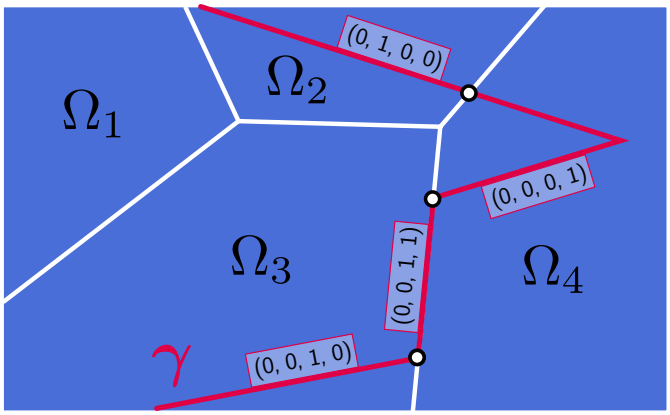

Figure 4.8: Example of a 1D CPWL path $\boldsymbol{\gamma}\colon \mathbb{R} \to \mathbb{R}^2$. The value of the characteristic function $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f}}$ along $\boldsymbol{\gamma}$ is given as a 4D vector and allow one to identify the 3 knots along $\boldsymbol{\gamma}$.

Definition 4.6 (Knot density along 1D CPWL curves)**.** *Let $\boldsymbol{f}\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ be a CPWL function, $\boldsymbol{\gamma}$ a 1D CPWL path, and $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f}}$ the characteristic function of $\boldsymbol{f}$ along $\boldsymbol{\gamma}$. The number $\mathrm{kt}^{\boldsymbol{\gamma}}_{\boldsymbol{f}}$ of knots of $\boldsymbol{f}$ along $\boldsymbol{\gamma}$ is the number of discontinuous points of the piecewise-constant function $\varphi^{\boldsymbol{\gamma}}_{\boldsymbol{f}}$. The knot density $\lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}}$ of $\boldsymbol{f}$ along $\boldsymbol{\gamma}$ is defined as*

$$\lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}} = \mathrm{kt}^{\boldsymbol{\gamma}}_{\boldsymbol{f}} / \mathrm{Len}(\boldsymbol{\gamma}), \tag{4.24}$$

where $\mathrm{Len}(\boldsymbol{\gamma})$ is the length of $\boldsymbol{\gamma}$.

We stress that alternative definitions of the knot density that correspond to the same informal intuition are possible, but they would differ when the path $\boldsymbol{\gamma}$ follows the boundaries of some projection regions. In the sequel, this will not matter since, in any reasonable stochastic framework, the path does not follow some boundaries almost surely.

The knot density along a path is subadditive for the sum and vectorization of CPWL functions.

Proposition 4.3. *Let $\boldsymbol{\gamma}\colon S \to \mathbb{R}^d$ be a 1D CPWL path on the segment $S \subset \mathbb{R}$ and let $\boldsymbol{f}_1\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ and $\boldsymbol{f}_2\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ be two CPWL functions. The knot density along $\boldsymbol{\gamma}$ of either the sum $\boldsymbol{f}_1 + \boldsymbol{f}_2$ or of the vectorized function $(\boldsymbol{f}_1, \boldsymbol{f}_2)$ is bounded as*

$$\lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1+\boldsymbol{f}_2} \leq \lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1} + \lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_2}, \tag{4.25}$$

$$\lambda^{\boldsymbol{\gamma}}_{(\boldsymbol{f}_1,\boldsymbol{f}_2)} \leq \lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1} + \lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_2}, \tag{4.26}$$

where $\lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1}$ and $\lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_2}$ are the knot density of $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ along $\boldsymbol{\gamma}$, respectively.

Proof. Consider two CPWL functions $\boldsymbol{f}_1, \boldsymbol{f}_2$ with characteristic functions $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_1$ and $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_2$ along $\boldsymbol{\gamma}$ and projection regions $(\Omega^1_k)_{k=1}^{K_1}$ and $(\Omega^2_k)_{k=1}^{K_2}$. Let $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{(1,2)}$ denote the characteristic

function of $(\boldsymbol{f}_1, \boldsymbol{f}_2)$ and $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{1+2}$ the one of $\boldsymbol{f}_1 + \boldsymbol{f}_2$ along $\boldsymbol{\gamma}$. Consider a subset $R \subset S$ on which $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_1$ and $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_2$ are continuous. Following the definition of the characteristic function, any projection region Ω of $\boldsymbol{f}_1$ or of $\boldsymbol{f}_2$ either entirely contains $\boldsymbol{\gamma}(R)$ or does not intersect with it; otherwise, $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_1$ or $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_2$ would not be continuous on R. Any projection region $\Omega^{(1,2)}$ of $(\boldsymbol{f}_1, \boldsymbol{f}_2)$ is a nonempty intersection of the form $\Omega^1_p \cap \Omega^2_q$, with $p \in [K_1]$ and $q \in [K_2]$. Since the regions Ω^1_p and Ω^2_q either entirely contain $\boldsymbol{\gamma}(R)$ or do not intersect with it, the same holds true for $\Omega^{(1,2)}$, which implies that $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{(1,2)}$ must be continuous on R. The same argument holds true for $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{(1+2)}$, the only difference being that the projection regions of $\boldsymbol{f}_1 + \boldsymbol{f}_2$ are unions of nonempty subsets of the form $\Omega^1_p \cap \Omega^2_q$, which also has the same implications. So, on the one hand, we proved that, where $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ are continuous, $\boldsymbol{\varphi}_{1+2}$ and $\boldsymbol{\varphi}_{(1,2)}$ are also continuous. On the other hand, the number of points where either $\boldsymbol{\varphi}_1$ or $\boldsymbol{\varphi}_2$ is discontinuous is no greater than the number of points where $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ are discontinuous, which concludes the proof. □

Proposition 4.4. *Let $\boldsymbol{\gamma}\colon S \to \mathbb{R}^{d_1}$ be a 1D CPWL path on $S \subset \mathbb{R}$ and let $\boldsymbol{f}_1\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ and $\boldsymbol{f}_2\colon \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ be two CPWL functions. Then, the knot density of $\boldsymbol{f}_2 \circ \boldsymbol{f}_1$ on $\boldsymbol{\gamma}$ is bounded as*

$$\lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_2 \circ \boldsymbol{f}_1} \leq \lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1} + \left(\frac{\mathrm{Len}(\boldsymbol{f}_1 \circ \boldsymbol{\gamma})}{\mathrm{Len}(\boldsymbol{\gamma})} \right) \lambda^{\boldsymbol{f}_1 \circ \boldsymbol{\gamma}}_{\boldsymbol{f}_2}, \tag{4.27}$$

where $\lambda^{\boldsymbol{\gamma}}_1$ is the knot density of $\boldsymbol{f}_1$ along $\boldsymbol{\gamma}$ and $\lambda^{\boldsymbol{\gamma}}_2$ the one of $\boldsymbol{f}_2$ along $\boldsymbol{f}_1 \circ \boldsymbol{\gamma}$.

Proof. Consider the characteristic function $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1}$ of $\boldsymbol{f}_1$ on $\boldsymbol{\gamma}$ and $\boldsymbol{\varphi}^{\boldsymbol{f}_1 \circ \boldsymbol{\gamma}}_{\boldsymbol{f}_2}$ of $\boldsymbol{f}_2$ on $\boldsymbol{f}_1 \circ \boldsymbol{\gamma}$. Moreover, consider a subset $R \subset S$ on which both $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1}$ and $\boldsymbol{\varphi}^{\boldsymbol{f}_1 \circ \boldsymbol{\gamma}}_{\boldsymbol{f}_2}$ are continuous. The projection regions of $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are denoted by $(\Omega^1_k)_{k=1}^{K_1}$ and $(\Omega^2_k)_{k=1}^{K_2}$. On any region $\Lambda_{p,q} = \Omega^1_p \cap \boldsymbol{f}_1^{-1}(\Omega^2_q)$, the function $\boldsymbol{f}_2 \circ \boldsymbol{f}_1$ is affine, meaning that each projection region of $\boldsymbol{f}_2 \circ \boldsymbol{f}_1$ is a union of some of the regions $\Lambda_{p,q}$. Each region Ω^1_p either entirely contains the subset $\boldsymbol{\gamma}(R)$ or does not intersect with it; otherwise, $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1}$ would not be continuous on R. We now remark that $\boldsymbol{\varphi}^{\boldsymbol{f}_1 \circ \boldsymbol{\gamma}}_{\boldsymbol{f}_2}(t) = (\mathbb{1}_{\Omega^2_1}(\boldsymbol{f}_1 \circ \boldsymbol{\gamma}(t)), \dots, \mathbb{1}_{\Omega^2_{K_2}}(\boldsymbol{f}_1 \circ \boldsymbol{\gamma}(t))) = (\mathbb{1}_{\boldsymbol{f}_1^{-1}(\Omega^2_1)}(\boldsymbol{\gamma}(t)), \dots, \mathbb{1}_{\boldsymbol{f}_1^{-1}(\Omega^2_{K_2})}(\boldsymbol{\gamma}(t)))$, which means that each region $\boldsymbol{f}_1^{-1}(\Omega^2_q)$ either entirely contains the subset $\boldsymbol{\gamma}(R)$ or does not intersect with it since, otherwise, $\boldsymbol{\varphi}^{\boldsymbol{f}_1 \circ \boldsymbol{\gamma}}_{\boldsymbol{f}_2}$ would not be continuous on R. We therefore have that each region $\Lambda_{p,q}$ either entirely contains the subset $\boldsymbol{\gamma}(R)$ or does not intersect with it. Consequently, the same holds true for union of regions $\Lambda_{p,q}$ and, as a result, for the projection regions of $\boldsymbol{f}_2 \circ \boldsymbol{f}_1$. This shows that, where $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1}$ and $\boldsymbol{\varphi}^{\boldsymbol{f}_1 \circ \boldsymbol{\gamma}}_{\boldsymbol{f}_2}$ are continuous, $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f}_2 \circ \boldsymbol{f}_1}$ is also continuous. In short, the number of points of discontinuities of $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f}_2 \circ \boldsymbol{f}_1}$ is no greater than the the number of points of discontinuity of either $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f}_1}$ or $\boldsymbol{\varphi}^{\boldsymbol{f}_1 \circ \boldsymbol{\gamma}}_{\boldsymbol{f}_2}$, which concludes the proof. □

4.4.2 Knot Density of CPWL Layers

The goal of this subsection is to show that the knot density is well behaved for classical CPWL NN layers, which justifies the assumption (i) of Theorem 4.3 and Corollary 4.3.

The proofs can be found in 4.7.

Proposition 4.5 (Knot density - ReLU)**.** *Let $(\boldsymbol{w}, b) \in \mathbb{R}^d \times \mathbb{R}$ be independent random variables with bounded probability density functions ρ_b for b and ρ_w for the components of $\boldsymbol{w}$, which are i.i.d. Then, the expected knot density of the ReLU CPWL component $\mathbf{x} \mapsto \mathrm{ReLU}(\boldsymbol{w}^T\mathbf{x} + b)$ along any 1D CPWL path $\boldsymbol{\gamma}$ is bounded as*

$$\mathbb{E}\left[\lambda^{\gamma}_{\mathrm{ReLU}}\right] \leq \sqrt{\mathbb{E}[w_1^2]} \sup_{t\in\mathbb{R}} \rho_b(t). \tag{4.28}$$

In particular, when b and the components of $\boldsymbol{w}$ are normally distributed with zero mean and standard deviation σ_b and σ_w, respectively, the following tighter bound holds true

$$\mathbb{E}\left[\lambda^{\gamma}_{\mathrm{ReLU}}\right] \leq \frac{\sigma_w}{\pi\sigma_b}. \tag{4.29}$$

When the ReLU activation function is replaced by a 1D CPWL function with a given number K of knots, we conjecture that the bounds can simply be multiplied by K.

Proposition 4.6 (Knot density - Maxout)**.** *Let $((w_{k1}, \ldots, w_{kd}), b_k) \in \mathbb{R}^d \times \mathbb{R}$ for $k = 1, \ldots, K$ be independent random variables with bounded probability density functions ρ_b for any b_k and ρ_w for all components w_{kl} of $\boldsymbol{w}_k$, which are i.i.d. over both $k \in [K]$ and $l \in [d]$. Then, the expected knot density of the rank K Maxout unit $f\colon \mathbf{x} \mapsto \max_{k=1,\ldots,K}(\boldsymbol{w}_k^T\mathbf{x} + b_k)$ along any 1D CPWL path $\boldsymbol{\gamma}$ is bounded as*

$$\mathbb{E}\left[\lambda^{\gamma}_{\mathrm{Maxout}}\right] \leq \sqrt{2}\binom{K}{2}\sigma_w \sup_{t\in\mathbb{R}} \rho_b(t), \tag{4.30}$$

where σ_w is the standard deviation of any w_{kl}. In particular, when b_k and w_{kl} are normally distributed with zero mean and standard deviation σ_b and σ_w, respectively, a tighter bound holds true, according to

$$\mathbb{E}\left[\lambda^{\gamma}_{\mathrm{Maxout}}\right] \leq \sqrt{2}\binom{K}{2}\frac{\sigma_w}{\pi\sigma_b}. \tag{4.31}$$

The bounds provided in Proposition 4.6 grow quadratically in terms of the number of Maxout units; we conjecture the existence of a tighter linear bound.

Proposition 4.7 (Knot density - GroupSort)**.** *Let $(\boldsymbol{w}_k, b_k)$ be as in Proposition 4.6. Then, the expected knot density $\lambda^{\gamma}_{\boldsymbol{f}}$ of the GroupSort layer $\boldsymbol{f}\colon \mathbb{R}^d \to \mathbb{R}^d\colon \mathbf{x} \mapsto \mathrm{GS}_{n_g,g_s}(\boldsymbol{W}\mathbf{x})$, where GS_{n_g,g_s} is the GroupSort activation with n_g groups of size g_s, is bounded along any 1D CPWL path $\boldsymbol{\gamma}$ as*

$$\mathbb{E}\left[\lambda^{\gamma}_{\mathrm{GroupSort}}\right] \leq \frac{\sqrt{2}}{2} d(g_s - 1)\sigma_w \sup_{t\in\mathbb{R}} \rho_b(t), \tag{4.32}$$

where σ_w is the standard deviation of any $w_{k,l}$. In particular, when b_k and w_{kl} are normally distributed with zero mean and standard deviation σ_b and σ_w, respectively, a

tighter bound can be given as

$$\mathbb{E}\left[\lambda^{\gamma}_{\text{GroupSort}}\right] \leq \frac{\sqrt{2}}{2} d(g_s - 1)\frac{\sigma_w}{\pi\sigma_b}. \tag{4.33}$$

For ReLU and Maxout layers with multidimensional outputs, the bounds given in Proposition 4.5 and 4.6 are simply multiplied by the output dimension (see Proposition 4.3). We note that all bounds proposed take the form $(\kappa W \sigma_w \sup_{t\in\mathbb{R}} \rho_b(t))$, where the prefactor κ only depends on the activation function and W is the number of outputs of the layer. The learnable parameters are typically initialized by sampling a uniform or normal distribution with the same characteristics for the biases and the weights of a same layer. In this case, although the characteristics of the distribution usually depend on the input and output dimensions of the layer [173], the quantity $\sigma_w \sup_{t\in\mathbb{R}} \rho_b(t)$ is determined only by the distribution: normal or uniform (since, for these distributions, the supremum of the probability density function is proportional to the standard deviation). All in all, it should be reminded that

- the expected knot density is well defined for learnable CPWL layers;
- with standard initialization methods, it is reasonable to assume that the expected knot density of the components of a CPWL layer depends neither on its width nor on the total depth of the NN (at least at initialization stage).

It is tempting to take advantage of the previous results to adjust the distributions of the weights and biases at initialization in the hope to increase the upper bound and, possibly, the knot density of a NN. The effect is, however, subtle: for instance, if one narrows the distribution of the biases, the bound increases as $\sup_{t\in\mathbb{R}} \rho_b(t)$ increases. While this may increase the average knot density at some specific locations, it will inevitably decrease it elsewhere.

4.4.3 Bounds on the Expected Knot Density of CPWL NNs

In Theorem 4.3 and Corollary 4.3, we introduce two different settings to bound the expected knot density of CPWL NNs. Theorem 4.3 highlights the role played by the gradients of the composed layers: larger gradients allow for a more intense splitting process within the composition and should lead to a greater knot density. With Corollary 4.3, we propose a more practical analysis: given a learning task that dictates the input and output dimensions, how does the expected density of linear regions along 1D curves relate to the depth, width, and activation complexity of the CPWL NN? In accordance with the intuition given in Figure 4.7, depth cannot provide exponentially more linear regions on average. This key result relies mainly on the assumption (ii), which is discussed in Section 4.4.3.

The directional derivative of the function $\boldsymbol{f}$ in the direction $\mathbf{u}$ is denoted by $D_{\mathbf{u}}\{\boldsymbol{f}\}$.

Theorem 4.3. *Let $\boldsymbol{f}_{\boldsymbol{\theta}_1}, \dots, \boldsymbol{f}_{\boldsymbol{\theta}_L}$, with $\boldsymbol{f}_{\boldsymbol{\theta}_\ell} \colon \mathbb{R}^W \to \mathbb{R}^W$, be CPWL functions parameterized by the independent and identically distributed random variables $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_L$. Suppose that there exist $\lambda_0, D_0 \in \mathbb{R}$ such that*

(i) for any 1D CPWL path $\boldsymbol{\gamma}$, $\mathbb{E}[\lambda^{\boldsymbol{\gamma}}_{f_{\boldsymbol{\theta}_\ell,k}}] \leq \lambda_0$, where $f_{\boldsymbol{\theta}_\ell,k}$ is the kth component of $\boldsymbol{f}_{\boldsymbol{\theta}_\ell}$ (bounded expected knot density of the components);

(ii) for any $\mathbf{x}, \mathbf{u} \in \mathbb{R}^W$ with $\|\mathbf{u}\|_2 = 1$, $\mathbb{E}[D_{\mathbf{u}}\{\boldsymbol{f}_{\boldsymbol{\theta}}\}(\mathbf{x})] \leq D_0$ (bounded expected directional derivative).

Then, on any 1D CPWL path $\boldsymbol{\gamma}$, the expected knot density of the CPWL NN is bounded as

$$\mathbb{E}[\lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_{\boldsymbol{\theta}_L} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{\theta}_1}}] \leq \begin{cases} \lambda_0 W \left(\frac{1-D_0^L}{1-D_0}\right), & D_0 \neq 1 \\ \lambda_0 W L, & D_0 = 1. \end{cases} \tag{4.34}$$

Corollary 4.3. *Let $\boldsymbol{f}_{\boldsymbol{\theta}_1}, \dots, \boldsymbol{f}_{\boldsymbol{\theta}_L}$, with $\boldsymbol{f}_{\boldsymbol{\theta}_\ell} \colon \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell+1}}$, be CPWL functions parameterized by the independent and identically distributed random variables $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_L$ and $d_2 = \cdots = d_L = W > d_{L+1}$. Suppose that there exist $\lambda_0, D_0 \in \mathbb{R}$ such that*

(i) for any 1D CPWL path $\boldsymbol{\gamma}$, $\mathbb{E}[\lambda^{\boldsymbol{\gamma}}_{f_{\boldsymbol{\theta}_\ell,k}}] \leq \lambda_0$, where $f_{\boldsymbol{\theta}_\ell,k}$ is the kth component of $\boldsymbol{f}_{\boldsymbol{\theta}_\ell}$ (bounded expected knot density of the components),

(ii) for any $\mathbf{x}, \mathbf{u} \in \mathbb{R}^d$ with $\|\mathbf{u}\|_2 = 1$, $\mathbb{E}[D_{\mathbf{u}}\{\boldsymbol{f}_{\boldsymbol{\theta}_\ell} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{\theta}_1}\}(\mathbf{x})] \leq D_0$, for $1 \leq \ell \leq L$ (bounded expected directional derivative within the composition).

Then, on any 1D CPWL path $\boldsymbol{\gamma}$, the expected knot density of the CPWL NN is bounded as

$$\mathbb{E}[\lambda^{\boldsymbol{\gamma}}_{\boldsymbol{f}_{\boldsymbol{\theta}_L} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{\theta}_1}}] \leq D_0^*(\lambda_0 W L), \tag{4.35}$$

where $D_0^* = \max(D_0, 1)$.

The proof of Theorem 4.3 relies on Lemma 4.4. In the bound presented in this lemma, the expected value is evaluated before taking the supremum, whilst a switch of the order of the operators would yield a much looser bound.

Lemma 4.4. *Let $\boldsymbol{\gamma} \colon S \to \mathbb{R}^d$ be a 1D CPWL path and $\boldsymbol{f}_{\boldsymbol{\theta}} \colon \mathbb{R}^d \to \mathbb{R}^{d'}$ a CPWL function parameterized by the random variable $\boldsymbol{\theta}$ such that, for any $\mathbf{x}, \mathbf{u} \in \mathbb{R}^d$, $\boldsymbol{f}_{\boldsymbol{\theta}}$ is differentiable at $\mathbf{x}$ in direction $\mathbf{u}$ with probability 1. Then, the expected length of the 1D CPWL path $\boldsymbol{f}_{\boldsymbol{\theta}} \circ \boldsymbol{\gamma} \colon S \to \mathbb{R}^{d'}$ is bounded as*

$$\mathbb{E}[\mathrm{Len}(\boldsymbol{f}_{\boldsymbol{\theta}} \circ \boldsymbol{\gamma})] \leq \mathrm{Len}(\boldsymbol{\gamma}) \sup_{\substack{\mathbf{x},\mathbf{u} \in \mathbb{R}^d \\ \|\mathbf{u}\|_2 = 1}} \mathbb{E}[\|D_{\mathbf{u}}\{\boldsymbol{f}_{\boldsymbol{\theta}}\}(\mathbf{x})\|_2]. \tag{4.36}$$

Proof. In what follows, the technical developments originate from the fact that $\boldsymbol{f_\theta}$ is not differentiable everywhere. The function $\boldsymbol{f_\theta} \circ \boldsymbol{\gamma}$ is the composition of two CPWL functions, hence it is CPWL and, therefore, differentiable for almost every $t \in S$. Note, however, that we cannot assert that the Jacobian of $\boldsymbol{f_\theta}$ is well defined at $\boldsymbol{\gamma}(t)$ for almost every $t \in S$. Indeed, whenever $\boldsymbol{\gamma}$ follows the boundary of two projection regions of $\boldsymbol{f_\theta}$, the Jacobian of $\boldsymbol{f_\theta}$ along $\boldsymbol{\gamma}$ becomes ill-posed. This is why the notion of directional derivative is better suited. The characteristic function $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f_\theta}}$ of $\boldsymbol{f_\theta}$ along $\boldsymbol{\gamma}$ is piecewise-constant on S: We can partition S into finitely many convex regions where $\boldsymbol{\varphi}^{\boldsymbol{\gamma}}_{\boldsymbol{f_\theta}}$ is constant. Let $P \subset S$ denote one of these regions and let $Q \subset S$ be a linear convex region of $\boldsymbol{\gamma}$. Following the definition of the characteristic function, there exists a projection region Ω of $\boldsymbol{f_\theta}$ such that $\boldsymbol{\gamma}(\text{int}(P \cap Q))$ either lies entirely in the interior of Ω, or entirely on its boundary. In the first case, $\boldsymbol{f_\theta}$ is differentiable in $\boldsymbol{\gamma}(\text{int}(P \cap Q))$ and we have that $(\boldsymbol{f_\theta} \circ \boldsymbol{\gamma})'(t) = J_{\boldsymbol{f_\theta}}(\boldsymbol{\gamma}(t))\boldsymbol{\gamma}'(t) = D_{\boldsymbol{\gamma}'(t)}\{\boldsymbol{f_\theta}\}(\boldsymbol{\gamma}(t))$ for $t \in \text{int}(P \cap Q)$. In the second case, $\boldsymbol{\gamma}$ is differentiable on $\text{int}(P \cap Q)$ as well, but the Jacobian of $\boldsymbol{f_\theta}$ is undefined. Fortunately, the directional derivative of $\boldsymbol{f_\theta}$ is well defined along $\boldsymbol{\gamma}(t)$ since, for any $t \in \text{int}(P \cap Q)$, there exists $\epsilon > 0$ such that $\tau \mapsto \boldsymbol{f}_\theta(\boldsymbol{\gamma}(t) + \tau\boldsymbol{\gamma}'(t))$ is affine on $(-\epsilon, \epsilon)$. All in all, the relation $(\boldsymbol{f_\theta} \circ \boldsymbol{\gamma})'(t) = D_{\boldsymbol{\gamma}'(t)}\{\boldsymbol{f_\theta}\}(\boldsymbol{\gamma}(t))$ is well defined for any $t \in \text{int}(P \cap Q)$ and, more generally, for almost any $t \in S$ because of the properties of P and Q. We can now write that

$$\begin{aligned}
\mathbb{E}\left[\text{Len}(\boldsymbol{f_\theta} \circ \boldsymbol{\gamma})\right] &= \mathbb{E}\left[\int_{t\in S} \left\|(\boldsymbol{f_\theta} \circ \boldsymbol{\gamma})'(t)\right\|_2 \mathrm{d}t\right] \\
&= \mathbb{E}\left[\int_{t\in S} \left\|D_{\boldsymbol{\gamma}'(t)}\{\boldsymbol{f_\theta}\}(\boldsymbol{\gamma}(t))\right\|_2 \mathrm{d}t\right] \\
&= \int_{t\in S} \mathbb{E}\left[\left\|D_{\boldsymbol{\gamma}'(t)}\{\boldsymbol{f_\theta}\}(\boldsymbol{\gamma}(t))\right\|_2\right] \mathrm{d}t \\
&\leq \sup_{\substack{\mathbf{x},\mathbf{u}\in\mathbb{R}^d \\ \|\mathbf{u}\|_2=1}} \mathbb{E}[\|D_{\mathbf{u}}\{\boldsymbol{f_\theta}\}(\mathbf{x})\|_2] \int_{t\in S} \left\|\boldsymbol{\gamma}'(t)\right\|_2 \mathrm{d}t \\
&= \text{Len}(\boldsymbol{\gamma}) \sup_{\substack{\mathbf{x},\mathbf{u}\in\mathbb{R}^d \\ \|\mathbf{u}\|_2=1}} \mathbb{E}[\|D_{\mathbf{u}}\{\boldsymbol{f_\theta}\}(\mathbf{x})\|_2], \qquad (4.37)
\end{aligned}$$

where we have used Tonelli's theorem to interchange the expectation and the integral. □

Proof. **of Theorem 4.3** Let $\boldsymbol{F}_\ell = \boldsymbol{f}_{\boldsymbol{\theta}_\ell} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{\theta}_1}$. With Lemma 4.4, we have that

$$\mathbb{E}\left[\text{kt}^{\boldsymbol{\gamma}}_{\boldsymbol{F}_\ell}\right] = \mathbb{E}\left[\text{kt}^{\boldsymbol{\gamma}}_{\boldsymbol{f}_{\boldsymbol{\theta}_\ell}\circ\boldsymbol{F}_{L-1}}\right] \leq \mathbb{E}\left[\text{kt}^{\boldsymbol{F}_{\ell-1}\circ\boldsymbol{\gamma}}_{\boldsymbol{f}_{\boldsymbol{\theta}_\ell}}\right] + \mathbb{E}\left[\text{kt}^{\boldsymbol{\gamma}}_{\boldsymbol{F}_{\ell-1}}\right]. \qquad (4.38)$$

We now apply the law of the iterated expectation to obtain that

$$\begin{aligned}
\mathbb{E}\left[\text{kt}^{\boldsymbol{F}_{\ell-1}\circ\boldsymbol{\gamma}}_{\boldsymbol{f}_{\boldsymbol{\theta}_\ell}}\right] &= \mathbb{E}_{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{\ell-1}}\left[\mathbb{E}_{\boldsymbol{\theta}_\ell}\left[\text{kt}^{\boldsymbol{F}_{\ell-1}\circ\boldsymbol{\gamma}}_{\boldsymbol{f}_{\boldsymbol{\theta}_\ell}} | \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{\ell-1}\right]\right] \\
&\leq \mathbb{E}_{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{\ell-1}}\left[d\lambda_0 \text{Len}(\boldsymbol{F}_{\ell-1} \circ \boldsymbol{\gamma})\right], \qquad (4.39)
\end{aligned}$$

where the inequality follows from the first assumption of the theorem, the application of Proposition 4.3, and requires the independence of the random variables. We can now apply Lemma 4.4 recursively to $\boldsymbol{F}_\ell$ and invoke the second assumption of the theorem to infer that

$$\mathbb{E}\left[\mathrm{kt}_{\boldsymbol{f}_{\boldsymbol{\theta}_\ell}}^{\boldsymbol{F}_{\ell-1}\circ\boldsymbol{\gamma}}\right] \leq d\lambda_0 \mathrm{Len}(\boldsymbol{\gamma}) D_0^{\ell-1}. \tag{4.40}$$

All in all, we just proved that

$$\mathbb{E}\left[\mathrm{kt}_{\boldsymbol{F}_\ell}^{\boldsymbol{\gamma}}\right] \leq \mathbb{E}\left[\mathrm{kt}_{\boldsymbol{F}_{\ell-1}}^{\boldsymbol{\gamma}}\right] + d\lambda_0 \mathrm{Len}(\boldsymbol{\gamma}) D_0^{\ell-1}, \tag{4.41}$$

which reads in term of linear densities as

$$\mathbb{E}\left[\lambda_{\boldsymbol{F}_\ell}^{\boldsymbol{\gamma}}\right] \leq \mathbb{E}\left[\lambda_{\boldsymbol{F}_{\ell-1}}^{\boldsymbol{\gamma}}\right] + d\lambda_0 D_0^{\ell-1}. \tag{4.42}$$

This recurrence relation directly yields the announced bound. □

Proof. **of Corollary 4.3** The proof is similar to the proof of Theorem 4.2 except that, with the different second assumption, the quantity $\mathbb{E}_{\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_{\ell-1}}[\mathrm{Len}(\boldsymbol{F}_{\ell-1}\circ\boldsymbol{\gamma})]$ can be bounded by $D_0\mathrm{Len}(\boldsymbol{\gamma})$ (Lemma 4.4). In the end the recurrence relation (4.42) is changed into

$$\mathbb{E}\left[\lambda_{\boldsymbol{F}_\ell}^{\boldsymbol{\gamma}}\right] \leq \mathbb{E}\left[\lambda_{\boldsymbol{F}_{\ell-1}}^{\boldsymbol{\gamma}}\right] + D_0\lambda_0 W. \tag{4.43}$$

□

Discussion of the Compositional Bounds

Our approach relies on the independence of the randomly generated CPWL functions. It usually holds at initialization stage, but it is not true anymore in the learning stage. While this can be regarded as a limitation, it is a legitimate and convenient way to explore and depict the whole function space that a given architecture gives access to.

Assumption (i) of Theorem 4.3 and its corollary (bounded expected knot density of the learnable CPWL components) have been discussed in details in Section 4.4.2, where it was remarked that it is reasonable to assume that λ_0 is independent of W and L.

Theorem 4.3 and Corollary 4.3 differ on Assumption (ii) (well behaved gradients). While the assumption of the theorem seems more natural at first sight (gradient controlled for each layer), the one of the corollary is closer to practical observations. Assumption (ii) of Corollary 4.3 was invoked to bound the expected length of the image of any finite-length 1D CPWL path, independently of the depth of the composition. While early works suggested that this expected length grows exponentially with depth [237], it was recently

shown otherwise in a more realistic setup, both theoretically and experimentally [238]. For instance, for ReLU NNs, with the usual 2/fan-in weight variance, depth typically does not affect the expected length [238]. More generally, a control of the magnitude of the directional derivatives that is independent of the depth is highly desirable in the learning stage for a stable back-propagation algorithm [173] and, in the inference stage, to produce robust models [239]. In short, it is also reasonable to assume that the parameter D_0 depends neither on W nor on L.

The previous discussion suggests a simple and important bound on the density of regions of CPWL NNs. It attributes an identical role to the three sources of complexity, namely depth, width, and activation complexity.

The quality of the proposed bounds seems to be completely determined by the tightness of the bounds in Assumptions (i) and (ii). Based on the proofs of Theorem 4.3 and Corollary 4.3, we believe that the compositional bounds are sharp provided that the expected knot density is uniform (i.e., the same for any 1D CPWL curve) and that the expected norm of the directional derivative is uniform and isotropic within the NN.

4.5 Conclusion

In this chapter, we have investigated the role of depth, width, and activation complexity in the expressiveness of CPWL NNs. By invoking results from combinatorial geometry, we have found that depth has a predominant role over width and activation complexity: it is the only descriptor able to increase the number of linear regions exponentially. However, this exponential growth is only observed for the maximal number of regions. Indeed, when exploring the whole function space produced by a given CPWL NN, we have found that, on average, the number of regions along a line is bounded by a quantity that only depends on the product of the three descriptors. In that perspective, the three complexity parameters have an identical role: no exponential behavior with depth is observed anymore.

The ability to train deeper and deeper NNs has led to major improvements in machine learning. However, depth comes at a price in applications where the NN needs to be stable, for instance by constraining its global Lipschitz constant. In such settings, we, therefore, believe that expressive learnable activations should always be regarded as a valuable opportunity to increase substantially the expressiveness of the model without resorting to deeper NNs, and we develop this idea in Chapter 5 and 6.

4.6 Appendix – Proofs for Section 4.3

4.6.1 Number of Convex vs Projection Regions

Proof. **of Proposition 4.1** The first inequality follows from the fact that there cannot be fewer linear convex regions than affine pieces. Consider two neighboring projection regions Ω_k and Ω_p of $\boldsymbol{f}$, where $1 \leq k < p \leq \rho$, with corresponding affine pieces $\boldsymbol{f}^k \colon \mathbf{x} \mapsto \mathbf{W}_k^T \mathbf{x} + \mathbf{b}_k$ and $\boldsymbol{f}^p \colon \mathbf{x} \mapsto \mathbf{W}_p^T \mathbf{x} + \mathbf{b}_p$, where $\mathbf{W}_k, \mathbf{W}_p \in \mathbb{R}^{d' \times d}$ and $\mathbf{b}_k, \mathbf{b}_p \in \mathbb{R}^{d'}$. Since $\boldsymbol{f}$ is continuous, any $\mathbf{x} \in \Omega_k \cap \Omega_p$ satisfies that $(\mathbf{W}_k - \mathbf{W}_p)^T \mathbf{x} + (\mathbf{b}_k - \mathbf{b}_p) = 0$. The set of all boundary points of $\boldsymbol{f}$ is therefore included in $\cup_{1 \leq k < p \leq \rho} H_{kp}$, where $H_{kp} = \{\mathbf{x} \in \mathbb{R}^d \colon (\mathbf{W}_k - \mathbf{W}_p)^T \mathbf{x} + (\mathbf{b}_k - \mathbf{b}_p) = 0\}$ is an affine subspace of dimension at most $(d-1)$ since $k \neq p$. The arrangement of the $\rho(\rho-1)/2$ hyperplanes H_{kq} with $k \neq p \in [\rho]$ yields convex regions on which $\boldsymbol{f}$ is affine since these regions do not contain boundary points. The number of such regions is, therefore, an upper bound on the number of convex regions of $\boldsymbol{f}$. It is known from [208] that the number of convex regions formed by an arrangement of N hyperplanes in $\mathbb{R}^d$ is at most $\sum_{k=0}^{\min(d,N)} \binom{N}{k}$. Hence, for $\rho(\rho-1)/2 > d$, we directly reach the announced result. Otherwise, the bound yields $\sum_{k=0}^{\rho(\rho-1)/2} \binom{\rho(\rho-1)/2}{k} = 2^{\rho(\rho-1)/2}$. □

Proof. **of Lemma 4.1** Let $e = \dim(E)$. The natural candidate for Π_E is the partition

$$\Pi' = \left\{ P' \colon P' = P \cap E, P \in \Pi, \text{ and } \mathrm{Int} P' \neq \emptyset \right\}, \tag{4.44}$$

which is unfortunately not necessarily a proper convex partition. Indeed, if E contains an e-face of a region, then some elements of Π' will not have disjoint interiors. Since the regions of Π are polyhedrons, there exist a given number n_H of distinct boundary hyperplanes $H_p = \{\mathbf{x} \in \mathbb{R}^d \colon \mathbf{a}_p^T \mathbf{x} + b_p = 0\}$ and such that for each $P_k \in \Pi$, there exists a subset $I_k \subset [n_H]$ and $\epsilon_{k,p} \in \{-1, 1\}$ for $p \in I_k$ such that

$$P_k = \{\mathbf{x} \in \mathbb{R}^d \colon \epsilon_{k,p}(\mathbf{a}_p^T \mathbf{x} + b_p) \geq 0 \quad \forall p \in I_k\}. \tag{4.45}$$

We now consider a mapping ϕ that assigns to each hyperplane H_p a unique region $\phi(p)$ such that $p \in I_{\phi(p)}$. We can now define n new pairwise-disjoint convex regions as

$$P'_k = \left\{ \mathbf{x} \in \mathbb{R}^d \colon \begin{array}{ll} \epsilon_{k,p}(\mathbf{a}_p^T \mathbf{x} + b_p) \geq 0 & \text{for } p \in [n_k] \text{ and } \phi(p) = k \\ \epsilon_{k,p}(\mathbf{a}_p^T \mathbf{x} + b_p) > 0 & \text{for } p \in [n_k] \text{ and } \phi(p) \neq k \end{array} \right\}. \tag{4.46}$$

It is clear that $\cup_{k=1}^n P'_k = \mathbb{R}^d$. From these new regions, one can eventually build the proper convex partition

$$\Pi_E = \left\{ P_E = \overline{P'_k \cap E} \colon k \in [n] \text{ and } \mathrm{Int} P_E \neq \emptyset \right\}. \tag{4.47}$$

By construction, all regions of Π_E are closed with nonempty interiors; their union covers

E. Let $P_{E,1} = P'_{k_1} \cap E$ and $P_{E,2} = P'_{k_2} \cap E$ be two (nonempty) regions of Π_E. We have that $\text{Int}(P_{E,1}) \cap \text{Int}(P_{E,2}) = \text{Int}(P_{E,1} \cap P_{E,2}) = \text{Int}(P'_{k_1} \cap P'_{k_2} \cap E) = \emptyset$ for $k_1 \neq k_2$. We, therefore, proved that Π_E is a convex partition of E; it has at most n regions and is such that, for any $P_E \in \Pi_E$, there is $P \in \Pi$ with $P_E \subset P$. □

Proof. **of Lemma 4.2** Let $P \in \Pi$. Recall that P is a closed and convex subset of the affine space $\boldsymbol{f}(\mathbb{R}^d)$ with dimension $\dim(\boldsymbol{f}(\mathbb{R}^d))$. We first prove that $\boldsymbol{f}^{-1}(P)$ meets the requirements to form a convex partition of $\mathbb{R}^d$.

- The continuity of $\boldsymbol{f}$ implies that $\boldsymbol{f}^{-1}(P)$ is closed.
- The function $\boldsymbol{f}$ is written as $\boldsymbol{f}\colon \mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{b}$ with $\mathbf{A} \in \mathbb{R}^{d' \times d}$ and $\mathbf{b} \in \mathbb{R}^{d'}$. For $\mathbf{x}, \mathbf{y} \in \boldsymbol{f}^{-1}(P)$ and $\beta \in [0,1]$, we have that $\boldsymbol{f}(\beta\mathbf{x}+(1-\beta)\mathbf{y}) = \mathbf{A}(\beta\mathbf{x}+(1-\beta)\mathbf{y})+\mathbf{b} = \beta\boldsymbol{f}(\mathbf{x}) + (1-\beta)\boldsymbol{f}(\mathbf{y}) \in P$ since P is convex. Therefore, $\boldsymbol{f}^{-1}(P)$ is also convex.
- We have that $\cup_{P\in\Pi}\boldsymbol{f}^{-1}(P) = \boldsymbol{f}^{-1}(\cup_{P\in\Pi}P) = \boldsymbol{f}^{-1}(\boldsymbol{f}(\mathbb{R}^d)) = \mathbb{R}^d$.
- For two distinct regions $P_1, P_2 \in \Pi$, we have that $\boldsymbol{f}^{-1}(P_1) \cap \boldsymbol{f}^{-1}(P_2) = \boldsymbol{f}^{-1}(P_1 \cap P_2)$. Since P_1 and P_2 are distinct regions of Π, $\dim(P_1 \cap P_2) < \dim(\boldsymbol{f}(\mathbb{R}^d))$, which implies that $\dim(\boldsymbol{f}^{-1}(P_1) \cap \boldsymbol{f}^{-1}(P_2)) < d$ and proves that P_1 and P_2 have disjoint interiors.
- We decompose the input space as the direct sum $\mathbb{R}^d = \ker(\mathbf{A}) \oplus U$. Note that $\boldsymbol{f}(U) = \boldsymbol{f}(\mathbb{R}^d)$. It is clear that, for any $\mathbf{x} \in \boldsymbol{f}^{-1}(P)$ and $\mathbf{y} \in \ker(\mathbf{A})$, we have that $\mathbf{x}+\mathbf{y} \in \boldsymbol{f}^{-1}(P)$, which implies that $\dim(\text{Proj}_U(\boldsymbol{f}^{-1}(P))) = \ker(\mathbf{A})$. In addition, we use the fact that $\boldsymbol{f}$ restricted to U has full rank and write $\dim(\text{Proj}_{\ker(\mathbf{A})}(\boldsymbol{f}^{-1}(P))) = \dim(P) = \dim(\boldsymbol{f}(\mathbb{R}^d))$. All in all, we have proved that $\dim(\boldsymbol{f}^{-1}(P)) = d$, which implies that the regions of $\boldsymbol{f}^{-1}(\Pi)$ have nonempty interiors.

□

Proof. **of Lemma 4.3** The result stems from the fact that the rank of a product of matrices is bounded by the smallest rank of these matrices. □

4.6.2 Upper Bound on the Number of Regions of Arrangements

Proof. **of Theorem 4.1** First, we prove that the expression given in the theorem is an upper bound. To that end, we need to formalize our problem with the notion of abstract simplicial complex so as to focus solely on the combinatorial structure of the task and be compliant with the formalism of [234]. Let $\Pi_k^* = \{\text{int}(P)\colon P \in \Pi_k\}$, where $\text{int}(P)$ denotes the interior of P in $\mathbb{R}^d$, and let $\mathcal{F} = \cup_{k=1}^N \{P^*\colon P^* \in \Pi_k^*\}$ be the set that contains the elements of the N sets Π_k^*. The the nerve K of $\mathcal{F}$ is defined as

$$\text{K} = \{X \subset \mathcal{F}\colon \cap X \neq \emptyset\}. \tag{4.48}$$

In simple words, K is made of all the nonempty intersections of sets in any of the Π_k^*. The nerve of an open covering is an abstract simplicial complex which, therefore, applies to K since $\mathcal{F}$ is an open covering of $\mathbb{R}^d$. This more simply follows from the definition of an abstract simplicial complex: it is a family of sets that is closed under taking subsets. In the sequel, we need K to be a d-representable simplicial complex, which is granted because it is the nerve of a finite family of convex sets in $\mathbb{R}^d$ (more details in [234]). In our problem, the faces of dimension 0 of the complex, also known as vertices, are the elements of $\mathcal{F}$. More generally, a face of K of dimension p is a nonempty intersection of $p+1$ elements of $\mathcal{F}$. Each set Π_k^* induces a sub-complex $\mathrm{K}[\Pi_k^*] = \{X \subset \Pi_k^* \colon \cap X \neq \emptyset\}$ of K. The dimension of this sub-complex, which is the largest dimension of its faces, is 0 because the elements of Π_k^* are disjoint. We note that the interior of the regions of the arrangement of the convex partitions are $(N+1)$-faces of the abstract simplicial complex K, which are also called $\mathbf{1}$-colorful faces, where $\mathbf{1} = (1, \dots, 1) \in \mathbb{R}^N$ specifies that each region of the arrangement is built from one region per partition. We are therefore looking to bound the number $f_{\mathbf{1}}(\mathrm{K})$ of $\mathbf{1}$-colorful faces of the complex K. Since we have now fully translated our problem into the framework of [234], we can apply [234, Theorem 10] to $\mathcal{F}$. The parameter $\mathbf{r} = (r_1, \dots, r_N)$ can be chosen so that $\dim(\mathrm{K}[\Pi_k^*]) \leq (r_k - 1)$. Therefore, we simply choose $\mathbf{r} = \mathbf{1}$ and obtain that

$$\beta^d(n_1, \dots, n_N) = f_{\mathbf{1}}(\mathrm{K}) \leq p_{\mathbf{1}}(\mathbf{n}, d, \mathbf{1}), \tag{4.49}$$

where

$$p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}) = \sum_{\ell=(\ell_1,\dots,\ell_N) \in L_{\mathbf{k}}(d)} \prod_{i=1}^{N} \binom{n_i - r_i}{\ell_i} \binom{r_i}{k_i - \ell_i} \tag{4.50}$$

and

$$L_{\mathbf{k}}(d) = \{\ell = (\ell_1, \dots \ell_N) \in \mathbb{N}^N \colon \ell_1 + \cdots + \ell_N \leq d \text{ and } \ell_i \leq k_i \text{ for } i \in [N]\}. \tag{4.51}$$

In our problem, $\mathbf{k} = \mathbf{1}$ and

$$L_{\mathbf{1}}(d) = \{\ell = (\ell_1, \dots \ell_N) \in \mathbb{N}^N \colon \ell_1 + \cdots + \ell_N \leq d \text{ and } \ell_i \in \{0, 1\} \text{ for } i \in [N]\}. \tag{4.52}$$

With $\mathbf{r} = \mathbf{1}$, we have that

$$\begin{aligned}
\beta^d(n_1, \dots, n_N) \leq p_{\mathbf{1}}(\mathbf{n}, d, \mathbf{1}) &= \sum_{\ell=(\ell_1,\dots,\ell_N)\in L_{\mathbf{1}}(d)} \prod_{i=1}^{N} \binom{n_i - 1}{\ell_i} \binom{1}{1-\ell_i} \\
&= \sum_{\ell=(\ell_1,\dots,\ell_N)\in L_{\mathbf{1}}(d)} \prod_{i=1}^{N} \binom{n_i - 1}{\ell_i} \\
&= \sum_{k=0}^{d} \sum_{\substack{\ell_1,\dots,\ell_N\in\{0,1\} \\ \ell_1+\cdots+\ell_N=k}} \prod_{i=1}^{N} \binom{n_i - 1}{\ell_i} \\
&= \sum_{k=0}^{d} \sum_{\substack{\ell_1,\dots,\ell_N\in\{0,1\} \\ \ell_1+\cdots+\ell_N=k}} \prod_{\substack{i=1 \\ \ell_i=1}}^{N} \binom{n_i - 1}{\ell_i} \prod_{\substack{i=1 \\ \ell_i=0}}^{N} \binom{n_i - r_i}{\ell_i} \\
&= \sum_{k=0}^{d} \sum_{\substack{\ell_1,\dots,\ell_N\in\{0,1\} \\ \ell_1+\cdots+\ell_N=k}} \prod_{\substack{i=1 \\ \ell_i=1}}^{N} (n_i - 1) \\
&= 1 + \sum_{k=1}^{d} \sum_{1\leq\ell_1<\cdots<\ell_k\leq N} \prod_{i=1}^{k} (n_{\ell_i} - 1), \qquad (4.53)
\end{aligned}$$

which proves that the bound given in the Theorem holds true.

Now we show that this upper bound is sharp. To that end, consider that each partition Π_k is made of the regions of the arrangement of the $(n_k - 1)$ distinct parallel hyperplanes H_q^k for $q = 1, \dots, (n_k - 1)$ so that the hyperplanes are in general position when only one per partition is selected. Recall that N hyperplanes are in general position if any collection of k of them intersect in a $(d-k)$-dimensional plane for $1 \leq k \leq d$ and have empty intersection for $k > d$. The number of regions of the arrangement $\mathcal{A}(\Pi_1, \dots, \Pi_N)$ is exactly the number of regions of the arrangement of all the hyperplanes H_q^k for $q = 1, \dots, (n_k - 1)$ and $k = 1, \dots, N$. Following Zavalasky's Theorem, the number of regions can be computed by

$$\#\mathcal{R}(\mathcal{A}) = (-1)^d \chi_{\mathcal{A}}(-1), \qquad (4.54)$$

where $\chi_{\mathcal{A}}$ is the characteristic polynomial of the arrangement. There is no need here to define the characteristic polynomial in detail since Whitney's formula provides a direct way to evaluate it as

$$\chi_{\mathcal{A}}(-1) = \sum_{\substack{\mathcal{B}\subset\mathcal{A} \\ \cap_{H\in\mathcal{B}} H\neq\emptyset}} (-1)^{\#\mathcal{B}} (-1)^{\dim(\cap_{H\in\mathcal{B}} H)}. \qquad (4.55)$$

To further explicitize the formula, we note that the subsets $\mathcal{B} \subset \mathcal{A}$ that have a nonempty intersection can be written as $\mathcal{B} = \{H_{q_{k_1}}^{k_1}, \dots, H_{q_{k_p}}^{k_p}\}$ with $1 \leq k_1 < \cdots < k_p \leq N$, $q_{k_i} \in [n_{k_i} - 1]$ where $i = 1, \dots, p$ and $0 \leq p \leq d$. This holds because, for $q \neq q'$,

$H_q^k \cap H_{q'}^k = \emptyset$. Note that, by convention, the set $\mathcal{B} = \emptyset$ is also considered in the sum. Because of the particular choice of the hyperplanes, for a given p, $\mathcal{B}$ is the nonempty intersection of p hyperplanes and there are $\sum_{1\leq \ell_1 < \cdots < \ell_p \leq N} \prod_{i=1}^{p} (n_{\ell_i} - 1)$ such subsets of $\mathcal{A}$. The intersection of the elements of $\mathcal{B}$ has dimension $(d-p)$ (recall that the hyperplanes of $\mathcal{B}$ are in general position). All in all, we have that

$$\begin{aligned}
\#\mathcal{R}(\mathcal{A}) &= (-1)^d \sum_{\substack{\mathcal{B}\subset\mathcal{A}:\\ \cap_{H\in\mathcal{B}} H \neq \emptyset}} (-1)^{\#\mathcal{B}} (-1)^{\dim(\cap_{H\in\mathcal{B}} H)} \\
&= 1 + (-1)^d \sum_{k=1}^{d} \sum_{1\leq \ell_1 < \cdots < \ell_k \leq N} (-1)^k (-1)^{d-k} \prod_{i=1}^{k} (n_{\ell_i} - 1) \\
&= 1 + \sum_{k=1}^{d} \sum_{1\leq \ell_1 < \cdots < \ell_k \leq N} \prod_{i=1}^{k} (n_{\ell_i} - 1), \qquad (4.56)
\end{aligned}$$

which is the upper bound given in the theorem.

When $N \leq d$, we readily check that the bound is giving $n_1 \cdots n_N$. To prove the second additional bound for $N > d$ given in the theorem, we invoke the binomial theorem and remark that

$$\begin{aligned}
\left(1 + \sum_{p=1}^{N} (n_p - 1)\right)^d &= 1 + \sum_{k=1}^{d} \binom{d}{k} \left(\sum_{p=1}^{N} (n_p - 1)\right)^k \\
&= 1 + \sum_{k=1}^{d} \binom{d}{k} \sum_{1\leq \ell_1, \ldots, \ell_k \leq N} \prod_{i=1}^{k} (n_{\ell_i} - 1) \qquad (4.57) \\
&\geq 1 + \sum_{k=1}^{d} \sum_{1\leq \ell_1, \ldots, \ell_k \leq N} \prod_{i=1}^{k} (n_{\ell_i} - 1) \\
&\geq 1 + \sum_{k=1}^{d} \sum_{1\leq \ell_1 < \cdots < \ell_k \leq N} \prod_{i=1}^{k} (n_{\ell_i} - 1). \qquad (4.58)
\end{aligned}$$

□

4.6.3 Sum and Vectorization

Proof. **of Proposition 4.2** Let Π_k be a linear convex partition of f_k for $k = 1, \ldots, N$. On each region of the arrangement $\mathcal{A}(\Pi_1, \ldots, \Pi_N)$, the f_k are affine, and so is their sum and their vectorization. This implies that $\mathcal{A}(\Pi_1, \ldots, \Pi_N)$ is a linear convex partition of both the sum and the vectorization of the scalar-valued CPWL functions, which shows that $\beta^d(\kappa_1, \ldots, \kappa_N)$ is a valid upper bound on the number of convex linear regions.

We now prove that the bounds are sharp. First, consider N convex partitions Π_k where

each Π_k is made of the regions of the arrangement of $(\kappa_k - 1)$ distinct parallel hyperplanes $H_k^p = \{\mathbf{x} \in \mathbb{R}^d \colon \mathbf{w}_k^T \mathbf{x} = b_k^p\}$, $p = 1, \dots, (\kappa_k - 1)$, and such that the hyperplanes are in general position when only one per partition is selected. In such a way, the arrangement $\mathcal{A}(\Pi_1, \dots, \Pi_N)$ has exactly $\beta^d(\kappa_1, \dots, \kappa_N)$ convex regions (see proof of Theorem 4.1). Second, for each partition, we consider a CPWL function $\varphi_k \colon \mathbb{R} \to \mathbb{R}$ with knots $(b_k^p)_{p=1}^{\kappa_k - 1}$ and κ_k distinct affine pieces $(\varphi_k^p)_{p=1}^{\kappa_k}$. In the sequel, the affine pieces are written $\varphi_k^p \colon x \to a_k^p x + c_k^p$. The function $f_k \colon \mathbf{x} \mapsto \varphi(\mathbf{w}_k^T \mathbf{x})$ has exactly n_k linear convex regions and Π_k is a linear convex partition of it. The construction implies that $\mathcal{A}(\Pi_1, \dots, \Pi_N)$ is a linear convex partition of both $(f_1 + \cdots + f_N)$ and $(f_1, \dots, f_N)$. Because the affine pieces of each φ_k are distinct, the vector-valued function $(f_1, \dots, f_N)$ will agree with distinct affine pieces on each region of $\mathcal{A}(\Pi_1, \dots, \Pi_N)$, which proves that this partition has the minimal number of linear convex regions. This yields CPWL functions such that $\kappa_{(f_1, \dots, f_N)} = \beta^d(\kappa_1, \dots, \kappa_N)$. On the contrary, in the case of the sum $(f_1 + \cdots + f_N)$, there is no guarantee that $\mathcal{A}(\Pi_1, \dots, \Pi_N)$ is a partition with the minimal number of linear convex regions. To ensure that the regions of this partition have different affine pieces, it is sufficient to choose the pieces (φ_k^p) such that $\sum_{k=1}^N \varphi_k^{p_k^1}(\mathbf{w}_k^T \cdot) \neq \sum_{k=1}^N \varphi_k^{p_k^2}(\mathbf{w}_k^T \cdot)$ for any $1 \leq p_k^1, p_k^2 \leq \kappa_k$ and $(p_1^1, \dots, p_N^1) \neq (p_1^2, \dots, p_N^2)$. An explicit choice is $a_k^p = p m^{k-1}$ with $m = \max(\kappa_k)$. The biases b_k^p are then set such that φ_k^p is continuous. In such a way, the slope of $\sum_{k=1}^N \varphi_k^{p_k}(\mathbf{w}_k^T \cdot)$ is $\sum_{k=1}^N p_k m^k$. This number can be represented in base m as "$(p_N \cdots p_1)_m$", which shows that it is uniquely related to the choice of indices (p_k). Although this choice seems very specific, a random choice of the slopes would also satisfy the condition almost surely. We have therefore found a collection of CPWL functions whose sum has exactly $\beta^d(\kappa_1, \dots, \kappa_N)$ linear convex regions. □

4.6.4 Compositional Bounds

Proof. **of Theorem 4.2** We use the notation $m_\ell = \min(d_1, \dots, d_\ell)$ and $\boldsymbol{F}_\ell = \boldsymbol{f}_\ell \circ \cdots \circ \boldsymbol{f}_1$. First, we prove by induction the validity of the proposed upper bound. The initial step is given by Proposition 4.2. Now suppose that the result holds for $\boldsymbol{F}_{\ell-1}$ with $\ell - 1 > 0$. Let Ω be a linear convex region of $\boldsymbol{F}_{\ell-1}$ and let $\boldsymbol{g}_\Omega$ be the corresponding affine function. The affine space $\boldsymbol{g}_\Omega(\mathbb{R}^{d_1}) \subset \mathbb{R}^{d_\ell}$ is of dimension at most $\min(d_1, \cdots, d_\ell)$ (Lemma 4.3). Each linear convex partition $\Pi_{\ell,k}$ of the components of $\boldsymbol{f}_\ell$ yields a convex partition $\Pi'_{\ell,k}$ of the affine subspace $\boldsymbol{g}_\Omega(\mathbb{R}^{d_1})$ with no more than $\kappa_{\ell,k}$ regions on which $f_{\ell,k}$ is affine (Lemma 4.1). The arrangement of the partitions $\Pi'_{\ell,1}, \dots, \Pi'_{\ell,d_{\ell+1}}$ results in a convex partition of $\boldsymbol{g}_\Omega(\mathbb{R}^{d_1})$ with no more than $\beta^{m_\ell}(\kappa_{\ell,1}, \dots, \kappa_{\ell,d_\ell+1})$ regions (Theorem 4.1). Lemma 4.2 shows that $\boldsymbol{g}_\Omega^{-1}(\mathcal{A}(\Pi'_{\ell,1}, \dots, \Pi'_{\ell,d_{\ell+1}}))$ is a convex partition of $\mathbb{R}^{d_1}$ with $\boldsymbol{F}_\ell$ affine on each of its sets. In short, each linear convex region of $\boldsymbol{F}_{\ell-1}$ is partitioned into no more than $\beta^{m_\ell}(\kappa_{\ell,1}, \dots, \kappa_{\ell,d_{\ell+1}})$ linear convex regions, which concludes the first part of the proof.

Second, we propose a construction inspired from [214] to derive the lower bound given on the maximal number of regions. Let the sawtooth function sw_p of order p be the unique 1D CPWL function with knots located at k/p for $k = 1, \dots, (p-1)$ that satisfies

$\mathrm{sw}_p(k/p) = \frac{1}{2}(1-(-1)^k)$ for $k = 0, \dots, p$. The key properties of the sawtooth function of order p that will prove useful in the sequel are

- it has p projection regions that are also convex linear regions;
- it can be decomposed as

$$\mathrm{sw}_p \colon x \mapsto \sum_{k=1}^{p} \varphi_{k,p}, \tag{4.59}$$

  where $\varphi_{k,p} = p(x + 2(-1)^p |x - k/p|)$ is a CPWL function with 2 projection regions;
- the composition of sawtooth functions is a sawtooth function whose order is the product of the orders of the composed functions, as in

$$\mathrm{sw}_p \circ \mathrm{sw}_q = \mathrm{sw}_{pq}, \tag{4.60}$$

  for $p, q \in \mathbb{N}$.

The strategy is now to build a CPWL NN which mimics a given NN with independent sawtooth components. Let $\mathbf{e}_{d,k}$ be the kth element of the canonical basis of $\mathbb{R}^d$, $d^* = \min(d_\ell)$ and $\tau_\ell \colon \{1, \dots, d_{\ell+1}\} \to \{1, \dots, d^*\}$ for $\ell = 1, \dots, L$. Consider the dimension-reduction linear operator $\boldsymbol{u}_\ell \colon \mathbb{R}^{d_\ell} \to \mathbb{R}^{d^*}$ associated to $\tau_{\ell-1}$, which is defined on the canonical basis by

$$\boldsymbol{u}_\ell \colon \mathbf{e}_{d_\ell,k} \mapsto \mathbf{e}_{d^*,\tau_{\ell-1}(k)}, \tag{4.61}$$

for $\ell = 2, \dots, L$ and

$$\boldsymbol{u}_1(\mathbf{e}_{d_\ell,k}) = \begin{cases} e_{d^*,k}, & k \le d^* \\ 0, & \text{otherwise.} \end{cases} \tag{4.62}$$

Similarly, let $\boldsymbol{v}_\ell \colon \mathbb{R}^{d^*} \to \mathbb{R}^{d_{\ell+1}}$ be the dimension-augmentation linear operator

$$\boldsymbol{v}_\ell \colon \mathbf{e}_{d^*,k} \mapsto \sum_{q \in \tau_\ell^{-1}(\{k\})} \mathbf{e}_{d_{\ell+1},q}, \tag{4.63}$$

for $\ell = 1, \dots, L$.

We now define the nonlinear pointwise function $\boldsymbol{\phi}_\ell \colon \mathbb{R}^{d_{\ell+1}} \to \mathbb{R}^{d_{\ell+1}}$. For $r \in \{1, \dots, d^*\}$ let $p_{\ell,r} = \sum_{k \in \tau_\ell^{-1}(\{r\})} \kappa_{\ell,k}$ and $\{J_{\ell,\tau_\ell,i}\}_{i=1}^{|\tau_\ell^{-1}(\{r\})|}$ be a partition of the set $\{1, \dots, p_{\ell,r}\}$, where the cardinality of the subsets is in one-to-one correspondence with $\{\kappa_{\ell,q}\}_{q \in \tau_\ell^{-1}(\{r\})}$. In this way, we assign to each $k \in \{1, \dots, d_\ell + 1\}$ a set of indices J_{ℓ,τ_ℓ,i_k} that allows us to define the kth component of $\boldsymbol{\phi}_\ell$ as

$$\phi_{\ell,k} = \sum_{j \in J_{\ell,\tau_\ell,i_k}} \varphi_{|\tau_\ell^{-1}(\{\tau_\ell(k)\})|,j}. \tag{4.64}$$

This ensures that

$$\sum_{k\in\tau_\ell^{-1}(\{r\})} \phi_{\ell,k} = \mathrm{sw}_{p_{\ell,r}}. \tag{4.65}$$

From the pointwise property of $\boldsymbol{\phi}$, we deduce that, for any $t_1,\dots,t_{d^*}\in\mathbb{R}$,

$$\begin{aligned}
(\boldsymbol{u}_{\ell+1}\circ\boldsymbol{\phi}\circ\boldsymbol{v}_\ell)\left(\sum_{r=1}^{d^*} t_r\mathbf{e}_{d^*,r}\right) &= (\boldsymbol{u}_{\ell+1}\circ\boldsymbol{\phi})\left(\sum_{r=1}^{d^*}\sum_{k\in\tau_\ell^{-1}(\{r\})} t_r\mathbf{e}_{d_\ell,k}\right)\\
&= \boldsymbol{u}_{\ell+1}\left(\sum_{r=1}^{d^*}\sum_{k\in\tau_\ell^{-1}(\{r\})} \phi_{\ell,k}(t_r)\mathbf{e}_{d_\ell,k}\right)\\
&= \sum_{r=1}^{d^*}\sum_{k\in\tau_\ell^{-1}(\{r\})} \phi_{\ell,k}(t_r)\boldsymbol{u}_{\ell+1}(\mathbf{e}_{d_\ell,k})\\
&= \sum_{r=1}^{d^*}\sum_{k\in\tau_\ell^{-1}(\{r\})} \phi_{\ell,k}(t_r)\mathbf{e}_{d^*,\tau_\ell(k)}\\
&= \sum_{r=1}^{d^*}\left(\sum_{k\in\tau_\ell^{-1}(\{r\})} \phi_{\ell,k}(t_r)\right)\mathbf{e}_{d^*,r}\\
&= \sum_{r=1}^{d^*} \mathrm{sw}_{p_{\ell,r}}(t_r)\mathbf{e}_{d^*,r},
\end{aligned} \tag{4.66}$$

which means that $\boldsymbol{u}_{\ell+1}\circ\boldsymbol{\phi}\circ\boldsymbol{v}_\ell$ is a pointwise multivariate function with 1D sawtooth components of order $p_{\ell,r}$ for $r=1,\dots,d^*$. We denote it by $\mathbf{sw}_{\mathbf{p}_\ell}$ with $\mathbf{p}_\ell=(p_{\ell,1},\dots,p_{\ell,d^*})$. The function $\boldsymbol{f}_\ell$ of the NN is chosen to be $\boldsymbol{f}_\ell=\boldsymbol{\phi}_\ell\circ\boldsymbol{v}_\ell\circ\boldsymbol{u}_\ell$. Each component $f_{\ell,k}$ can be written in the form of $f_{\ell,k}\colon\mathbf{x}\mapsto\phi_{\ell,k}(\mathbf{w}_{\ell,k}^T\mathbf{x})$ with $\mathbf{w}_{\ell,k}=\sum_{q\in\tau_{\ell-1}^{-1}(\{\tau_\ell(k)\})}\mathbf{e}_{d_\ell,q}$. This shows that $f_{\ell,k}$ has the same number of projection regions as $\phi_{\ell,k}$ ($\kappa_{\ell,k}$) whenever $\mathbf{w}_{\ell,k}\neq\mathbf{0}$.

All in all, we have that

$$\begin{aligned}
&\boldsymbol{f}_L\circ\boldsymbol{f}_{L-1}\circ\cdots\circ\boldsymbol{f}_2\circ\boldsymbol{f}_1\\
&=(\boldsymbol{\phi}_L\circ\boldsymbol{v}_L\circ\boldsymbol{u}_L)\circ(\boldsymbol{\phi}_{L-1}\circ\boldsymbol{v}_{L-1}\circ\boldsymbol{u}_{L-1})\circ\cdots\circ(\boldsymbol{\phi}_2\circ\boldsymbol{v}_2\circ\boldsymbol{u}_2)\circ(\boldsymbol{\phi}_1\circ\boldsymbol{v}_1\circ\boldsymbol{u}_1) &(4.67)\\
&=\boldsymbol{\phi}_L\circ\boldsymbol{v}_L\circ(\boldsymbol{u}_L\circ\boldsymbol{\phi}_{L-1}\circ\boldsymbol{v}_{L-1})\circ(\boldsymbol{u}_{L-1}\circ\cdots\circ\boldsymbol{\phi}_2\circ\boldsymbol{v}_2)\circ(\boldsymbol{u}_2\circ\boldsymbol{\phi}_1\circ\boldsymbol{v}_1)\circ\boldsymbol{u}_1)\\
&=\boldsymbol{\phi}_L\circ\boldsymbol{v}_L\circ\mathbf{sw}_{\mathbf{p}_{L-1}}\circ\cdots\circ\mathbf{sw}_{\mathbf{p}_1}\circ\boldsymbol{u}_1. &(4.68)
\end{aligned}$$

We now note that there are no fewer projection regions of $\boldsymbol{f}_L\circ\boldsymbol{f}_{L-1}\circ\cdots\circ\boldsymbol{f}_2\circ\boldsymbol{f}_1$ than the number of projection regions of $\boldsymbol{h}=\boldsymbol{u}_{L+1}\circ\boldsymbol{f}_L\circ\cdots\circ\boldsymbol{f}_1$ because $\boldsymbol{u}_{L+1}$ is a linear

mapping. In addition,

$$\begin{aligned}\boldsymbol{u}_{L+1} \circ \boldsymbol{f}_L \circ \cdots \circ \boldsymbol{f}_1 &= \mathbf{sw}_{\mathbf{p}_L} \circ \cdots \circ \mathbf{sw}_{\mathbf{p}_1} \circ \boldsymbol{u}_1 \\ &= \mathbf{sw}_{\mathbf{q}} \circ \boldsymbol{u}_1, \end{aligned} \tag{4.69}$$

where $\mathbf{q} = (q_1, \ldots, q_{d^*})$ and $q_r = \prod_{\ell=1}^{L} p_{\ell,r}$. The properties of the sawtooth functions and the special form of $\boldsymbol{u}_1$ yields the projection regions for $\boldsymbol{h}$ as

$$\{\mathbf{x} \in \mathbb{R}^{d_1} \colon \text{for } r = 1, \ldots, d^*, \begin{cases} -\infty < x_r \leq 1/q_r, & i_r = 0 \\ 1 - 1/q_r \leq x_r < +\infty, & i_r = q_r - 1\}, \\ i_r/q_r \leq x_r \leq (i_r+1)/q_r, & \text{otherwise} \end{cases} \tag{4.70}$$

where $i_r = 0, \ldots, (q_r - 1)$ for $r = 1, \ldots, d^*$. In summary, the number of projection regions of the constructed CPWL NN is at least

$$\prod_{r=1}^{d^*} q_r = \prod_{\ell=1}^{L} \prod_{r=1}^{d^*} p_{\ell,r} = \prod_{\ell=1}^{L} \prod_{r=1}^{d^*} \sum_{k \in \tau_\ell^{-1}(\{r\})} \kappa_{\ell,k}. \tag{4.71}$$

The conclusion is reached by noticing that the reasoning does not depend on any property of the mappings τ_ℓ: one can therefore pick the ones that yield the largest lower bound.

□

Proof. **of Corollary 4.1** To get the upper bound, we combine Theorem 4.2 and the simplified version of the bound given in Theorem 4.1 with the assumption that $W \geq d_{\text{in}} \geq d^* \coloneqq \min(d_{\text{in}}, d_{\text{out}})$. To get the lower bound, we have to compute

$$\alpha^{d^*}(\kappa_{\ell,1}, \ldots, \kappa_{\ell,d_{\ell+1}}) = \max_{\tau \in \mathcal{T}_{d_\ell}} \prod_{r=1}^{d^*} \sum_{k \in \tau^{-1}(\{r\})} \kappa_{\ell,k}. \tag{4.72}$$

We lower-bound this quantity by selecting an arbitrary mapping $\tau \colon [W] \to [d^*]$ such that, for $r \in [d^*]$, the cardinality of $\tau^{-1}(\{r\})$ is at least $\lfloor W/d^* \rfloor$. In this way, we obtain that

$$\alpha^{d^*}(\kappa_{\ell,1}, \ldots, \kappa_{\ell,d_{\ell+1}}) \geq (\kappa \lfloor W/d^* \rfloor)^{d^*} \tag{4.73}$$

for $\ell = 1, \ldots, L$ and reach the given lower bound. □

Proof. **of Corollary 4.2** The upper bound is a direct consequence of Proposition 4.1: the number of convex linear regions is never smaller than the number of projection regions. The CPWL NN built to provide the lower bound of Theorem 4.2 had exactly as many projection regions as convex linear regions, hence justifying the lower bound. □

4.7 Appendix – Proofs for Section 4.4.2

Proof. **of Proposition 4.5** First, we prove the result when $\boldsymbol{\gamma}$ parameterizes a linear segment. Let $\mathbf{x}_0, \mathbf{u} \in \mathbb{R}^d$ with $\|\mathbf{u}\|_2 = 1$ and $\boldsymbol{\gamma}\colon t \mapsto \mathbf{x}_0 + t\mathbf{u}$ for $t \in S$, where $S = [0, |S|] \subset \mathbb{R}$ is a segment. The first step is to compute the probability $\mathbb{P}(\mathrm{kt}_f^{\boldsymbol{\gamma}} = 1)$ that f has a knot along $\boldsymbol{\gamma}$. The hyperplane $\{\mathbf{x} \in \mathbb{R}^d\colon \boldsymbol{w}^T\mathbf{x} + b = 0\}$ intersects the line $\{\mathbf{x}_0 + t\mathbf{u}\colon t \in \mathbb{R}\}$ for t_0 such that $\boldsymbol{w}^T(\mathbf{x}_0 + t_0\mathbf{u}) + b = 0$ or, equivalently, $b = (-\boldsymbol{w}^T(\mathbf{x}_0 + t_0\mathbf{u}))$. In order to have a knot along $\boldsymbol{\gamma}_{|S}$, t_0 has to lie in S. For a given $\boldsymbol{w}$, this implies that b should be in an interval of length $|S||\boldsymbol{w}^T\mathbf{u}|$, more precisely $[-\boldsymbol{w}^T\mathbf{x}_0, |S|\boldsymbol{w}^T\mathbf{u} - \boldsymbol{w}^T\mathbf{x}_0]$ if $\boldsymbol{w}^T\mathbf{u} < 0$ and $[|S|\boldsymbol{w}^T\mathbf{u} - \boldsymbol{w}^T\mathbf{x}_0, -\boldsymbol{w}^T\mathbf{x}_0]$ otherwise. Therefore, $\mathbb{P}(\mathrm{kt}_f^{\boldsymbol{\gamma}} = 1|\boldsymbol{w}) \leq \sup_{t\in\mathbb{R}} \rho_b(t)|S||\boldsymbol{w}^T\mathbf{u}|$. From the independence of the random variables and from the fact that $\mathrm{kt}_f^{\boldsymbol{\gamma}} = 0$ or $\mathrm{kt}_f^{\boldsymbol{\gamma}} = 1$ almost surely, we infer that $\mathbb{E}[\lambda_f^{\boldsymbol{\gamma}_{|S}}] \leq \sup_{t\in\mathbb{R}} \rho_b(t)\mathbb{E}[|\boldsymbol{w}^T\mathbf{u}|] \leq \sup_{t\in\mathbb{R}} \rho_b(t)\sqrt{\mathbb{E}[|\boldsymbol{w}^T\mathbf{u}|^2]} = \sup_{t\in\mathbb{R}} \rho_b(t)\sqrt{\mathbf{u}^T\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T]\mathbf{u}} \leq \sup_{t\in\mathbb{R}} \rho_b(t)\sqrt{\mathbb{E}[w^2]}$. In the last step, the assumption that the random variables w_k are i.i.d. has allowed us to infer that $\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T] = \mathbb{E}[w^2]\mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is the identity matrix.

If b is normally distributed with standard deviation σ_b, then $\sup_{t\in\mathbb{R}} \rho_b(t) = (\sigma_b\sqrt{2\pi})^{-1}$. In addition, suppose that the components w_k are independent and normally distributed with standard deviation σ_w. The random variable $\boldsymbol{w}^T\mathbf{u}$ is also normally distributed with standard deviation σ_w (since $\|\mathbf{u}\|_2 = 1$). We can now compute explicitly $\mathbb{E}[|\boldsymbol{w}^T\mathbf{u}|] = \sigma_w\sqrt{2}/\sqrt{\pi}$ based on the properties of half-normal distributions.

The result is extended to any polygonal chain through the linearity of the expectation operator and by application of the result to the finitely many pieces of the polygonal chain. □

Proof. **of Proposition 4.6** A knot of f along a line $\boldsymbol{\gamma}$ must lie on a hyperplane $H_{p,q} = \{\mathbf{x}\colon (\boldsymbol{w}_p - \boldsymbol{w}_q)^T\mathbf{x} + (b_p - b_q) = 0\}$ with $1 \leq p < q \leq K$, since elsewhere the Maxout unit is affine. Therefore, the expected knot density is bounded as

$$\begin{aligned}
\mathbb{E}\left[\lambda_{\mathrm{Maxout}}^{\boldsymbol{\gamma}}\right] &\leq \frac{1}{|S|}\mathbb{E}\left[\sum_{1\leq p<q\leq K} (\boldsymbol{\gamma}(S) \cap H_{p,q} \neq \emptyset)\right] \\
&= \frac{1}{|S|}\sum_{1\leq p<q\leq K} \mathbb{E}\left[(\boldsymbol{\gamma}(S) \cap H_{p,q} \neq \emptyset)\right] \\
&\leq \frac{1}{|S|}\sum_{1\leq p<q\leq K} \sqrt{\mathbb{E}\left[(w_p - w_q)^2\right]} \sup_{t\in\mathbb{R}} \rho_b(t) \\
&= \frac{1}{|S|}\sum_{1\leq p<q\leq K} \sqrt{2}\sigma_w \sup_{t\in\mathbb{R}} \rho_b(t) \\
&= \frac{1}{|S|}\sqrt{2}\binom{K}{2}\sigma_w \sup_{t\in\mathbb{R}} \rho_b(t),
\end{aligned} \tag{4.74}$$

where we have taken advantage of the results derived in the proof of Proposition 4.5 to bound the probability that a randomly generated hyperplane intersects a segment of length $|S|$, along with the independence of the random variables. We also notate $(A \neq \emptyset)$ to encode the variable that takes the value 0 if $A = \emptyset$ and 1 otherwise. When the random variables are normally distributed, the reasoning is similar to the one in the proof of Proposition 4.5. □

Proof. **of Proposition 4.7** A knot of f along a line $\boldsymbol{\gamma}$ must lie on a hyperplane $H_{p,q} = \{\mathbf{x}\colon (\boldsymbol{w}_p - \boldsymbol{w}_q)^T\mathbf{x} + (b_p - b_q) = 0\}$, where $1 \leq p < q \leq K$ and where p, q belong to the same sorting group, since elsewhere the GroupSort layer is affine. One can now follow the same steps as those in the proof of Proposition 4.6 with $n_g\binom{g_s}{2} = n_g g_s(g_s - 1)/2 = d(g_s - 1)/2$ hyperplanes. □

5 Spline Activations for Stable and Expressive Deep Neural Networks

This chapter is concerned with theoretical considerations on Lipschitz-constrained neural networks (NNs). The implementation counterpart is then given in Chapter 6, which also provides a detailed introduction to plug-and-play methods for image reconstruction.

The text of this chapter is adapted from the published article

(A. Goujon*, S. Neumayer*), P. Bohra and M. Unser "Approximation of Lipschitz functions using deep spline neural networks", *SIAM Journal on Mathematics of Data Science*, volume 5, page 306-322, issue 2, June 2023,

and also contains results taken from

S. Ducotterd, A. Goujon, P. Bohra, D. Perdios, S. Neumayer and M. Unser "Improving Lipschitz-Constrained Neural Networks by Learning Activation Functions", submitted to *Journal of Machine Learning Research*.

Summary

Although many applications in machine learning require Lipschitz-constrained models, the design and training of expressive Lipschitz-constrained networks is very challenging. Since the popular rectified linear-unit networks have provable disadvantages in this setting, we propose to use learnable spline activation functions with at least 3 linear regions instead. We prove that our choice is universal among all component-wise 1-Lipschitz activation functions in the sense that no other weight-constrained architecture can approximate a larger class of functions. Additionally, we prove that learnable linear splines have provable advantages over the recently introduced GroupSort and Householder activation functions under 2-norm Lipschitz constraints, which corresponds to the plug-and-play setup explored in Chapter 6.

*equal contribution

5.1 Introduction

Lipschitz-constrained neural networks (NNs) have proven to be useful in several areas of machine learning, for instance, to obtain robustness guarantees in classification [184, 186, 239], to train Wasserstein generative adversarial networks (GANs) [79, 88], to learn normalization flows with invertible NNs [86, 87] and to improve the generalization abilities of NNs [180, 181, 240]. In the context of image reconstruction, Lipschitz-constrained NN denoisers are also widely used within plug-and-play algorithms to provide stability and convergence guarantees [81, 83, 89, 241–243]. This specific application is discussed at length in Chapter 6. The design and training of Lipschitz-constrained NNs is unfortunately very challenging since the computation of the Lipschitz constant of NNs is related to NP-hard problems [48] and not feasible in practice.

Regularization techniques offer a first way to bypass the exact computation of the Lipschitz constant [75, 76, 78, 88, 183, 244]. In practice, the training loss is augmented with a penalty term that promotes NNs with lower Lipschitz constants. For instance, one can penalize the norm of the Jacobian of the NN at well-chosen locations [75, 76]. While regularization improves stability and maintains good performance, it does not offer any theoretical guarantees on the Lipschitz constant.

Theoretical guarantees can be obtained by controlling upper bounds on the Lipschitz constant. A simple upper bound is given by the product of the Lipschitz constant of each layer, but it is usually a very coarse estimate. There exist more precise estimators based on semidefinite programming [245, 246], adversarial training [244, 247], or the derivation of sharper estimates for the composition of layers [48]. Unfortunately, these methods are either computationally expensive or do not provide a proper upper bound. A common strategy in practice is therefore to design the model architecture so that the fast-to-evaluate bounds become sharper. A general overview of NN architectures and, in particular, Lipschitz-constrained ones can be found in [248]. The most common approach toward Lipschitz-constrained architectures, referred to as the layer-wise approach, is to control the norm of each linear layer, typically with the spectral or other p-norms [80, 161, 210], or by enforcing orthogonality of the weight matrices [83, 249, 250]. In combination with 1-Lipschitz activations, this results in architectures with a Lipschitz constant bounded by the product of the norms of the weights. However, this estimate is in general quite pessimistic, especially for deep models. Consequently, this additional structural constraint often leads to vanishing gradients [251] and a seriously reduced expressivity of the model. Remarkably, the commonly used rectified linear-unit (ReLU) activation aggravates the situation. For instance, it is shown in [252] that ReLU NNs with ∞-norm weight constraints have a second-order total variation that is bounded independently of the depth. In addition, it is proven in [82] that, under spectral norm constraints, any scalar-valued ReLU NN Φ with $\|\boldsymbol{\nabla}\Phi\|_2 = 1$ a.e. is necessarily linear. To circumvent the described issues, several new activation functions have been proposed recently, such as GroupSort [82] or the related Householder [84] activation functions. Note

that, contrary to ReLU, all these activation functions are multivariate. Analyzing the expressivity of the resulting NNs and determining their applicability in practice is an active area of research.

It is by no means trivial to specify which class of functions can be approximated by a generic NN with 1-Lipschitz layers. Ideally, given a compact set $D \subset \mathbb{R}^d$ equipped with the p-norm, it is desirable to approximate all scalar-valued 1-Lipschitz functions, which are denoted by $\mathrm{Lip}_{1,p}(D)$. The first result in this direction was provided in [82], where the authors show that the use of the GroupSort activation function and ∞-norm-constrained weights indeed allows the universal approximation of $\mathrm{Lip}_{1,p}(D)$. The behavior of such NNs was then further investigated in [253, 254]. Unfortunately, the proof strategies published so far cannot be generalized to other norms. For instance, little is known for the 2-norm setting which is the one relevant to many applications, including plug-and-play methods for image reconstruction [81]. Therefore, being able to compare the approximation capabilities of different architectures is an important first step. For example, the approximation of the absolute value function, for which an exact representation with ReLU is impossible, provides a classic benchmark to compare architectures. From a practical perspective, GroupSort NNs have yielded promising results and compare favorably against ReLU NNs with similar architectures [82].

Currently, the most substantial results regarding layer-wise Lipschitz constraints rely on multivariate activation functions. Although the ReLU activation function is indeed too limiting, we claim that the class of component-wise activation functions ought not to be dismissed off-hand. Following this idea, we analyze deep spline NNs, whose activation functions are learnable linear splines (LLS) [216, 236, 255]. Since bounds on the Lipschitz constant of compositions are usually too pessimistic, our rationale is to increase the expressivity of the activation function while still being able to efficiently control its Lipschitz constant.

Outline and Contributions In Section 5.2, we revisit 1-Lipschitz continuous piecewise-linear (CPWL) functions and we show that they can approximate any function in $\mathrm{Lip}_{1,p}(D)$. Since the construction of 1-Lipschitz NNs is nontrivial, we then briefly discuss the layer-wise Lipschitz-constrained NNs along with the popular activation functions used under Lipschitz constraints. Then, in Section 5.3, we extend some known results on the limitations of weight-constrained NNs with ReLU activation functions. More precisely, we show that ReLU-like NNs cannot represent certain simple functions for any p-norm weight constraint. Based on a second-order total variation argument, we further show that they cannot be universal approximators for ∞-norm weight constraints. We also prove that linear spline activation functions with 2 linear regions have a limited representation power. Next, in Section 5.4, we propose a representer theorem to show the optimality of learnable linear splines for layer-wise Lipschitz-constrained NNs. We also prove that LLS-NNs with 3 linear-region splines achieve the maximum expressivity among NNs with

component-wise activation functions. Finally, we discuss the relation between LLS-NNs and GroupSort NNs and conclude in Section 5.6.

5.2 Layer-wise Lipschitz-Constrained Neural Networks

5.2.1 Preliminaries

A function $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^n$ is said to be L-Lipschitz if

$$\|\boldsymbol{f}(\mathbf{x}) - \boldsymbol{f}(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \tag{5.1}$$

for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, where the norm $\|\cdot\|$ in the input and output spaces shall be specified when necessary. The Lipschitz constant $\mathrm{Lip}(\boldsymbol{f})$ of $\boldsymbol{f}$ is the smallest constant L such that $\boldsymbol{f}$ is L-Lipschitz, and it is denoted by $\mathrm{Lip}_p(\boldsymbol{f})$ when $\|\cdot\|$ is chosen to be the p-norm $\|\cdot\|_p$ for both the input and output space.

In this chapter, we investigate feedforward NN architectures that consist of $K \in \mathbb{N}$ layers with widths $n_1, \ldots, n_K$ that are given by mappings $\boldsymbol{\Phi} \colon \mathbb{R}^d \to \mathbb{R}^{n_K}$ of the form

$$\boldsymbol{\Phi}(\mathbf{x}) \coloneqq \boldsymbol{A}_K \circ \boldsymbol{\sigma}_{K-1,\boldsymbol{\alpha}_{K-1}} \circ \boldsymbol{A}_{K-1} \circ \boldsymbol{\sigma}_{K-2,\boldsymbol{\alpha}_{K-2}} \circ \cdots \circ \boldsymbol{\sigma}_{1,\boldsymbol{\alpha}_1} \circ \boldsymbol{A}_1(\mathbf{x}). \tag{5.2}$$

Here, the affine functions $\boldsymbol{A}_k \colon \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}$ are given by

$$\boldsymbol{A}_k(\mathbf{x}) \coloneqq \mathbf{W}_k\mathbf{x} + \mathbf{b}_k, \qquad k = 1, \ldots, K, \tag{5.3}$$

with weight matrices $\mathbf{W}_k \in \mathbb{R}^{n_k, n_{k-1}}$, $n_0 = d$ and bias vectors $\mathbf{b}_k \in \mathbb{R}^{n_k}$. For multilayer perceptrons, $\mathbf{W}_k$ is learned as a full matrix, while for convolutional NNs, $\mathbf{W}_k$ is parametrized via a convolution operator whose kernel is learned. The model includes parameterized nonlinear activation functions $\boldsymbol{\sigma}_{k,\boldsymbol{\alpha}_k} \colon \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ with corresponding parameters $\boldsymbol{\alpha}_k$, $k = 1, \ldots, K-1$. For the case of component-wise activation functions, we have that $\boldsymbol{\sigma}_{k,\boldsymbol{\alpha}_k}(\mathbf{x}) = (\sigma_{k,\boldsymbol{\alpha}_k,j}(x_j))_{j=1}^{n_k}$. We sometimes drop the index k in the activation function $\sigma_{k,\boldsymbol{\alpha}_k}$ to simplify the notation. The complete parameter set of the NN is denoted by $\boldsymbol{\theta} \coloneqq (\mathbf{W}_k, \mathbf{b}_k, \boldsymbol{\alpha}_k)_{k=1}^K$ and the NN by $\boldsymbol{\Phi}(\cdot, \boldsymbol{\theta})$ whenever the dependence on the parameters is explicitly needed. For an illustration, see Figure 5.1. Architecture (5.2) results in a CPWL function whenever the activation functions themselves are CPWL functions such as the ReLU. Next, we investigate the approximation properties of this architecture under Lipschitz constraints on $\boldsymbol{\Phi}(\cdot, \boldsymbol{\theta})$.

5.2.2 Universality of 1-Lipschitz ReLU Networks

First, we briefly revisit the approximation of Lipschitz functions by CPWL functions, for which we give a precise definition.

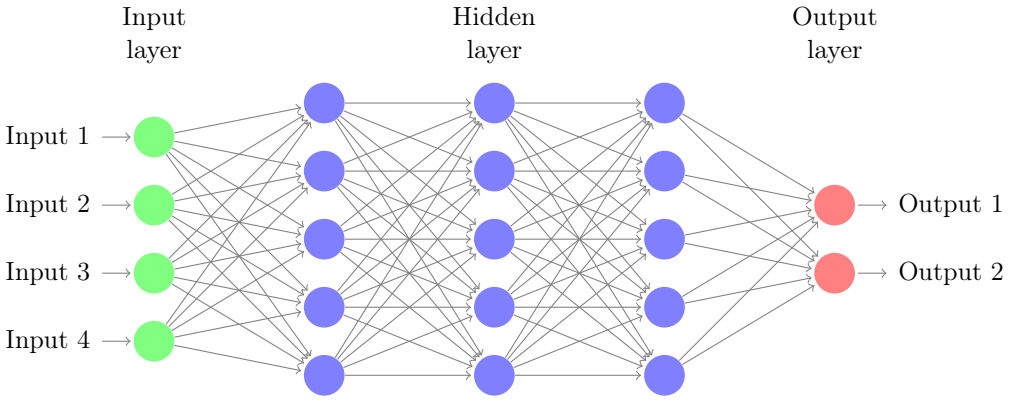

Figure 5.1: Feedforward NN with three hidden layers, where $d = 4$, $K = 4$, $n_1 = n_2 = n_3 = 5, n_4 = 2$.

Definition 5.1. *A continuous function $\boldsymbol{f}\colon \mathbb{R}^d \to \mathbb{R}^n$ is called continuous and piecewise linear if there exist a finite set $\{\boldsymbol{f}^m\colon m = 1, \dots, M\}$ of affine functions, also called affine pieces, and closed sets $(\Omega_m)_{m=1}^M \subset \mathbb{R}^d$ with nonempty and pairwise-disjoint interiors, also called projection regions [222], such that $\cup_{m=1}^M \Omega_m = \mathbb{R}^d$ and $\boldsymbol{f}_{|\Omega_m} = \boldsymbol{f}^m_{|\Omega_m}$.*

Assume that we are given a collection of tuples $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, $i = 1, \dots, N$, which can be interpreted as samples from a function $f\colon \mathbb{R}^d \to \mathbb{R}$. Let

$$L^p_{x,y} \coloneqq \max_{i \neq j} \frac{|y_i - y_j|}{\|\mathbf{x}_i - \mathbf{x}_j\|_p} \tag{5.4}$$

denote the Lipschitz constant associated with these points. Then, a first natural question is whether it is always possible to find an interpolating CPWL function g with p-norm Lipschitz constant $\mathrm{Lip}_p(g) = L^p_{x,y}$.

Proposition 5.1. *For the tuples $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, $i = 1, \dots, N$ and $p \in [1, +\infty]$, there exists a CPWL function f with $\mathrm{Lip}_p(f) = L^p_{x,y}$ such that $g(\mathbf{x}_i) = y_i$ for all $i = 1, \dots, N$.*

Since we are unaware of proof, we provide one below.

Proof. Let q be such that $1/p + 1/q = 1$. For $p < +\infty$, define $\mathbf{u}_{i,j} \in \mathbb{R}^d$ as the vector given by

$$(\mathbf{u}_{i,j})_k = \mathrm{sgn}((\mathbf{x}_i - \mathbf{x}_j)_k)|(\mathbf{x}_i - \mathbf{x}_j)_k|^{p/q}. \tag{5.5}$$

If $p = +\infty$, we choose k_0 with $\|\mathbf{x}_i - \mathbf{x}_j\|_\infty = |(\mathbf{x}_i - \mathbf{x}_j)_{k_0}|$, and define $(\mathbf{u}_{i,j})_{k_0} = \mathrm{sgn}(\mathbf{x}_i - \mathbf{x}_j)_{k_0}$ with all other components of $\mathbf{u}_{i,j}$ set to 0. This saturates Hölder's inequality with

$$\langle \mathbf{u}_{i,j}, \mathbf{x}_j - \mathbf{x}_i \rangle = \sum_{k=1}^d |(\mathbf{u}_{i,j})_k (\mathbf{x}_j - \mathbf{x}_i)_k| = \|\mathbf{u}_{i,j}\|_q \|\mathbf{x}_j - \mathbf{x}_i\|_p, \tag{5.6}$$

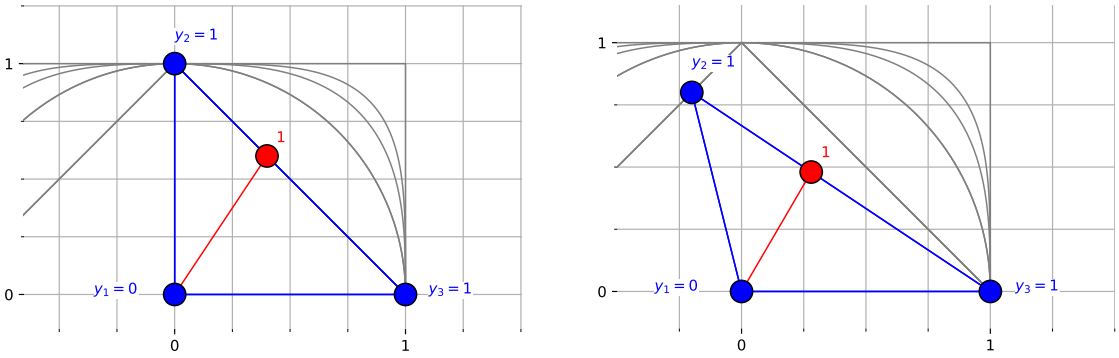

Figure 5.2: Interpolation based on a Triangulation: Let $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathbb{R}^2$ be input data points (blue dots) with corresponding target values $y_1 = 0$, $y_2 = 1$ and $y_3 = 1$. The gray curves depict the ℓ_p unit balls for $p \in \{1, 2, 3, 4, +\infty\}$. For the left plot, we set $p > 1$ and get $L^p_{x,y} = 1$. On the right, we set $p = 1$ and also get $L^p_{x,y} = 1$. The unique affine function $g\colon \mathbb{R}^2 \to \mathbb{R}$ interpolating the data is the simplest CPWL function that fits the data. On any point $\mathbf{x}$ lying between $\mathbf{x}_2$ and $\mathbf{x}_3$ (red dot), it holds that $g(\mathbf{x}) = 1$, hence $|g(\mathbf{x}_1) - g(\mathbf{x})| = 1$. However, in both settings, x is in the unit ball for the according p which implies that $\|\mathbf{x}_1 - \mathbf{x}\|_p < 1$. Hence, $\mathrm{Lip}_p(g) > L^p_{x,y}$ and g does not interpolate the data with the minimal Lipschitz constant.

where we used that $\mathbf{u}_{i,j}$ and $(\mathbf{x}_j - \mathbf{x}_i)$ have components with the same sign. For $i \neq j$, we define the linear function

$$f_{i,j}(x) = y_i + \frac{y_j - y_i}{\|\mathbf{x}_j - \mathbf{x}_i\|_p \|\mathbf{u}_{i,j}\|_q} \langle \mathbf{u}_{i,j}, \mathbf{x} - \mathbf{x}_i \rangle, \tag{5.7}$$

which is such that $f_{i,j}(\mathbf{x}_i) = y_i$ and $\mathrm{Lip}_p(f_{i,j}) = |y_j - y_i| / \|\mathbf{x}_j - \mathbf{x}_i\|_p$, as $\sup_{\|\mathbf{x}\|_p \leq 1} \langle \mathbf{u}_{i,j}, \mathbf{x} \rangle = \|\mathbf{u}_{i,j}\|_q$. Next, set $f_i(\mathbf{x}) = \max_{j, j \neq i} f_{i,j}(\mathbf{x})$, for which it holds that $f_i(\mathbf{x}_i) = y_i$ and $\mathrm{Lip}_p(f_i) = \max_j |y_j - y_i| / \|\mathbf{x}_j - \mathbf{x}_i\|_p$. Then, we define $f(\mathbf{x}) = \min_i f_i(\mathbf{x})$ and directly obtain that $f(\mathbf{x}_j) \leq y_j$ for any $j = 1, \ldots, N$. However, we also have that

$$f_i(\mathbf{x}_j) \geq f_{i,j}(\mathbf{x}_j) = y_i + y_j - y_i = y_j, \tag{5.8}$$

which then implies that $f(\mathbf{x}_j) = y_j$ for any $j = 1, \ldots, N$. Further, we directly get that $\mathrm{Lip}_p(f) = L^p_{x,y}$. Finally, by recalling that the maximum and the minimum of any number of CPWL functions is CPWL as well [222], we conclude that f is CPWL and the claim follows. □

Remark 5.1. *The d-dimensional construction is more involved than the 1D case, for which a simple interpolation is sufficient. A natural way to fit the data in any dimension is to form a triangulation with vertices $(\mathbf{x}_i)_{i=1}^N$. Then, with the use of the CPWL hat basis functions of the triangulation, one can directly form an interpolating CPWL function. Unfortunately, the Lipschitz constant of this function can exceed $L^p_{x,y}$. An example of this issue is provided in Figure 5.2.*

Since the maximum and minimum of finitely many affine functions can be represented by ReLU NNs, the same holds true for the CPWL function constructed in Proposition 5.1.

This directly leads us to a well-known corollary, Corollary 5.1. Note that this Corollary can also be proven by noticing that 1-Lipschitz ReLU NNs can represent any 1-Lipschitz affine function, and then by applying [82, Lemma 1].

Corollary 5.1. *Let $D \subset \mathbb{R}^d$ be compact and $p \in [1, +\infty]$. Then, the ReLU NNs $\Phi \colon D \to \mathbb{R}$ with $\mathrm{Lip}_p(\Phi) \leq 1$ are dense in $\mathrm{Lip}_{1,p}(D)$.*

Since computing the Lipschitz constant of a generic NN is NP-hard, Corollary 5.1 has limited practical relevance. To circumvent this issue, one can either design algorithms that provide tight estimates, or design special architectures for which simple upper bounds are not too pessimistic. In this chapter, we pursue the second direction. To this end, we introduce tools to build Lipschitz-constrained architectures in the remainder of this section and investigate the universality of these architectures in Section 5.4.

5.2.3 Layer-wise 1-Lipschitz NNs

The Lipschitz constant is known to be sub-multiplicative for the composition operation and thus the Lipschitz constant $\mathrm{Lip}(\boldsymbol{\Phi})$ of the NN (5.2) can be bounded as in

$$\mathrm{Lip}(\boldsymbol{\Phi}) \leq \mathrm{Lip}(\boldsymbol{A}_K) \prod_{k=1}^{K-1} \mathrm{Lip}(\boldsymbol{\sigma}_{k,\boldsymbol{\alpha}_k}) \mathrm{Lip}(\boldsymbol{A}_k). \tag{5.9}$$

Layer-wise 1-Lipschitz NN A layer-wise 1-Lipschitz (LW 1-Lip) NN is defined as a NN with 1-Lipschitz layers. Following (5.9), LW 1-Lip NNs are 1-Lipschitz. The use of LW 1-Lip NNs is currently the most popular way to build 1-Lip NNs. The bound (5.9) being not necessarily sharp, the design of the linear layers and activation function of the NN requires special care to ensure good performance. Note that while we are mainly interested in 1-Lip NNs, one can similarly build LW L-Lipschitz NNs and most results for LW 1-Lip NN directly generalize to any $L > 0$.

5.2.4 1-Lip Linear Layers

A first step toward Lipschitz-constrained NNs is to constrain the norm of the weights.

Operator-Norm Constraints The $p \to q$ operator norm is given for $\mathbf{W} \in \mathbb{R}^{n,m}$ and $p, q \in [1, +\infty]$ by

$$\|\mathbf{W}\|_{p,q} \coloneqq \max_{\mathbf{x} \in \mathbb{R}^m, \|\mathbf{x}\|_p = 1} \|\mathbf{W}\mathbf{x}\|_q, \tag{5.10}$$

and $\|\cdot\|_p \coloneqq \|\cdot\|_{p,p}$. Note that $\|\cdot\|_1$ and $\|\cdot\|_\infty$ correspond to the maximum ℓ_1 norm of the columns and rows of $\mathbf{W}$, respectively. The norm $\|\cdot\|_2$, also known as the spectral

norm, corresponds to the largest singular value of $\mathbf{W}$. To obtain a non-expansive NN of the form (5.2) in the p-norm sense, the weight matrices can be constrained as

$$\|\mathbf{W}_k\|_p \leq 1, \quad k = 1, \dots, K, \tag{5.11}$$

which we shall henceforth refer to as p-norm-constrained weights, since $\mathrm{Lip}_p(\boldsymbol{A}_k) = \|\mathbf{W}_k\|_p$. For matrices $\mathbf{W} \in \mathbb{R}^{1,n}$ it holds that $\|\mathbf{W}\|_p = \|\mathbf{W}^T\|_q$ with $1/p + 1/q = 1$. In other words, if we interpret these matrices as vectors, then we have to constrain the q-norm instead. In the case of scalar-valued NNs, we can also constrain the weights as $\|\mathbf{W}_k\|_q \leq 1$, $k = 2, \dots, K$, and $\|\mathbf{W}_1\|_{p,q} \leq 1$, since all standard norms are identical in $\mathbb{R}$. There exist several methods to enforce such constraints in the training stage [80, 161, 210], see Remark 5.2 for more details.

Orthonormality Constraints Instead of imposing $\|\mathbf{W}\|_2 \leq 1$, we can also require that either $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ or $\mathbf{W}\mathbf{W}^T = \mathbf{I}$, depending on the shape of $\mathbf{W}$. This constraint corresponds to imposing that either $\mathbf{W}$ or $\mathbf{W}^T$ lie in the so-called Stiefel manifold. Compared to the spectral-norm constraint, the orthonormality constraint enforces all singular values of $\mathbf{W}$ to be unity. From a computational perspective, this approach is more challenging than the previous one but helps to mitigate the problem of vanishing gradients in deep NNs. For more details, including possible implementations, we refer to [83, 249–251, 256].

Remark 5.2. *Many of the implementations for the schemes of Section 5.2.3 enforce the p-norm constraint or orthonormality only approximately. For theoretical guarantees, it is however necessary to ensure that the constraint is satisfied exactly. In practice, this means that sufficient numerical accuracy or additional postprocessing after training might be necessary.*

5.2.5 1-Lip Activation Functions

While the quest for optimal activation functions in the last decade leaves us with many choices, the 1-Lipschitz constraint is the game-changer and the relevance of each activation function must be reassessed. In Section 5.3, we provide results that explain why the ReLU activation function is not suited for LW Lipschitz-constrained NNs. Hence, we need to resort to other activation functions that lead to increased expressivity. In this context, the gradient norm preserving (GNP) property is desirable: the Jacobian of a GNP mapping has all eigenvalues equal to one a.e. In the following, we briefly discuss activation functions with promising theoretical and/or empirical results in constrained settings, both element-wise and multivariate ones.

GroupSort (GS) The sort operation takes a vector of dimension n and simply outputs its components sorted in ascending order. This operation has complexity $\mathcal{O}(n\log(n))$, which is slightly worse than the linear complexity of component-wise activation functions. The GS activation function [82] is a generalization of this operation: it splits the pre-activation into groups of prescribed length and performs the sort operation within each group. This results in near-linear complexity when the group length is small enough. If the group length is two, then the activation function is known as the MaxMin or norm-preserving orthogonal-permutation linear unit [257]. Let us remark that any arbitrary GS activation function can be written as a composition of MaxMin activation functions, i.e., larger group lengths do not increase the theoretical expressivity. Although not obvious at first glance, the GS activation function is a CPWL operation. The rationale for this activation function is to perform a nonlinear and norm-preserving operation, which mitigates the issue of vanishing gradients in deep constrained architectures. More precisely, we have that the Jacobian of the GS activation function is a.e. given by a permutation matrix, which is indeed an orthogonal matrix, and hence GS is GNP.

Householder (HH) The GS activation was recently generalized with the Householder activation function [84] $\boldsymbol{\sigma}_{\mathbf{v}}\colon \mathbb{R}^d \to \mathbb{R}^d$ with $\mathbf{v} \in \mathbb{R}^d$, $\|\mathbf{v}\|_2 = 1$, given by

$$\boldsymbol{\sigma}_{\mathbf{v}}(\mathbf{z}) = \begin{cases} \mathbf{z} & \text{if } \mathbf{v}^T\mathbf{z} > 0, \\ (\mathbf{I} - 2\mathbf{v}\mathbf{v}^T)\mathbf{z} & \text{otherwise.} \end{cases} \tag{5.12}$$

On the hyperplane that separates the two cases (i.e., $\mathbf{v}^T\mathbf{z} = 0$) we have that $(\mathbf{I}-2\mathbf{v}\mathbf{v}^T)\mathbf{z} = \mathbf{z} - 2(\mathbf{v}^T\mathbf{z})\mathbf{v} = \mathbf{z}$. Thus, $\boldsymbol{\sigma}_{\mathbf{v}}$ is continuous and, moreover, the Jacobian is either $\mathbf{I}$ or $(\mathbf{I}-2\mathbf{v}\mathbf{v}^T)$, which are both square orthogonal matrices, and hence HH is GNP. For practical purposes, the authors of [84] recommend using groups of size 2. This construction can be iterated to obtain higher-order HH activation functions with more linear regions.

Absolute Value (AV) The absolute value activation function is component-wise, 1-Lipschitz, and arguably the simplest GNP activation.

Parametric ReLU (PReLU) The parametric ReLU activation function [173] is a learnable and component-wise activation that generalizes both ReLU and the AV activation. It is given by $\mathrm{PReLU}_{\boldsymbol{a}}(\mathbf{x}) = (\max(a_n x_n, x_n))_{n=1}^N$ with learnable parameters $(a_n)_{n=1}^N$. It holds that $\mathrm{Lip}(\mathrm{PReLU}_{\boldsymbol{a}}) = \max(\max_{1\leq n\leq N}|a_n|, 1)$. Hence, to be 1-Lip one must clip the parameters $(a_n)_{n=1}^N$ in $[-1, 1]$.

Learnable Linear Splines (LLS) The so-called deep spline NNs [216, 236, 258], also referred to as LLS-NNs, uses learnable component-wise linear-spline activation functions,

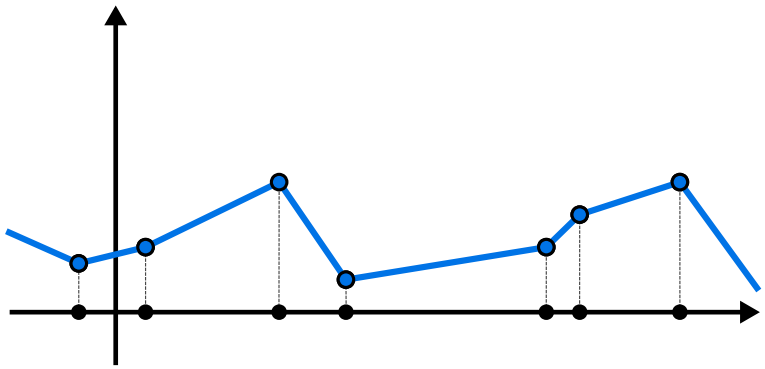

Figure 5.3: LLS_8: Linear spline with 7 knots (also known as breakpoints) and 8 linear regions.

see Figure 5.3 for an illustration. It is known that deep spline NNs are solutions to a functional optimization problem; namely, the training of a neural network with free-form activation functions whose second-order total-variation is regularized [216]. A linear-spline activation function is fully characterized by its linear regions and the corresponding values at the boundaries. In the unconstrained setting, any linear spline can be implemented by means of a scalar one-hidden-layer ReLU NN as

$$x \mapsto \sum_{m=1}^{M} u_m \text{ReLU}(v_m x + b_m), \tag{5.13}$$

where $u_m, v_m, b_m \in \mathbb{R}$ and $M \in \mathbb{N}$. This parameterization, however, lacks expressivity under p-norm constraints on the weights, as it is not able to produce linear splines with second-order total variation greater than 1, as discussed in Lemma 5.1 and Section 5.3.1. Instead, it is more convenient to rely on local B-spline atoms [236]. In practice, the linear-spline activation functions have a fixed number of uniformly spaced breakpoints—typically between 10 and 50—and are expressed as a weighted sum of cardinal B-splines. This amounts to adding a learnable parameter for each breakpoint and two additional ones to set the slope at both ends for a linear extrapolation. This local parameterization yields an evaluation complexity that remains independent of the number of breakpoints, in contrast, with (5.13). The B-spline framework can be adapted to learn 1-Lipschitz activation functions via the use of a suitable projector on the B-spline coefficients as proposed and further discussed in Chapter 6. In the sequel LLS_M refers to a LLS with no more than M linear regions.

Representation Power of Layer-wise 1-Lipschitz Neural Networks

All activation functions discussed in this chapter are CPWL, and hence the corresponding LW 1-Lip NNs parameterize a subset of the 1-Lip CPWL functions. We now introduce the representation sets

$$\mathcal{S}_p^{d,n}(\mathrm{X}) \tag{5.14}$$

of functions corresponding to a given activation function "X", which is defined as the $\mathbb{R}^d \to \mathbb{R}^n$ functions that can be expressed by a LW 1-Lip NN of the form (5.2) with p-norm constraints and 1-Lip activation of type "X".

One of the main goals of this chapter is to compare the following sets

$$\mathcal{S}_p^{d,n}(\text{ReLU}), \mathcal{S}_p^{d,n}(\text{AV}), \mathcal{S}_p^{d,n}(\text{PReLU}), \mathcal{S}_p^{d,n}(\text{LLS}_M), \mathcal{S}_p^{d,n}(\text{GS}), \text{ and } \mathcal{S}_p^{d,n}(\text{HH}).$$

While for unconstrained NNs all sets are equal to the set of CPWL functions, the situation is radically different with the LW Lipschitz constraints. In the sequel, we are mainly interested in the case $p = 2$, which is the most relevant for PnP methods (see Chapter 6), but whenever the result also holds for a generic $p \geq 1$, we state the more general result, and if the result is valid for any $d, n > 0$, we drop the superscripts and simply use the notation $\mathcal{S}_p(\text{X})$ for readability.

The following relations are either already known, or obvious:

- $\mathcal{S}_2(\text{ReLU})$ is included in and distinct from all the other sets of interest, see [82] and Section 5.3 for a further characterization;
- $\mathcal{S}_2(\text{GS}) = \mathcal{S}_2(\text{HH})$;
- $\mathcal{S}_2(\text{GS}) \subseteq \mathcal{S}_2(\text{AV})$, the reverse is also true on compact sets [82];

At this point, we know that

$$\mathcal{S}_2(\text{ReLU}) \subsetneq \mathcal{S}_2(\text{GS}) = \mathcal{S}_2(\text{HH}) \subseteq \mathcal{S}_2(\text{AV}) \subseteq \mathcal{S}_2(\text{PReLU}) \subseteq \mathcal{S}_2(\text{LLS}_2) \subseteq \mathcal{S}_2(\text{LLS}_{M>2}). \tag{5.15}$$

The goal is now to determine which inclusions are strict, and which ones are equalities.

5.3 2 Linear Regions are not Enough

In this section, we discuss the representation power of LW 1-Lip NN with 2-linear region spline activations (LLS_2), which includes ReLU, AV, PReLU and indirectly GS and HH. We first provide results that explain why an activation should have more than one region with slope one. We then show that LLS with 2 linear regions also have some limited representation power compared to LLS with more linear regions.

5.3.1 Limitations of ReLU

Limited Representation Power

Component-wise and monotone activation functions are detrimental to the expressivity of NNs with spectral-norm-constrained weights [82, Theorem 1]. Here, we generalize this result to NNs with p-norm-constrained weights and certain CPWL activation functions, along with a more precise characterization. In particular, we also cover the case where $\|J\Phi\|_p$ is not 1 a.e.

Proposition 5.2. *Let $p \in (1, +\infty]$, let $I \subset \mathbb{R}$ be a closed interval, and let $\sigma\colon \mathbb{R} \to \mathbb{R}$ be a CPWL activation function satisfying*

- $\sigma(x) = x + b$, $b \in \mathbb{R}$, *for* $x \in I$,
- $|\sigma'(x)| < 1$ *for* $x \notin I$.

Then, any LW 1-Lip NN $\Phi\colon \mathbb{R}^d \to \mathbb{R}$ of the form (5.2) *with p-norm-constrained weights and component-wise activation function of the form σ, has at most one affine region Ω_i with $\|\boldsymbol{J}\{\Phi\}|_{\Omega_i}\|_p = 1$, where $\boldsymbol{J}\{\Phi\}|_{\Omega_i}$ denotes the Jacobian of Φ over Ω_i.*

Proof. We proceed via induction over the number K of layers of Φ. For $K = 1$, the mapping is affine and the statement holds trivially. Now, assume that the result holds for some $K > 1$. Let

$$\Phi_{K+1} = \boldsymbol{A}_{K+1} \circ \boldsymbol{\sigma} \circ \boldsymbol{A}_K \circ \cdots \circ \boldsymbol{\sigma} \circ \boldsymbol{A}_1, \tag{5.16}$$

which we decompose as $\Phi_{K+1} = \Phi_K \circ \boldsymbol{h}$ with $\Phi_K = \boldsymbol{A}_{K+1} \circ \boldsymbol{\sigma} \circ \boldsymbol{A}_K \circ \cdots \circ \boldsymbol{\sigma} \circ \boldsymbol{A}_2$ and $\boldsymbol{h} = \boldsymbol{\sigma} \circ \boldsymbol{A}_1$. The induction assumption implies that $\|\boldsymbol{J}\{\Phi_K\}\|_p < 1$ on all affine regions except possibly one. The corresponding affine function $f_K^1\colon \mathbb{R}^{n_1} \to \mathbb{R}$ with projection region $\Omega_K \subset \mathbb{R}^{n_1}$ takes the form $\mathbf{x} \mapsto \mathbf{v}^T\mathbf{x} + c$, where $\mathbf{v} \in \mathbb{R}^{n_1}$ is such that $\|\mathbf{v}\|_q \leq 1$, $1/p + 1/q = 1$, and $c \in \mathbb{R}$. Now, we define the set

$$\Omega_{K+1} = \{\mathbf{x} \in \mathbb{R}^d : (\boldsymbol{A}_1(\mathbf{x}))_l \in I \text{ for any } l \text{ s.t. } \mathbf{v}_l \neq 0\} \cap \boldsymbol{h}^{-1}(\Omega_K). \tag{5.17}$$

By construction, Φ_{K+1} is affine on Ω_{K+1} and coincides with $\Phi_K \circ (\boldsymbol{A}_1 + b)$ on this set. Any other affine piece of Φ_{K+1} can be written in the form of $f_K^i \circ \boldsymbol{h}^j$, where f_K^i and $\boldsymbol{h}^j$ are affine pieces of Φ_K and $\boldsymbol{h}$, respectively. For this composition, either of the following holds.

i) It holds that $f_K^i \neq f_K^1$, which results in $\|\boldsymbol{J}\{f_K^i \circ \boldsymbol{h}^j\}\|_p < 1$ due to $\|\boldsymbol{J}\{f_K^i\}\|_p < 1$.

ii) It holds that $f_K^i = f_K^1$. Further, note that $\boldsymbol{J}\{h^j\} = \operatorname{diag}(\mathbf{d})\mathbf{W}_1$ for some $\mathbf{d} \in \mathbb{R}^{n_1}$ with entries $|\mathbf{d}_l| \leq 1$. Due to the definition of Ω_{K+1}, there exists l^* such that $\mathbf{v}_{l^*} \neq 0$ and $|\mathbf{d}_{l^*}| < 1$. Hence, the Jacobian of the affine piece is given by $\tilde{v}^T\mathbf{W}_1$

with $\tilde{\mathbf{v}} = \mathrm{diag}(\mathbf{d})\mathbf{v}$. Since $p \neq 1$, we get that $q < +\infty$ and $\|\tilde{\mathbf{v}}\|_q < \|\mathbf{v}\|_q \leq 1$. Consequently, $\|\boldsymbol{J}\{f_K^i \circ h^j\}\|_p = \|\tilde{\mathbf{v}}^T \mathbf{W}_1\|_p \leq \|\tilde{\mathbf{v}}\|_q \|\mathbf{W}_1\|_p < 1$.

This concludes the induction argument. □

For $p > 1$, Proposition 5.2 implies that ReLU NNs with p-norm constraints on the weights can reproduce neither the absolute value nor a whole family of simple functions, including the triangular hat function (also known as the B-spline of degree 1) and the soft-thresholding function. Further, this result suggests that activation functions with more than one linear region with maximal slope are better suited within the scope of this approximation framework. Typically, learnable spline activation functions are capable of having this property.

Limited Approximation Power

A meaningful metric for the expressivity of a model is its ability to produce functions with high variations. In this section, we investigate the impact of the Lipschitz constraint on the maximal second-order total variation of such a NN. Note that we partially rely on results from [252] for our proofs. The second-order total variation of a function $f : \mathbb{R} \to \mathbb{R}$ is defined as $\mathrm{TV}^{(2)}(f) := \|\mathrm{D}^2 f\|_{\mathcal{M}}$, where $\|\cdot\|_{\mathcal{M}}$ is the total-variation norm related to the space $\mathcal{M}$ of bounded Radon measures, and D is the distributional derivative operator. The space of functions with bounded second-order total variation is denoted by

$$\mathrm{BV}^{(2)}(\mathbb{R}) = \{f : \mathbb{R} \to \mathbb{R} \text{ s.t. } \mathrm{TV}^{(2)}(f) < +\infty\}. \tag{5.18}$$

For more details, we refer the reader to Section 5.7 and [216, 259]. Further, we recall that $\mathrm{TV}^{(2)}$ is a seminorm that, for a CPWL function on the real line, is given by the finite sum of its absolute slope changes. Based on Lemma 5.1, we infer for the p-norm-constrained setting that, in general, a linear-spline activation function cannot be replaced with a one-layer ReLU NN without losing expressivity.

Lemma 5.1. *Let $f : \mathbb{R} \to \mathbb{R}$ be parameterized by a one-hidden-layer NN with component-wise activation function σ and p-norm-constrained weights, $p \in [1, +\infty]$. If $\sigma \in \mathrm{BV}^{(2)}(\mathbb{R})$, then*

$$\mathrm{TV}^{(2)}(f) \leq \mathrm{TV}^{(2)}(\sigma). \tag{5.19}$$

Proof. Let f be a $\mathbb{R} \to \mathbb{R}$ function given by $x \mapsto \mathbf{u}^T \sigma(\mathbf{w}x + b) = \sum_{n=1}^N u_n \sigma(w_n x + b_n)$ with $\mathbf{u} := (u_1, \ldots, u_N) \in \mathbb{R}^N$, $\mathbf{w} := (w_1, \ldots, w_N) \in \mathbb{R}^N$ and $\mathbf{b} := (b_1, \ldots, b_N) \in \mathbb{R}^N$. The p-norm weight constraints imply that $\|\mathbf{w}\|_p \leq 1$ and $\|\mathbf{u}\|_q \leq 1$ with $1/p + 1/q = 1$. Since

$\mathrm{TV}^{(2)}$ is a semi-norm, we get

$$\mathrm{TV}^{(2)}(f) \leq \sum_{n=1}^{N} |u_n| \mathrm{TV}^{(2)}(\sigma(w_n \cdot + b_n)) \leq \sum_{n=1}^{N} |u_n w_n| \mathrm{TV}^{(2)}(\sigma) \leq \mathrm{TV}^{(2)}(\sigma), \tag{5.20}$$

where the last step follows from Hölder's inequality. □

In principle, the composition operation suffices to increase the second-order total variation of a mapping exponentially. For instance, the n-fold composition f_n of $f\colon \mathbb{R} \to \mathbb{R}$ with $x \mapsto 2|x - 1/2|$ yields the sawtooth function with 2^n linear regions and

$$\mathrm{TV}^{(2)}(f_n) = 2(2^n - 1). \tag{5.21}$$

This highly desirable property is, however, not achievable by ReLU NNs with ∞-norm-constrained weights [252, Theorem 1]. As shown in Proposition 5.3, this has a drastic impact on the size of the class of functions that can be approximated by ReLU NNs.

Proposition 5.3. *Let $D \subset \mathbb{R}^d$ be compact with a nonempty interior. Then, there exists $f \in \mathrm{Lip}_{1,\infty}(D)$ that cannot be approximated by ReLU NNs $\Phi\colon \mathbb{R}^d \to \mathbb{R}$ with Architecture* (5.2)*, and ∞-norm-constrained weights.*

Proof. By [252, Theorem 1], we know that, for any $\mathbf{u} \in \mathbb{R}^d$ with $\|\mathbf{u}\|_\infty = 1$ and any ReLU NN Φ with ∞-norm weight constraints, it holds that

$$\mathrm{TV}^{(2)}(\Phi \circ \boldsymbol{\varphi}_{\mathbf{u}}) \leq 2, \tag{5.22}$$

where $\boldsymbol{\varphi}_{\mathbf{u}}\colon \mathbb{R} \to \mathbb{R}^d$ with $t \mapsto t\mathbf{u}$. Let $(\Phi_n)_{n\in\mathbb{N}}$ be a sequence of ReLU NNs with ∞-norm-constrained weights that converges uniformly to Φ on D. Since D has nonempty interior, we can pick $\mathbf{u} \in \mathbb{R}^d$ with $\|u\|_\infty = 1$ such that $\boldsymbol{\varphi}_{\mathbf{u}}^{-1}(D)$ contains an open interval $I \subset \mathbb{R}$. Then, $(\Phi_n \circ \boldsymbol{\varphi}_{\mathbf{u}})_{n\in\mathbb{N}}$ converges uniformly to $\Phi \circ \boldsymbol{\varphi}_{\mathbf{u}}$ on I. Since $\mathrm{TV}^{(2)}$ is lower semicontinuous with respect to uniform convergence [259, Prop. 3.14], we infer that the restriction to I satisfies

$$\mathrm{TV}^{(2)}(\Phi \circ \boldsymbol{\varphi}_{\mathbf{u}}) \leq 2. \tag{5.23}$$

In other words, any $f \in \mathrm{Lip}_{1,\infty}(D)$ with $\mathrm{TV}^{(2)}(f \circ \boldsymbol{\varphi}_{\mathbf{u}}) > 2$ cannot be approximated by ∞-norm-constrained ReLU NNs. However, there exist sawtooth-like functions on I that have this property, with an explicit example constructed in Proposition 5.4. □

Unlike ReLU networks, deep spline networks can produce arbitrarily complex mappings thanks to the composition operation, even in the norm-constrained setting.

Proposition 5.4. *Let $C > 0$, $p \in [1, +\infty]$, $I \subset \mathbb{R}$ open and $\mathbf{u} \in \mathbb{R}^d$. Then, there exists a NN $\Phi\colon \mathbb{R}^d \to \mathbb{R}$ with Architecture* (5.2)*, p-norm-constrained weights, and 1-Lipschitz*

linear-spline activation functions with one knot such that, for $\boldsymbol{\varphi}_{\mathbf{u}}\colon I \to \mathbb{R}^d$ *with* $\varphi(t) = t\mathbf{u}$*, it holds that*

$$\mathrm{TV}^{(2)}(\Phi \circ \boldsymbol{\varphi}_{\mathbf{u}}) > C. \tag{5.24}$$

Proof. Pick $b \in \mathbb{R}$, $c > 0$ such that $[b-c, b+c] \subset I$. Let σ_1 with $x \mapsto (|x-b| - c/2)$, σ_k with $x \mapsto (|x| - c/2^k)$, $k = 2, \dots, m$, and $F_m = \sigma_m \circ \cdots \circ \sigma_1$. The function F_m is a sawtooth-like CPWL function with 2^m linear regions all contained in $[b-c, b+c]$. Further, it holds for all $t \in \mathbb{R}$ that $|F_m'(t)| = 1$, and the sign of the slope is different for neighboring regions. From this, we directly infer that

$$\mathrm{TV}^{(2)}(F_m) = 2(2^m - 1). \tag{5.25}$$

Now, we build a deep spline NN $\Phi_K\colon \mathbb{R}^d \to \mathbb{R}$ with K hidden layers of widths $n_1, \dots, n_K = d$ and $n_{K+1} = 1$. The activation function used in the k-th hidden layer is σ_k for the first neuron and zero otherwise, the weight matrices are chosen as the identity matrix except for the last layer, where it is chosen such that

$$\Phi_K(x) = F_K(x_1). \tag{5.26}$$

This construction results for $\boldsymbol{\varphi}_{\mathbf{e}_1}\colon I \to \mathbb{R}^d$ in

$$\mathrm{TV}^{(2)}(\Phi_K \circ \boldsymbol{\varphi}_{\mathbf{e}_1}) = 2(2^K - 1), \tag{5.27}$$

and the claim follows for $\mathbf{u} = \mathbf{e}_1$ by taking K sufficiently large. The general case $\mathbf{u} \neq \mathbf{e}_1$ follows by using an appropriate weight matrix in the first layer. □

5.3.2 Representation Power of 2-Linear Region Splines

We now focus on LW 1-Lip NN with any LLS_2 activation functions.

Proposition 5.5. *There exist* $\mathbb{R}^p \to \mathbb{R}^q$ *CPWL functions that can be represented by a LW 1-Lip NN with* 2*-norm-constrained weights and with linear spline activations with 3 linear regions but not by any LW 1-Lip NN with* 2*-norm-constrained weights and with linear-spline activations with 2 linear regions. In other words,*

$$\mathcal{S}_2(\mathrm{LLS}_2) \subsetneq \mathcal{S}_2(\mathrm{LLS}_{M>2}). \tag{5.28}$$

Proof. • **One-dimensional functions** $(p = q = 1)$.

We first show that LW 1-Lip NN with spline activations with 2 linear regions cannot represent nonaffine $\mathbb{R} \to \mathbb{R}$ CPWL functions that have a slope of one at both ends, for instance, the soft-thresholding function. Let f be a $\mathbb{R} \to \mathbb{R}$ LW 1-Lip NN (with 2-norm constraints) with $L > 0$ hidden layers so that $f(x) = \mathbf{w}^T \boldsymbol{\varphi}(\boldsymbol{f}^{L-1}(x))$, where $\boldsymbol{\varphi}$ is an element-wise activation function with 2-linear region

splines components φ_k, $k = 1, \dots, d$. In the sequel we use the notation $\boldsymbol{g} = \boldsymbol{\varphi} \circ \boldsymbol{f}^{L-1}$ and $h'(\pm\infty) = \lim_{t \to \pm\infty} h'(t)$, which is well-defined for any $\mathbb{R} \to \mathbb{R}$ CPWL function. We now suppose that $f'(+\infty) = f'(-\infty) = 1$. The LW 1-Lip assumption implies that $\|\mathbf{w}\|_2 \leq 1$ and $\|\boldsymbol{g}'(x)\|_2 \leq 1$. Given that $f' = \mathbf{w}^T \boldsymbol{g}'$, the Cauchy–Schwarz inequality implies that $f'(x) \leq 1$, and, for any $x_0 \in \mathbb{R}$ such that $f'(x_0) = 1$, it must hold that $\boldsymbol{g}'(x_0) = \mu \mathbf{w}$ for some $\mu \in \mathbb{R}_{\geq 0}$ and since we also have that $\|\mathbf{w}\|_2 = \|\boldsymbol{g}'(x_0)\|_2 = 1$, we conclude that $\boldsymbol{g}'(x_0) = \mathbf{w}$. Without loss of generality, we can suppose that $w_k \neq 0$ for $k = 1, \dots, d$, otherwise the corresponding neuron can be removed. We infer that $g_k'(\pm\infty) = w_k \neq 0$, which means, for a CPWL function, that it is surjective. Given that $g_k = \varphi_k \circ f_k^{L-1}$, the activation φ_k must also be surjective, which, for a linear spline with 2 linear regions implies that φ_k is bijective. Hence, for g_k to be surjective, f_k^{L-1} also needs to be surjective. We now note that $\boldsymbol{g}'(x) = \boldsymbol{\varphi}'(\boldsymbol{f}^{L-1}(x)) \odot (\boldsymbol{f}^{L-1})'(x)$, where $\odot$ is the Hadamard product. Let $\mathbf{a} = \boldsymbol{\varphi}'(\boldsymbol{f}^{L-1}(+\infty))$. From the assumptions, it holds that $\|\mathbf{a}\|_\infty \leq 1$ (1-Lip activations), and since f_k^{L-1} is surjective, $f_k^{L-1}(+\infty) = \pm\infty$. Let $\mathbf{b} = (\boldsymbol{f}^{L-1})'(+\infty)$. It holds that $\|\mathbf{b}\|_2 \leq 1$ and $\mathbf{b}$ has nonzero entries since f_k^{L-1} is surjective. Recalling that $\|\boldsymbol{g}'(+\infty)\|_2 = 1$, it holds $1 = \|\mathbf{a} \odot \mathbf{b}\|_2^2 = \sum_k a_k^2 b_k^2 \leq \|\mathbf{a}\|_\infty^2 \|\mathbf{b}\|_2^2 \leq 1$. Since the last inequality is an equality, it must hold that $a_k = \pm 1$ since $b_k \neq 0$ for $k = 1, \dots, d$. With a similar reasoning at $-\infty$, we eventually infer that $\varphi_k'(f_k^{L-1}(\pm\infty)) = 1$. Given the surjectivity of f_k^{L-1}, $f_k^{L-1}(+\infty) = -f_k^{L-1}(-\infty) = \pm\infty$, and hence the 2-linear spline φ_k has similar slopes at both ends and hence is affine. In other words, the last layer can be replaced by an affine one. By iterating over the depth of the NN depth, we obtain that the function is affine.

- **Multi-dimensional functions**

  Let $\mathbf{u} \in \mathbb{R}^p, \mathbf{v} \in \mathbb{R}^q$ with $\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1$, where and $p, q \in \mathbb{N}$. Let $\mathbf{h} \colon \mathbf{x} \mapsto f(\mathbf{u}^T \mathbf{x})\mathbf{v}$, where f is a nonaffine function with slopes of one at both ends as considered in the first part of the proof. By contradiction, suppose that $\mathbf{h}$ can be represented by a LW 1-Lip NN with LLS_2 activations. Then, by the addition of linear layers at both ends of this NN, there exists a LW 1-Lip NN with LLS_2 activations that can represent f since $f(x) = \mathbf{v}^T \mathbf{h}(x\mathbf{u})$, which is a contradiction.

□

5.4 3 Linear Regions Suffice

In this section, we investigate the representation and approximation of 1-Lipschitz functions using LW 1-Lip NNs with LLS with more than 2 linear regions.

As a first step, we study the representation and approximation power of LLS with no restriction on the number of linear regions and then show that, in theory, 3 linear regions are enough to reach optimal representation power.

5.4.1 A Representer Theorem

Among LW 1-Lip NNs with component-wise activation functions, LLS-NNs achieve the optimal representation and approximation power.

Lemma 5.2. *Let $\mathbf{x}_m \in \mathbb{R}^d$, $m = 1, \dots, N$, and $\mathbf{\Phi}\colon \mathbb{R}^d \to \mathbb{R}^p$ a NN with K layers, parameter set $\boldsymbol{\theta}$, p-norm weight constraints and 1-Lipschitz activation functions. Then, there exists an LLS-NN $\mathbf{\Psi}$ with the same architecture as $\mathbf{\Phi}$, where the activation functions are replaced by 1-Lipschitz linear splines with no more than $(M-1)$ linear regions such that*

$$\mathbf{\Phi}(\mathbf{x}_m, \boldsymbol{\theta}) = \mathbf{\Psi}(\mathbf{x}_m, \boldsymbol{\theta}^*) \text{ for } m = 1, \dots, M. \tag{5.29}$$

Proof. On the data points $(\mathbf{x}_m)_{m=1}^M$, the activation functions of $\mathbf{\Phi}$ are evaluated for at most M different values. Hence, the result directly follows by interpolating these values using a linear spline, which yields 1-Lipschitz linear-spline activation functions with at most $(M-1)$ linear regions. □

Lemma 5.2 is somehow unsatisfying as the number of linear regions of each spline may be as large as the number of training points. In the unconstrained case, i.e. no control of the Lipschitz constant of the NN, it was proposed in [216] to penalize the second-order total variation of the LLS during training to favor activations with few linear regions. We now extend the framework proposed in [216] to LW 1-Lip NNs, and thereby derive an important result to motivate the universality of LLS.

Similar to the discussion in Section 5.3.1, we rely on the notion of second-order total variation, defined as $\mathrm{TV}^{(2)}(f) = \|\mathrm{D}^2 f\|_{\mathcal{M}}$, where D is the distributional derivative operator and $\|\cdot\|_{\mathcal{M}}$ is the total-variation norm. For any function f in the space $L_1(\mathbb{R})$ of absolutely integrable functions, it holds that $\|f\|_{\mathcal{M}} = \|f\|_{L_1}$. However, unlike the L_1 norm, the total-variation norm is also well-defined for any shifted Dirac impulse with $\|\delta(\cdot - \tau)\|_{\mathcal{M}} = 1$, $\tau \in \mathbb{R}$ (see Section 5.7 and [216] for more technical details). In the sequel, we consider activation functions in the space

$$\mathrm{BV}^{(2)}(\mathbb{R}) = \{f : \mathbb{R} \to \mathbb{R} \text{ s.t. } \mathrm{TV}^{(2)}(f) < +\infty\}$$

of functions with bounded second-order total variation.

Given a series $(\mathbf{x}_m, \mathbf{y}_m)$, $m = 1, \dots, M$, of data points and an NN $\mathbf{\Psi}(\cdot, \boldsymbol{\theta})\colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_K}$ with $\mathbf{\Psi}(\cdot, \boldsymbol{\theta}) = \boldsymbol{\sigma}_K \circ \mathbf{\Phi}(\cdot, \boldsymbol{\theta})$, where $\mathbf{\Phi}(\cdot, \boldsymbol{\theta})\colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_K}$ has architecture (5.2) but with free-form element-wise activation functions $\boldsymbol{\sigma}_k$ in $\mathrm{BV}^{(2)}$, and where $\boldsymbol{\theta}$ denotes the set of learnable elements, including the weights, the biases and the nonparametric activation

functions. We now propose the constrained regularized training problem

$$\underset{\substack{\mathbf{W}_k,\mathbf{b}_k,\sigma_{k,n}\in \mathrm{BV}^{(2)}(\mathbb{R}) \\ \text{s.t. } \mathrm{Lip}(\sigma_{k,n})\leq 1,\, \|\mathbf{W}_k\|\leq 1}}{\arg\min} \left(\sum_{m=1}^{M} E\Big(\mathbf{y}_m, \boldsymbol{\Psi}\left(\mathbf{x}_m, \boldsymbol{\theta}\right)\Big) + \lambda \sum_{k=1}^{K} \sum_{n=1}^{n_k} \mathrm{TV}^{(2)}\left(\sigma_{k,n}\right) \right), \tag{5.30}$$

where $E\colon \mathbb{R}^{n_k} \times \mathbb{R}^{n_k} \to \mathbb{R}^+$ is proper, lower-semicontinuous. Theorem 5.1 states that a neural network with linear-spline activation functions suffices to find a solution of (5.30).

Theorem 5.1. *A solution of* (5.30) *always exists and can be chosen as a neural network with linear spline activation functions with no more than* $(M-1)$ *linear regions each.*

Proof. Without loss of generality, all biases are assumed to be zero as they can be incorporated into the activation functions.

- **Existence.**

  First, we show that Problem (5.30) has a nonempty solution set. For $k = 1, \ldots, (K-1)$ and $\bar{\boldsymbol{\sigma}}_1 = \boldsymbol{\sigma}_1$, we can iteratively replace the activation $\bar{\boldsymbol{\sigma}}_k$ by $\boldsymbol{u}_k$ without changing the output as in

  $$\boldsymbol{\sigma}_{k+1}\Big(\mathbf{W}_{k+1}\bar{\boldsymbol{\sigma}}_k(\mathbf{z})\Big) = \boldsymbol{\sigma}_{k+1}\Big(\mathbf{W}_{k+1}(\bar{\boldsymbol{\sigma}}_k(\mathbf{z}) - \bar{\boldsymbol{\sigma}}_k(\mathbf{0})) + \mathbf{W}_{k+1}\bar{\boldsymbol{\sigma}}_k(\mathbf{0})\Big) \tag{5.31}$$

  $$= \bar{\boldsymbol{\sigma}}_{k+1}\Big(\mathbf{W}_k \boldsymbol{u}_k(\mathbf{z})\Big), \tag{5.32}$$

  where $\boldsymbol{u}_k(\cdot) = \bar{\boldsymbol{\sigma}}_k(\cdot) - \bar{\boldsymbol{\sigma}}_k(\mathbf{0}) \in \mathrm{BV}^{(2)}(\mathbb{R})$ and $\bar{\boldsymbol{\sigma}}_{k+1}(\cdot) = \boldsymbol{\sigma}_{k+1}(\cdot + \mathbf{W}_k\bar{\boldsymbol{\sigma}}_k(\mathbf{0}))$. For the last layer, we set

  $$\bar{\boldsymbol{\sigma}}_K\Big(\mathbf{W}_K \boldsymbol{u}_{K-1}(\mathbf{z})\Big) = \boldsymbol{u}_K\Big(\mathbf{W}_K u_{K-1}(\mathbf{z})\Big) + \mathbf{a}_K, \tag{5.33}$$

  where $\boldsymbol{u}_K(\cdot) = \bar{\boldsymbol{\sigma}}_K(\cdot) - \bar{\boldsymbol{\sigma}}_K(\mathbf{0}) \in \mathrm{BV}^{(2)}(\mathbb{R})$ and $\mathbf{a}_K = \bar{\boldsymbol{\sigma}}_K(\mathbf{0}) = \boldsymbol{\Psi}(\mathbf{0}, \boldsymbol{\theta}) \in \mathbb{R}^{n_K}$. It holds that $\mathrm{TV}^{(2)}\left(u_{k,n}\right) = \mathrm{TV}^{(2)}\left(\sigma_{k,n}\right)$, and $u_k(\mathbf{0}) = \mathbf{0}$, for $k = 1, \ldots, K$. Therefore, a solution of (5.30) exists if the restricted problem

  $$\underset{\substack{\mathbf{W}_k,\sigma_{k,n}\in \mathrm{BV}^{(2)}(\mathbb{R}) \\ \text{s.t. } \mathrm{Lip}(\sigma_{k,n})\leq 1,\, \|\mathbf{W}_k\|\leq 1 \\ \sigma_{k,n}(0)=0, |\sigma_{k,n}(1)|\leq 1 \\ \mathbf{a}_K\in\mathbb{R}^{n_K}}}{\arg\min} \left(\sum_{m=1}^{M} E\Big(\mathbf{y}_m, \boldsymbol{\Psi}\left(\mathbf{x}_m, \boldsymbol{\theta}\right) + \mathbf{a}_K\Big) + \lambda \sum_{k=1}^{K} \sum_{n=1}^{n_k} \mathrm{TV}^{(2)}\left(\sigma_{k,n}\right) \right) \tag{5.34}$$

  has a nonempty solution set. Since $\boldsymbol{\Psi}(\cdot, \boldsymbol{\theta})$ is 1-Lipschitz, it holds that

  $$\|\boldsymbol{\Psi}(\mathbf{x}_m, \boldsymbol{\theta}) - \mathbf{a}_K\| = \|\boldsymbol{\Psi}(\mathbf{x}_m, \boldsymbol{\theta}) - \boldsymbol{\Psi}(\mathbf{0}, \boldsymbol{\theta})\| \leq \|\mathbf{x}_m\|. \tag{5.35}$$

In addition, we have that

$$2\|\mathbf{a}_K\| = \|(\boldsymbol{\Psi}(\mathbf{x}_m, \boldsymbol{\theta}) - \mathbf{a}_K) - (\boldsymbol{\Psi}(\mathbf{x}_m, \boldsymbol{\theta}) + \mathbf{a}_K)\| \tag{5.36}$$

$$\leq \|\boldsymbol{\Psi}(\mathbf{x}_m, \boldsymbol{\theta}) - \mathbf{a}_K\| + \|\boldsymbol{\Psi}(\mathbf{x}_m, \boldsymbol{\theta}) + \mathbf{a}_K\| \tag{5.37}$$

and, hence, that

$$\|\boldsymbol{\Psi}(\mathbf{x}_m, \boldsymbol{\theta}) + \mathbf{a}_K\| \geq 2\|\mathbf{a}_K\| - \|\mathbf{x}_m\|. \tag{5.38}$$

The fact that E is coercive and positive implies that

$$\lim_{\|\mathbf{a}_K\| \to +\infty} \sum_{m=1}^{M} E\Big(\mathbf{y}_m, \boldsymbol{\Psi}\left(\mathbf{x}_m, \boldsymbol{\theta}\right) + \mathbf{a}_K\Big) = +\infty. \tag{5.39}$$

We conclude that there exists a constant $A > 0$ such that it suffices to minimize over $\|\mathbf{a}_K\| \leq A$ in (5.34). Now, the remaining steps for the proof of existence are the same as in the proof of Theorem 3 in [255].

- **LLS-NN solution.** For the second part of the claim, we follow the reasoning in [216]. Let the NN $\tilde{\boldsymbol{\Psi}}$ be a solution of (5.30) with weights $\tilde{\mathbf{W}}_k$, biases $\tilde{\mathbf{b}}_k$, and activation functions $\tilde{\sigma}_{k,n}$. When evaluating $\tilde{\boldsymbol{\Psi}}$ at the data point $\mathbf{x}_m$, we iteratively generate a series of vectors $\mathbf{z}_{m,k}, \tilde{\mathbf{y}}_{m,k} \in \mathbb{R}^{n_k}$ as follows.

  1. Initialization (input of the network): $\tilde{\mathbf{y}}_{m,0} = \mathbf{x}_m$.
  2. Iterative update: For $k = 1, \ldots, K$, calculate

$$\mathbf{z}_{m,k} = (z_{m,k,1}, \ldots, z_{m,k,n_k}) = \tilde{\mathbf{W}}_k \tilde{\mathbf{y}}_{m,k-1} + \tilde{\mathbf{b}}_k \tag{5.40}$$

  and define $\tilde{\mathbf{y}}_{m,k} = (\tilde{y}_{m,k,1}, \ldots, \tilde{y}_{m,k,n_k}) \in \mathbb{R}^{n_k}$ as

$$\tilde{y}_{m,k,n} = \tilde{\sigma}_{k,n}\left(z_{m,k,n}\right), \quad n = 1, \ldots, n_k. \tag{5.41}$$

We directly observe that $\tilde{\mathbf{y}}_{m,k}$ only depends on the values of $\tilde{\sigma}_{k,n} \colon \mathbb{R} \to \mathbb{R}$ at the locations $z_{m,k,n}$. Hence, the optimal $\tilde{\sigma}_{k,n}$ are the 1-Lipschitz interpolations between these points with minimal second-order total variation, as the regularizers for $\tilde{\sigma}_{k,n}$ in (5.34) do not depend on each other. More precisely, the $\tilde{\sigma}_{k,n}$ solve the problem

$$\tilde{\sigma}_{k,n} \in \underset{\substack{f \in \mathrm{BV}^{(2)}(\mathbb{R}) \\ \text{s.t. } \mathrm{Lip}(f) \leq 1}}{\arg\min} \ \mathrm{TV}^{(2)}(f) \quad \text{s.t.} \quad f\left(z_{m,k,n}\right) = \tilde{y}_{m,k,n}, \ m = 1, \ldots, M. \tag{5.42}$$

We assume that $z_{m,k,n}$ are distinct for $m = 1, \ldots, M$. Otherwise, we can remove the duplicates as $z_{m_1,k,n} = z_{m_2,k,n}$ implies that $\tilde{y}_{m_1,k,n} = \tilde{y}_{m_2,k,n}$. The unconstrained problem

$$\min_{f \in \mathrm{BV}^{(2)}(\mathbb{R})} \mathrm{TV}^{(2)}(f) \quad \text{s.t.} \quad f\left(z_{m,k,n}\right) = \tilde{y}_{m,k,n}, \ m = 1, \ldots, M, \tag{5.43}$$

has a linear-spline solution $v_{k,n}$ with no more than $M-2$ knots [260, Proposition 5]. It can be shown [261] that the Lipschitz constant of this *canonical solution* is given by $\max_{m_1 \neq m_2} |\tilde{y}_{m_1,k,n} - \tilde{y}_{m_2,k,n}|/|z_{m_1,k,n} - z_{m_2,k,n}| \leq \mathrm{Lip}(\tilde{\sigma}_{k,n})$. Hence, there exists a linear-spline $v_{k,n}$ with $\tilde{\sigma}_{k,n}(z_{m,k,n}) = v_{k,n}(z_{m,k,n})$ for all $m = 1, \ldots, M$, $\mathrm{Lip}(v_{k,n}) \leq \mathrm{Lip}(\tilde{\sigma}_{k,n})$, and $\mathrm{TV}^{(2)}(v_{k,n}) = \mathrm{TV}^{(2)}(\tilde{\sigma}_{k,n})$.

□

The result proposed in Theorem 5.1, which is closely related to the representer theorems in [216] and [255], shows that there exists an optimal solution with linear-spline activation functions. The important point in the statement of the theorem is that each neuron has its own free parameters, including the (a priori unknown) number of linear regions, the determination of which is part of the training procedure. Beside the strict control of the Lipschitz constant, which is not covered by the representer theorems in [216], a noteworthy improvement brought by Theorem 5.1 is the existence of a solution, which is still an open problem in the unconstrained case. To this end, we have assumed that the last layer of the neural network $\boldsymbol{\Psi}$ consists of an activation function $\boldsymbol{\sigma}_{K,\boldsymbol{\alpha}_K}$ only. This theoretical setup for the proof is not a strong restriction since the freeform activation $\boldsymbol{\sigma}_{K,\boldsymbol{\alpha}_K}$ has the possibility to be the identity mapping in practice.

5.4.2 3 Linear Regions Suffice

The previous section illustrates the theoretical motivations for the use of linear-spline activations, but the number of linear regions needed remains unclear. As the first step, we restrict our attention to functions on the real line. In particular, we show that any CPWL activation function $\sigma \colon \mathbb{R} \to \mathbb{R}$ can be written as a composition of linear splines with no more than 3 linear regions.

Proposition 5.6. *Let $g \colon \mathbb{R} \to \mathbb{R}$ be a 1-Lipschitz CPWL function. Then, there exist $n \in \mathbb{N}$ and 1-Lipschitz CPWL functions $g_i \colon \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, n$, with at most 3 linear regions such that $g = g_n \circ \cdots \circ g_1$.*

Proof. We proceed via induction over the number m of linear regions of g. For g with up to 3 linear regions, the claim is true. Now, assume that it holds for some $m \in \mathbb{N}$ and let g be linear on the $m+1 > 3$ intervals $[a_i, a_{i+1}]$, $i = 0, \ldots, m$, with $a_0 = -\infty$ and $a_{m+1} = +\infty$, and let $s_\pm = \lim_{x \to \pm\infty} g'(x)$. First, we can write $g = g_1 \circ g_2$ with

$$g_1(x) = \begin{cases} g(a_1) + \mathrm{sign}(s_-)(x - a_1) & \text{for } x < a_1, \\ g(x) & \text{for } a_1 \leq x \leq a_m, \\ g(a_m) + \mathrm{sign}(s_+)(x - a_m) & \text{otherwise,} \end{cases} \tag{5.44}$$

and

$$g_2(x) = \begin{cases} a_1 + |s_-|(x - a_1) & \text{for } x < a_1, \\ x & \text{for } a_1 \leq x \leq a_m, \\ a_m + |s_+|(x - a_m) & \text{otherwise.} \end{cases} \tag{5.45}$$

Since g_2 has three linear regions and g_1 has the same number of linear regions as g, we can limit our discussion to functions g with $\lim_{x\to\pm\infty} |g'(x)| = 1$.

Case 1: There exists some a_j, $j \in \{2, \dots, m-1\}$, such that the function g has an extremum in a_j when restricted to $(-\infty, a_j]$ or $[a_j, +\infty)$. As all possible cases are similar, we only provide the construction for $g(a_j)$ being a maximum of g on $(-\infty, a_j)$. To this end, we define the functions $\tilde{g}_1$, $\tilde{g}_2$ as

$$\tilde{g}_1(x) = \begin{cases} g(x) & \text{for } x \leq a_j, \\ g(a_j) + (x - a_j) & \text{otherwise,} \end{cases} \tag{5.46}$$

and

$$\tilde{g}_2(x) = \begin{cases} x & \text{for } x \leq g(a_j), \\ g\Big(x + a_j - g(a_j)\Big) & \text{otherwise,} \end{cases} \tag{5.47}$$

which are both 1-Lipschitz piecewise-linear functions with at most m linear regions and satisfying $\lim_{x\to\pm\infty} |\tilde{g}_i'(x)| = 1$. Further, it holds that $g = \tilde{g}_2 \circ \tilde{g}_1$, so that we can apply the induction assumption to conclude the argument.

Case 2: Case 1 does not apply and $\lim_{x\to+\infty} g'(x)/g'(-x) = 1$. In the following, we reduce this to Case 1. We only provide the construction for $\lim_{x\to-\infty} g'(x) = 1$, the other case being similar. Here, it holds that $g(a_1) \geq g(a_i) \geq g(a_m)$ for all $i = 1, \dots, m$ and we now define the functions $\tilde{g}_1$, $\tilde{g}_2$ as

$$\tilde{g}_1(x) = \begin{cases} g(x) & \text{for } x < a_1, \\ 2g(a_1) - g(x) & \text{for } a_1 \leq x \leq a_m, \\ g(x) + 2\Big(g(a_1) - g(a_m)\Big) & \text{otherwise,} \end{cases} \tag{5.48}$$

and

$$\tilde{g}_2(x) = \begin{cases} x & \text{for } x < g(a_1), \\ 2g(a_1) - x & \text{for } g(a_1) \leq x \leq 2g(a_1) - g(a_m), \\ 2\Big(g(a_m) - g(a_1)\Big) + x & \text{otherwise.} \end{cases} \tag{5.49}$$

Clearly, both of the functions satisfy $\lim_{x\to\pm\infty} |\tilde{g}_i'(x)| = 1$ and are 1-Lipschitz. Here, the first function has $m + 1$ linear regions and the second one has 3. Further, the first function now fits Case 1 and it remains to show that $g = \tilde{g}_2 \circ \tilde{g}_1$. However, this follows immediately from $g(a_1) \geq \tilde{g}_1(x) \geq (g(a_1) - g(a_m))$ for $x \in [a_1, a_m]$.

Case 3: Case 1 does not apply and $\lim_{x\to+\infty} g'(x)/g'(-x) = -1$. This case can be reduced to either Case 1 or Case 2. We assume that $\lim_{x\to-\infty} g'(x) = 1$ and note that the other case is again similar. Then, it holds that $\min\{g(a_1), g(a_m)\} \geq g(a_i)$ for all $i = 1, \dots, m$ and we choose $a^* \in \arg\max_{x\in\mathbb{R}} g(x) \in \{a_1, a_m\}$. Next, we define the functions $\tilde{g}_1$, $\tilde{g}_2$ as

$$\tilde{g}_1(x) = \begin{cases} g(x) & \text{for } x < a^*, \\ 2g(a^*) - g(x) & \text{otherwise,} \end{cases} \tag{5.50}$$

and

$$\tilde{g}_2(x) = \begin{cases} x & \text{for } x < g(a^*), \\ 2g(a^*) - x & \text{otherwise.} \end{cases} \tag{5.51}$$

Note that both functions satisfy $\lim_{x\to\pm\infty} |\tilde{g}_i'(x)| = 1$ and are 1-Lipschitz. Here, the first function has $m + 1$ linear regions and the second one has 2. Further, the first function now fits either Case 1 or Case 2 and, hence, it remains to show that $g = \tilde{g}_2 \circ \tilde{g}_1$. However, this follows immediately from the definition of a^*. □

Remark 5.3. *Proposition 5.6 can be interpreted as an approximation result for NNs with one neuron per hidden layer. Note that a similar approximation result for ResNets without Lipschitz constraints was given in [262].*

Proposition 5.6 implies that deep spline NNs with very simple activation functions already suffice to achieve the maximum representation and approximation power among LW 1-Lip NNs with element-wise activations.

Theorem 5.2. *Let $D \subset \mathbb{R}^d$ be compact. Then, LW 1-Lip NNs with p-norm-constrained weights, and linear spline activation functions with 3 linear regions can approximate any LW 1-Lip NNs with p-norm-constrained weights and arbitrary component-wise activation functions.*

Proof. We proceed by induction over the number K of layers of $\boldsymbol{\Phi}$. For $K = 1$, the NN $\boldsymbol{\Phi}$ produces an affine mapping and there is nothing to show. Assume that the statement holds for K layers. Let $\boldsymbol{\Phi}_{K+1} \colon \mathbb{R}^d \to \mathbb{R}^{n_{K+1}}$ be a NN of the form (5.2) with p-norm-constrained weights and $K + 1$ layers. Then, $\boldsymbol{\Phi}_{K+1} = \boldsymbol{A}_{K+1} \circ \boldsymbol{\sigma}_{\boldsymbol{\alpha}_K} \circ \boldsymbol{\Phi}_K$ with a K-layer NN $\boldsymbol{\Phi}_K \colon \mathbb{R}^d \to \mathbb{R}^{n_K}$ of the same form. By application of the induction assumption, for any $\epsilon \in \mathbb{R}_{>0}$ there exists a deep spline NN $\boldsymbol{\Psi}_1 \colon \mathbb{R}^d \to \mathbb{R}^{n_K}$ with p-norm-constrained weights such that $\max_{\mathbf{x}\in D} \|\boldsymbol{\Phi}_K(\mathbf{x}) - \boldsymbol{\Psi}_1(\mathbf{x})\|_p \leq \epsilon/2$. Due to the finite diameter of D, the range of 1-Lipschitz functions is compact. Hence, Proposition 5.6 implies that there exists a deep spline NN $\boldsymbol{\Psi}_2 \colon \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ with all affine transformations being identities such that $\max_{\mathbf{x}\in\boldsymbol{\Phi}_K(D)} \|\boldsymbol{\sigma}_{\boldsymbol{\alpha}_K}(\mathbf{x}) - \boldsymbol{\Psi}_2(\mathbf{x})\|_p \leq \epsilon/2$. For the deep spline NN $\boldsymbol{A}_{K+1} \circ \boldsymbol{\Psi}_2 \circ \boldsymbol{\Psi}_1$

with spectral-norm-constrained weights, we can bound the error as

$$\begin{aligned}
&\max_{\mathbf{x}\in D} \|\mathbf{\Phi}(\mathbf{x}) - \boldsymbol{A}_{K+1} \circ \mathbf{\Psi}_2 \circ \mathbf{\Psi}_1(\mathbf{x})\|_p \leq \max_{\mathbf{x}\in D} \|\boldsymbol{\sigma}_{\boldsymbol{\alpha}_K} \circ \mathbf{\Phi}_K(\mathbf{x}) - \mathbf{\Psi}_2 \circ \mathbf{\Psi}_1(\mathbf{x})\|_p \\
\leq &\max_{\mathbf{x}\in D} (\|\boldsymbol{\sigma}_{\boldsymbol{\alpha}_K} \circ \mathbf{\Phi}_K(\mathbf{x}) - \mathbf{\Psi}_2 \circ \mathbf{\Phi}_K(\mathbf{x})\|_p + \|\mathbf{\Psi}_2 \circ \mathbf{\Phi}_K(\mathbf{x}) - \mathbf{\Psi}_2 \circ \mathbf{\Psi}_1(\mathbf{x})\|_p) \\
\leq &\epsilon/2 + \max_{\mathbf{x}\in D} \|\mathbf{\Phi}_K(\mathbf{x}) - \mathbf{\Psi}_1(\mathbf{x})\|_p \leq \epsilon. \qquad (5.52)
\end{aligned}$$

This concludes the proof. □

Theorem 5.2 tells us that, among LW 1-Lip NNs with component-wise 1-Lipschitz activation functions, splines with 3 linear regions achieve the optimal representation power. Meanwhile, the question of whether deep spline networks with p-norm-constrained weights are universal approximators for $\mathrm{Lip}_{1,p}(D)$ is part of ongoing research, and it appears to be a very challenging problem.

5.5 Comparison with GroupSort and Summary

In this Section, we discuss how GroupSort NNs and deep spline NNs can be expressed in terms of each other. Here, the situation differs depending on the norm chosen to constrain the weights. First, we recall a framework specifically tailored to GroupSort NNs, where the weights in Architecture (5.2) satisfy $\|\mathbf{W}_k\|_\infty \leq 1$, $k = 2, \ldots, K$, and $\|\mathbf{W}_1\|_{p,\infty} \leq 1$. Then, the expression of an arbitrary deep spline NN using a GroupSort NN is made possible due to the following universality result proved in [82, Theorem 3].

Proposition 5.7. *Let $D \subset \mathbb{R}^d$ be compact and $p \in [1, +\infty]$. The GroupSort NNs with Architecture* (5.2)*, group size at least 2, and weight constraints $\|\mathbf{W}_k\|_\infty \leq 1$, $k = 2, \ldots, K$, and $\|\mathbf{W}_1\|_{p,\infty} \leq 1$ are dense in $\mathrm{Lip}_{1,p}(D)$.*

Proposition 5.7, according to which density holds for all $p \in [1, +\infty]$, has unfortunately some gap with the application relevant to us in which $p = 2$. The reasons, as hinted by the authors of [82] are listed below.

- The norm constraint on the weight matrices (mixed norm and ∞-norm) makes it more difficult to train the NN than 2-norm constraints, leading to a degradation of the performance.
- The result heavily relies on the assumption that the NN is scalar-valued. For multidimensional outputs, the result cannot be directly extended. Indeed, the weight constraint scheme would not even guarantee the nonexpansiveness of the NN, which would be problematic for image reconstruction for example, where the DNN outputs and image.

Note that the proof of Proposition 5.7 relies heavily on the maximum operation and the chosen norms, which makes it difficult to generalize it to other norm constraints or activation functions.

Now, we discuss the case of spectral-norm constraints, which are the usual choice in practice. For this setting, let us recall that it holds that

$$\max(x_1, x_2) = \frac{x_1 + x_2 + |x_1 - x_2|}{2}. \tag{5.53}$$

Hence, in the case of the spectral-constrained weights, the MaxMin activation function can be written as the deep spline NN $\text{MaxMin}(\mathbf{x}) = \mathbf{W}_2\boldsymbol{\sigma}_1(\mathbf{W}_1\mathbf{x})$, where

$$\mathbf{W}_1 = \mathbf{W}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{\sigma}_1(\mathbf{x}) = \begin{pmatrix} x_1 \\ |x_2| \end{pmatrix}. \tag{5.54}$$

This can be extended to any GroupSort operation since the MaxMin operation has the same expressivity as GroupSort under any p-norm constraint [82]. We are not aware of any results for the reverse direction, i.e., to express a deep spline NN using a GroupSort NN with spectral-norm-constrained weights.

The representation power of LW 1-Lip NNs with 2-norm-constrained weights with the activation functions discussed in this chapter is summarized in Figure 5.4 and mathematically reads:

$$\mathcal{S}_2(\text{ReLU}) \subsetneq \mathcal{S}_2(\text{GS}) = \mathcal{S}_2(\text{HH}) \tag{5.55}$$

$$\subseteq \mathcal{S}_2(\text{AV}) \subseteq \mathcal{S}_2(\text{PReLU}) \subseteq \mathcal{S}_2(\text{LLS}_2) \subsetneq \mathcal{S}_2(\text{LLS}_3) = \mathcal{S}_2(\text{LLS}_{M>3}). \tag{5.56}$$

5.6 Conclusions and Open Problems

In this chapter, we have shown that neural networks (NNs) with linear-spline activation functions with at least three linear regions can approximate the maximal class of functions among all NNs with p-norm weight constraints and component-wise activation functions. However, it remains an open question whether these NNs are universal approximators of $\text{Lip}_{1,p}(D)$, $D \subset \mathbb{R}^d$ compact. While this problem appears to be very challenging, our result could be a first step toward its solution.

There also remain other open questions. Some of the proposed results, e.g. Proposition 5.1, pertain to scalar-valued functions $f\colon \mathbb{R}^d \to \mathbb{R}$, and their extension to vector-valued functions is not straightforward. Additionally, many of the proposed results pertain

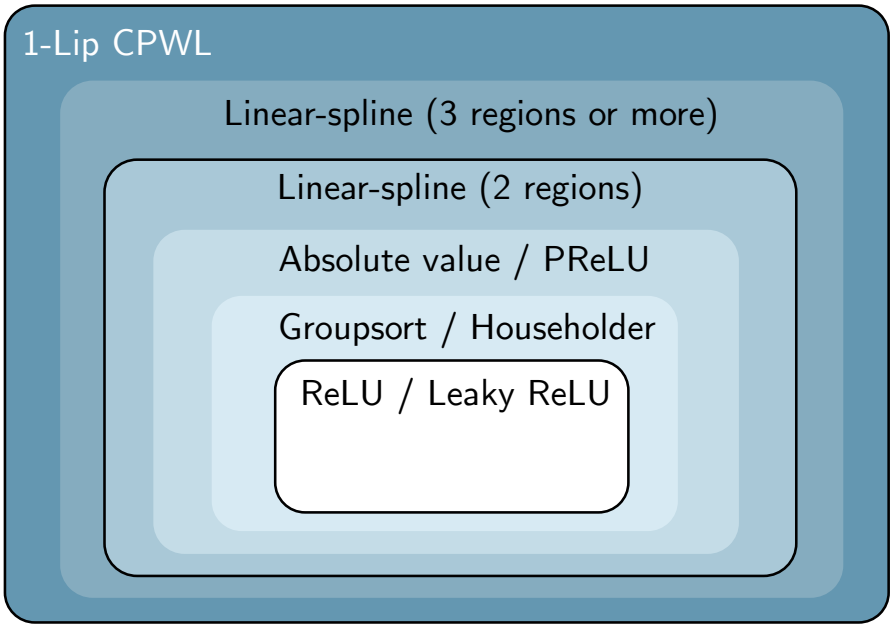

Figure 5.4: Representation power of layer-wise Lipschitz-constrained NNs with 2-norm-constrained weights for various activation functions. The name of the activation function is reported in the box representing the function space it generates (see (5.14)). The boundary between two spaces is a full line if the spaces are distinct. Otherwise, it is not known whether the spaces are distinct or not.

to representation power, leaving many unsolved questions regarding the approximation power.

In the next chapter, we will confirm the superiority of LLS over the other activation functions discussed in this chapter with empirical evidence.

5.7 Appendix – Second-Order Total Variation

Here, we mainly follow the exposition from [216]. Let $\mathcal{S}(\mathbb{R})$ denote the Schwartz space of smooth and rapidly decaying test functions equipped with the usual Schwartz topology, see [263]. The topological dual space of distributions is denoted by $\mathcal{S}'(\mathbb{R})$ and can be equipped with the total-variation norm

$$\|f\|_{\mathcal{M}} := \sup_{\varphi \in \mathcal{S}(\mathbb{R}): \|\varphi\|_\infty \leq 1} \langle f, \varphi \rangle. \tag{5.57}$$

As $\mathcal{S}(\mathbb{R})$ is dense in $C_0(\mathbb{R})$, the associated space $\mathcal{M}(\mathbb{R}) = \{f \in \mathcal{S}'(\mathbb{R}) : \|f\|_{\mathcal{M}} < \infty\}$ can be identified as the Banach space of Radon measures. Next, we require the second-order

distributional derivative $\mathrm{D}^2 \colon \mathcal{S}'(\mathbb{R}) \to \mathcal{S}'(\mathbb{R})$ defined via the identity

$$\langle \mathrm{D}^2 f, \varphi \rangle = \left\langle f, \frac{\mathrm{d}^2}{\mathrm{d}x^2} \varphi \right\rangle \qquad \forall \varphi \in \mathcal{S}(\mathbb{R}). \tag{5.58}$$

Based on this operator, the second-order total variation of $f \in \mathcal{S}'(\mathbb{R})$ is defined as

$$\mathrm{TV}^{(2)}(f) = \|\mathrm{D}^2 f\|_{\mathcal{M}} = \sup_{\varphi \in \mathcal{S}(\mathbb{R}) : \|\varphi\|_\infty \leq 1} \left\langle f, \frac{\mathrm{d}^2}{\mathrm{d}x^2} \varphi \right\rangle. \tag{5.59}$$

In order to make things more interpretable, we introduce the space of continuous functions $C_{b,1}(\mathbb{R}) = \{f \in C(\mathbb{R}) : \|f\|_{\infty,1} < \infty\}$ that grow at most linearly, which is equipped with the norm $\|f\|_{\infty,1} \coloneqq \sup_{x \in \mathbb{R}} |f(x)|(1 + |x|)^{-1}$. Now, we are ready to define the space of distributions with bounded second-order total variation as

$$\mathrm{BV}^{(2)}(\mathbb{R}) = \{f \in \mathcal{S}'(\mathbb{R}) : \mathrm{TV}^{(2)}(f) < \infty\} = \{f \in C_{b,1}(\mathbb{R}) : \mathrm{TV}^{(2)}(f) < \infty\}. \tag{5.60}$$

Note that $\mathrm{TV}^{(2)}(f) = 0$ if f is affine and, consequently, $\mathrm{TV}^{(2)}$ is only a seminorm.

6 Spline Activations for Stable Plug-and-Play Image Reconstruction

This chapter is the implementation counterpart of Chapter 5. A general introduction to the use and design of Lipschitz-constrained neural networks (NNs) can be found in Chapter 5, and we now mostly focus on plug-and-play image reconstruction methods. A higher-level introduction to the state-of-the-art in image reconstruction beyond plug and play can be found in Part III of this thesis. The text of this chapter is inspired by

S. Ducotterd, A. Goujon, P. Bohra, D. Perdios, S. Neumayer and M. Unser "Improving Lipschitz-constrained neural networks by learning activation functions", submitted to *Journal of Machine Learning Research*,

and

P. Bohra, D. Perdios, A. Goujon, S. Emery and M. Unser, "Learning Lipschitz-controlled activation functions in neural networks for plug-and-play image reconstruction methods", *NeurIPS 2021 Workshop on Deep Learning and Inverse Problems*.

Summary

Lipschitz-constrained neural networks give rise to convergent and stable plug-and-play methods. Unfortunately, it has been shown both theoretically and empirically that they perform poorly when equipped with ReLU activation functions. By contrast, neural networks with learnable 1-Lipschitz linear splines have promising theoretical advantages in this setup, as proven in Chapter 6. To put the theory into practice, we propose an efficient method to train learnable 1-Lipschitz linear splines. In particular, we show that the constraint is best handled via a reparameterization method compared to projected gradient methods. Our numerical experiments show that learnable linear splines improve the performance over the other existing 1-Lipschitz activation functions for denoising, and plug-and-play image reconstruction for CT and MRI.

6.1 Introduction to Plug-and-Play Methods

6.1.1 Motivations

Many image-reconstruction tasks can be formulated as linear inverse problems, with applications in medical-imaging [16], including magnetic resonance imaging (MRI) and X-ray computed tomography (CT). Specifically, the task is to recover an image $\mathbf{x} \in \mathbb{R}^n$ from the noisy measurement

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \in \mathbb{R}^m, \tag{6.1}$$

where $\mathbf{H} \in \mathbb{R}^{m \times n}$ is a measurement operator and $\mathbf{n} \in \mathbb{R}^m$ represents an additive noise. Problem (6.1) is nondeterministic and often ill-posed, in the sense that multiple images yield the same measurements. To make (6.1) well-posed, one usually incorporates regularization, which leads to the reconstruction problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + g(\mathbf{x}), \tag{6.2}$$

where the regularizer $g \colon \mathbb{R}^n \to \mathbb{R}^+$ allows for enforcing some prior knowledge on the solution to counteract the ill-posedness and the noise. Popular choices of regularizer include the Tikhonov [19] or total-variation (TV) [22, 30, 264] ones. These classical regularizers are convex, which makes it possible to numerically approximate a global minimum of Problem (6.1) to arbitrary precision. To do so, there exist various iterative algorithms, including the forward-backward splitting (FBS) algorithm [265], also known as proximal gradient descent, and the vanilla gradient descent (GD). The update rule of these iterative methods reads

$$\mathbf{x}^{k+1} = \operatorname{prox}_{\alpha g}\Big(\mathbf{x}^k - \alpha(\mathbf{H}^T\mathbf{H}\mathbf{x}^k - \mathbf{H}^T\mathbf{y})\Big), \qquad \text{(FBS)} \tag{6.3}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha(\mathbf{H}^T\mathbf{H}\mathbf{x}^k - \mathbf{H}^T\mathbf{y} + \boldsymbol{\nabla} g(\mathbf{x}^k)\Big), \qquad \text{(GD)} \tag{6.4}$$

where the proximal operator of g is defined as $\operatorname{prox}_g(\mathbf{z}) = \arg\min_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|^2 + g(\mathbf{x})$, and α is the stepsize. Both algorithms target a solution of (6.2), but FBS can handle nonsmooth regularizers, including the TV one [27]. For simplicity, we have only given the explicit update rule of two canonical algorithms, but there exist many other solvers. In particular, both FBS and GD have variants that seek to improve convergence speed and/or allow for enforcing constraints on the solution. In some applications, e.g. MRI reconstruction, other algorithms that resort to the proximal operator of both the data-fidelity term and g are popular, including the Douglas–Rachford splitting (DRS) method [266] and the alternating directions method of multipliers (ADMM) [267].

Classical regularizers typically come with theoretical guarantees but they remain too simple to compete with deep-learning-based methods. The idea behind PnP algorithms

[89, 93] is to extend classical algorithms by replacing g with an implicit regularization prior built from an off-the-shelf denoiser $\boldsymbol{D} : \mathbb{R}^n \to \mathbb{R}^n$, which is typically data-driven. For instance, in FBS, $\text{prox}_{\alpha g}$ can be interpreted as a denoiser, and hence directly replaced with $\boldsymbol{D}$. In GD, the gradient $\boldsymbol{\nabla} g$ of the regularizer can be interpreted as a noise extractor, and hence directly replaced with $(\mathbf{Id} - \boldsymbol{D})$. The latter approach is often referred to as regularization by denoising (RED) [90, 268, 269]. The update rules of the PnP versions of GD and FBS are given below.

$$\mathbf{x}^{k+1} = \boldsymbol{D}_\sigma \left(\mathbf{x}^k - \alpha(\mathbf{H}^T\mathbf{H}\mathbf{x}^k - \mathbf{H}^T\mathbf{y}) \right), \qquad \text{(PnP-FBS)} \qquad (6.5)$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \left(\mathbf{H}^T\mathbf{H}\mathbf{x}^k - \mathbf{H}^T\mathbf{y} + \lambda(\mathbf{Id} - \boldsymbol{D}_\sigma)(\mathbf{x}^k) \right), \qquad \text{(PnP-GD)} \qquad (6.6)$$

where σ is a noise level that controls the strength of the denoiser and, along with $\lambda \in \mathbb{R}$, it can be tuned to adapt the strength of the regularization.

PnP methods rely on implicit regularization because $\boldsymbol{D}_\sigma$ is not necessarily the proximal operator of a $g\colon \mathbb{R}^n \to \mathbb{R}$ or, similarly, $(\mathbf{Id} - \boldsymbol{D}_\sigma)$ might not be the gradient of some $g\colon \mathbb{R}^n \to \mathbb{R}$. Consequently, all the theoretical properties given by classical methods to solve (6.2) are not automatically transferred to PnP methods. A careful analysis is therefore needed to build trustworthy PnP methods, all the more for deep-learning-based denoisers which are black boxes. Indeed, it is now well-documented that deep-learning-based reconstruction methods may be unstable [57] and they are prone to hallucinations, which may lead to diagnosis errors in medical imaging, for instance, [60, 61].

In this chapter, we propose to build trustworthy PnP methods based on two types of theoretical guarantees.

- **Convergence of the iterates.** All PnP methods are iterative, and hence a minimal requirement is to have theoretical convergence guarantees to ensure stability throughout the iterations.
- **Properties of the reconstruction map (data consistency and stability).** Convergence only pertains to the stability of the iterations, not to the stability of the measurement-to-reconstruction map $\boldsymbol{y} \mapsto \boldsymbol{x}^\infty$, also referred to as the reconstruction map. Trust can be greatly improved if one can prove some theoretical properties of the reconstruction map, for instance, Lipschitz continuity, which ensures that perturbations in the measurements have limited effects on the reconstruction.

6.1.2 Convergent Plug-and-Play Methods

We now briefly review the popular convergent PnP methods and discuss the corresponding reconstruction map. The three most popular strategies to ensure convergence are

summarized below.

- **Iterative relaxation.** Given a bounded denoiser*, a convergent variant of PnP-ADMM can be designed with an iteration-dependent denoiser strength σ_k that is iteratively decreased whenever a certain condition is violated [93]. This approach can be generalized to an unconstrained denoiser with the relaxed projected gradient descent (RPGD) [94]. There, a relaxed version $\boldsymbol{D}_{\sigma,k} = \mu_k \boldsymbol{D}_\sigma + (1-\mu_k)\boldsymbol{D}_\sigma$ of the denoiser is used and μ_k is iteratively decreased whenever a condition is violated. Unfortunately, as discussed in [94], the catch with iterative relaxation methods is that the reconstruction is not necessarily the fixed point of an operator and, even if it is, the operator of interest cannot be known a priori and depends on the measurement and the initialization. Hence, it is highly nontrivial to infer properties of the reconstruction map and, in particular, we are not aware of any data consistency and stability guarantees.

- **Lipschitz constraints.** The most widespread strategy to ensure convergence is to constrain the Lipschitz constant of the residual $(\mathbf{Id} - \boldsymbol{D}_\sigma)$ [75, 81, 101, 243, 270, 271]. The prescribed Lipschitz bound is a function of the smallest eigenvalue of $\mathbf{H}^T\mathbf{H}$ [81]. This approach allows for residual learning, which is known to boost the performance of data-driven denoisers [272]. For most inverse problems, $\mathbf{H}$ is noninvertible, and the usual sufficient condition becomes that $\mathbf{D}_\sigma$ must be nonexpansive. This strong constraint causes a performance drop and, hence, most works prefer to relax it, at the cost of losing all guarantees. For instance, it is common to use non-1-Lipschitz learning modules, such as batch normalization [81], or to only constrain the residual $(\mathbf{Id} - \boldsymbol{D}_\sigma)$ to be nonexpansive [81, 83, 101], with the caveat that $\mathrm{Lip}(\boldsymbol{D}_\sigma)$ can be as large as 2. Another relaxation approach consists of penalizing during training either the norm of the Jacobian of $\boldsymbol{D}_\sigma$ at a finite set of locations [75, 76] or of another local estimate of the Lipschitz constant [77, 78]. If one sticks to the constraint, not only convergence is granted, but also it confers strong properties to the reconstruction map, including some data consistency and stability results, as we further detail in Section 6.3.

- **Non-convex optimization.** Many recent PnP methods parameterize $\boldsymbol{D}_\sigma$ as the gradient of some $\mathbb{R}^n \to \mathbb{R}$ CNN, for both gradient-based [99, 273] and prox-based PnP [76, 98, 102]. Then, convergence can be proven with classic non-convex optimization results [96, 97]. In this case, the fixed-point equation satisfied by the reconstruction is known, but the non-convexity makes it hard to prove any property of the reconstruction map, which is typically not single-valued and highly dependent on the initialization of the algorithm.

Among the three strategies that provide convergence guarantees, only the second one seems able to offer stability guarantees for the reconstruction map. Hence, for the

*$\boldsymbol{D}_\sigma$ is said to be bounded if there exists $C \in \mathbb{R}$ such that $\|\boldsymbol{D}_\sigma(\mathbf{x}) - \mathbf{x}\|^2 \leq C\sigma^2 n$ for any $\mathbf{x} \in \mathbb{R}^n$

remainder of the chapter, we concentrate on PnP methods with Lipschitz-constrained denoisers, for which we will detail the general convergence and stability results.

Outline and Contributions In Section 6.2, we recall some mathematical concepts concerning averaged operators, fixed-point iterations and convergence of PnP algorithms. To further motivate the use of Lipschitz-constrained NNs in PnP methods, we show in Section 6.3 that they yield reconstruction maps with some stability and consistency guarantees. In Section 6.4, we propose a reparameterization technique to learn efficiently Lipschitz-constrained learnable linear spline (LLS) activation functions. In this way, the constraints are embedded into the forward and backward passes of the model. In Section 6.5, we first illustrate our method on toy problems, including 1-dimensional function fitting and image generation with Wasserstein GANs. Finally, we design Lipschitz-constrained denoisers in Section 6.6 and show in various MRI and CT setups that LLS yield superior performance than the other popular activation functions, including ReLU, GroupSort (GS) and householder (HH).

6.2 Preliminaries: Averaged Operators and Fixed-point Iterations

Averaged Operators An operator $\boldsymbol{T}\colon \mathbb{R}^n \to \mathbb{R}^n$ is said to be

- contractive if $\mathrm{Lip}(\boldsymbol{T}) < 1$,
- nonexpansive if $\mathrm{Lip}(\boldsymbol{T}) \leq 1$,
- β-averaged for $\beta \in (0,1)$ if $\boldsymbol{T} = \beta \boldsymbol{U} + (1-\beta)\mathbf{Id}$ where $\boldsymbol{U}$ is a nonexpansive operator,

where $\mathrm{Lip}(\boldsymbol{T})$ is the Lipschitz constant of $\boldsymbol{T}$ as defined in Chapter 5. Note that throughout this chapter we only consider the Lipschitz constant with respect to the 2-norm.

Fixed-Point Iterations In PnP methods, the reconstruction is computed as a fixed point of an update operator, say $\boldsymbol{T}\colon \mathbb{R}^n \to \mathbb{R}^n$, with the Banach-Picard iterations $\mathbf{x}_{k+1} = \boldsymbol{T}(\mathbf{x}_k)$. For a contractive $\boldsymbol{T}$, there exists a unique fixed point, and the iterates converge to it regardless of the initialization $\mathbf{x}_0$ (Banach fixed-point theorem). For an averaged operator $\boldsymbol{T}$, the fixed-point set is known to be convex and, if it is nonempty, convergence of the iterates is ensured. For a nonexpansive operator with a nonempty fixed-point set, convergence can be ensured via the use of the Krasnosel'skiĭ-Mann algorithm, which performs updates based on an averaged relaxation of $\boldsymbol{T}$.

Convergence of Lipschitz-constrained PnP Methods To prove the convergence of PnP methods, one can rely on the following fundamental properties [83, 274]

- the composition of averaged operators is averaged,
- the convex combination of averaged operators is averaged,
- $\mathbf{x} \mapsto \mathbf{x} - \alpha \mathbf{H}^T \mathbf{H}\mathbf{x}$ is averaged for $\alpha \in [0, 2/\|\mathbf{H}^T\mathbf{H}\|)$, where $\|\cdot\|$ is the spectral norm,
- $\mathbf{x} \mapsto \mathbf{x} - \mu(\mathbf{x} - \boldsymbol{D}(\mathbf{x}))$ is averaged for $\mathrm{Lip}(\boldsymbol{D}) \leq 1$ and $\mu \in [0, 1)$. In addition, if $\boldsymbol{D}$ is β-averaged, the second condition can be relaxed to $\mu \in [0, 1/\beta)$.

All in all, the use of a β-averaged denoiser yields convergent[†] PnP methods for [83, 274]

- $0 \leq \alpha < 2/\|\mathbf{H}^T\mathbf{H}\|$, (PnP-FBS)
- $0 \leq \alpha < 2/(\|\mathbf{H}^T\mathbf{H}\| + 2\lambda\beta))$. (PnP-GD)

6.3 Properties of the Reconstruction Map for PnP with Lipschitz Constrained Denoisers

The efforts in the past decade to improve the reliability of PnP methods have mostly been concentrated on the convergence guarantees, leaving many open questions on the properties of the reconstruction map. In convex optimization, there exist extensive stability results, see [31, 275–277], but they do not necessarily generalize to PnP methods. The stability of gradient-based PnP was recently studied in see [78]. Similar to our approach, the authors of [78] rely on Lipschitz-constrained operators, but their results are quite different since they concentrate on learning monotone operators, while we focus on learning averaged operators and, the constraints are relaxed in [78] for the experiments, which is not the case in our framework.

To theoretically motivate the use of Lipschitz-constrained denoisers in PnP methods, we now provide a stability analysis of the reconstruction map given by PnP-GD and PnP-FBS. For completeness, we also provide the proofs in Section 6.8.1, but recall that our principal contributions pertain to the implementation of Lipschitz-constrained LLS, as given in Section 6.4 and illustrated in Sections 6.5 and 6.6. Note that the PnP versions of DRS and ADMM share the same fixed point as PnP-FBS [241], and hence the stability results derived for PnP-FBS also apply to PnP-DRS and PnP-ADMM. In the sequel, for a reconstruction map $\boldsymbol{f}$, even if it is not single-valued, we use the notation $\boldsymbol{f}(\mathbf{y})$ to represent any solution corresponding to the measurement $\mathbf{y}$. This slight abuse of notation aims at improving readability and only concerns Equations (6.8) and (6.11).

[†]If a fixed point exists.

Proposition 6.1 (Stability of PnP-GD)**.** *Let $L_\sigma = \mathrm{Lip}(\boldsymbol{D}_\sigma)$, $\boldsymbol{f}_{\mathrm{GD}}$ be the reconstruction map of PnP-GD* (6.6) *with regularization parameter $\lambda \in \mathbb{R}_+$, and $\lambda_{\max}$ and $\lambda_{\min}$ be the largest and smallest eigen values of $\mathbf{H}^T\mathbf{H}$.*

- *If $L_\sigma < 1 + \lambda_{\min}/\lambda$, then $\boldsymbol{f}_{\mathrm{GD}}$ is Lipschitz continuous as in*

$$\|\boldsymbol{f}_{\mathrm{FBS}}(\mathbf{y}_2) - \boldsymbol{f}_{\mathrm{FBS}}(\mathbf{y}_1)\|_2 \leq \frac{\|\mathbf{H}\|}{(\lambda(1-L_\sigma)+\lambda_{\min})}\|\mathbf{y}_2 - \mathbf{y}_1\|_2, \tag{6.7}$$

  for any $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^m$.

- *If $L_\sigma \leq 1$, then $\boldsymbol{f}_{\mathrm{GD}}$ is Lipschitz continuous in the measurement domain as in*

$$\|\mathbf{H}\boldsymbol{f}_{\mathrm{GD}}(\mathbf{y}_2) - \mathbf{H}\boldsymbol{f}_{\mathrm{GD}}(\mathbf{y}_1)\|_2 \leq \|\mathbf{y}_2 - \mathbf{y}_1\|_2, \tag{6.8}$$

  for any $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^m$.

The result on the stability of PnP-FBS relies on the spectral norm of $\mathbf{I} - \alpha\mathbf{H}^T\mathbf{H}$, which can be expressed in terms of the largest and smallest eigenvalues $\lambda_{\max}$ and $\lambda_{\min}$ of $\mathbf{H}^T\mathbf{H}$ as

$$L_\alpha = \|\mathbf{I} - \alpha\mathbf{H}^T\mathbf{H}\| = \max(|1-\alpha\lambda_{\min}|, |1-\alpha\lambda_{\max}|). \tag{6.9}$$

Proposition 6.2 (Stability of PnP-{FBS, DRS, ADMM})**.** *Let $L_\sigma = \mathrm{Lip}(\boldsymbol{D}_\sigma)$, $L_\alpha = \|\mathbf{I} - \alpha\mathbf{H}^T\mathbf{H}\|$ and $\boldsymbol{f}_{\mathrm{FBS}}$ be the reconstruction map of PnP-FBS* (6.5)*.*

- *If $L_\sigma L_\alpha < 1$, then $\boldsymbol{f}_{\mathrm{FBS}}$ is Lipschitz continuous as in*

$$\|\boldsymbol{f}_{\mathrm{FBS}}(\mathbf{y}_2) - \boldsymbol{f}_{\mathrm{FBS}}(\mathbf{y}_1)\|_2 \leq \frac{\alpha L_\sigma}{1 - L_\sigma L_\alpha}\|\mathbf{H}\|\|\mathbf{y}_2 - \mathbf{y}_1\|_2, \tag{6.10}$$

  for any $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^m$.

- *If $\boldsymbol{D}_\sigma$ is half-averaged, then $\boldsymbol{f}_{\mathrm{FBS}}$ is Lipschitz continuous in the measurement domain as in*

$$\|\mathbf{H}\boldsymbol{f}_{\mathrm{FBS}}(\mathbf{y}_2) - \mathbf{H}\boldsymbol{f}_{\mathrm{FBS}}(\mathbf{y}_1)\|_2 \leq \|\mathbf{y}_2 - \mathbf{y}_1\|_2, \tag{6.11}$$

  for any $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^m$.

Overall, the theoretical properties of the reconstruction map proven in this section tell us that if two sets of measurements are close to each other, then the corresponding reconstructions must also be close to each other. While this property seems to be a minimal requirement, it usually does not hold for standard PnP methods. The Lipschitz continuity of the reconstruction map is also a sufficient condition for the solution to be unique for a given set of measurements, and hence independent of the initialization of the solver.

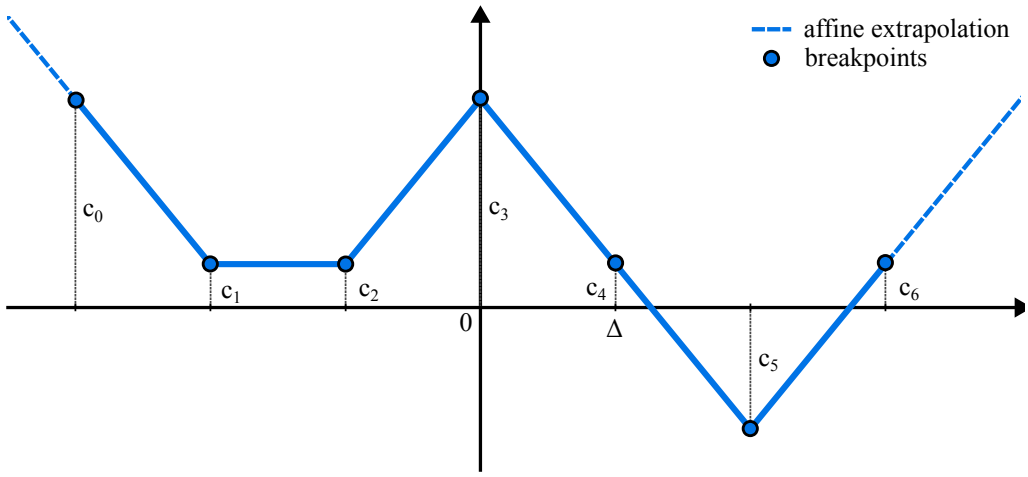

Figure 6.1: A LLS activation function with 7 learnable parameters $(c_k)_{k=0}^{6}$.

6.4 Implementing Lipschitz Constraints on Learnable Linear Splines

Following our theoretical results given in Chapter 5, we propose to use layer-wise (LW) 1-Lipschitz NNs with learnable linear spline activation to build convergent and stable PnP methods for ill-posed inverse problems.

6.4.1 Learnable Linear Splines

Similar to [189, 236], we use learnable linear splines (LLS) $\sigma_{\mathbf{c}} \colon \mathbb{R} \to \mathbb{R}$ with $(M+1)$ uniform knots $\nu_m = (m - M/2)\Delta$, $m = 0, \dots, M$, where Δ is the spacing of the knots. For simplicity, we assume that M is even. The learnable parameter $\mathbf{c} = (c_m)_{m=0}^{M} \in \mathbb{R}^{M+1}$ defines the value $\sigma_{\mathbf{c}}(\nu_m) = c_m$ of $\sigma_{\mathbf{c}}$ at the knots. To fully characterize $\sigma_{\mathbf{c}}$, we use affine extensions on $(-\infty, \nu_0]$ and $[\nu_M, +\infty)$. Additional details on the implementation of learnable linear splines can be found in [236]. An illustration of LLS can be found in Figure 6.1. Within LLS-NNs, the LLS can be initialized in many ways, some including popular activation functions such as ReLU, leaky ReLU, PReLU, or MaxMin. In addition, the LLS can also be shared by several neurons to save memory, and we usually use one activation function per channel in convolutional neural networks.

In practice, we choose a high number M and a small spacing Δ. We then ensure that a simple activation function is learned by using $\mathrm{TV}^{(2)}$ regularization, similar to [236]. Let $\mathbf{D}_{M+1} \in \mathbb{R}^{M \times (M+1)}$ be the one-dimensional finite-difference matrix with $(\mathbf{D}\mathbf{c})_m = c_{m+1} - c_m$ for $m = 0, \dots, (M-1)$. For our parameterization of LLS, it can be shown that $\mathrm{TV}^{(2)}(\sigma_{\mathbf{c}}) = \frac{1}{\Delta^2} \|\mathbf{D}_M \mathbf{D}_{M+1} \mathbf{c}\|_1$, which simply quantifies the total change of slope of the spline.

6.4.2 Constrained and Learnable Linear Splines

Lipschitz Constant

The LLS activation $\sigma_{\mathbf{c}}$ has an easily computable Lipschitz constant, namely

$$\mathrm{Lip}(\sigma_{\mathbf{c}}) = \frac{\|\mathbf{D}_{M+1}\mathbf{c}\|_\infty}{\Delta}. \tag{6.12}$$

This formula is a clear motivation for the use of LLS: while they bring additional expressivity compared to nonlearnable splines, they still have an efficiently computable Lipschitz constant.

Learning Linear Spline under Lipschitz Constraint

To optimize over the set $\mathcal{C}_{\mathrm{Lip}} = \{\mathbf{c} \in \mathbb{R}^{M+1} \colon -\Delta \leq \mathbf{D}_{M+1}\mathbf{c} \leq \Delta\}$, we propose two general approaches, namely a projected-gradient method, and a reparameterization strategy. To simplify the explanation, we only show how to learn a single LLS under the Lipschitz constraint, but the approach readily extends to multiple LLS deployed in CNNs.

In the sequel, $P_{\mathcal{C}_{\mathrm{Lip}}} \colon \mathbb{R}^{M+1} \to \mathbb{R}^{M+1}$ denotes any projection on $\mathcal{C}_{\mathrm{Lip}}$, $\mathrm{loss} \colon \mathbb{R}^{M+1} \to \mathbb{R}$ denotes a loss function that measures the performance of the spline on the problem of interest, e.g. interpolation, $\mathrm{optimizer} \colon \mathbb{R}^{M+1} \to \mathbb{R}^{M+1}$ denotes a generic gradient-based optimizer, which updates a parameter $\mathbf{c}_k$ based on the loss $\mathrm{loss}(\mathbf{c}_k)$.

PG-LLS (Projected-gradient Method) This method is inspired by constrained convex optimization methods, and we introduced it in the published paper [236]. The two-step update rule reads

$$\tilde{\mathbf{c}}_k = \mathrm{optimizer}(\mathrm{loss}(\mathbf{c}_k)), \qquad \text{(PG-LLS)} \tag{6.13}$$

$$\mathbf{c}_{k+1} = P_{\mathcal{C}_{\mathrm{Lip}}}(\tilde{\mathbf{c}}_k). \tag{6.14}$$

Given an appropriate convex loss function, and well-chosen optimizer and projection, this method allows one to find the optimal parameter $\mathbf{c}_k$ with guarantees that the Lipschitz constraint is met. For the PG-LLS, we will use the simple projector

$$P_{\mathcal{C}_{\mathrm{Lip}}} \colon \mathbf{c} \mapsto \frac{\Delta}{\|\mathbf{c}\|_\infty}\mathbf{c}, \tag{6.15}$$

as used in our first work on the topic [236]. In deep learning, the optimization being highly non-convex, PG-LLS is not necessarily well-suited.

R-LLS (Reparameterization Method) The reparameterization method that we now propose relies on a radically different philosophy. The idea is to parametrize directly $\mathcal{C}_{\text{Lip}}$ as $\{P_{\mathcal{C}_{\text{Lip}}}(\mathbf{c})\colon \mathbf{c} \in \mathbb{R}^{M+1}\}$, and to use the update rule

$$\mathbf{c}_{k+1} = \text{optimizer}(\text{loss}(P_{\mathcal{C}_{\text{Lip}}}(\mathbf{c}_k))) \qquad \text{(R-LLS)}. \tag{6.16}$$

The key observation here is that the spline used it not $\sigma_{\mathbf{c}}$, but rather the reparameterized version $\sigma_{P_{\mathcal{C}_{\text{Lip}}}(\mathbf{c})}$. In other words, the constraint is embedded into the forward pass of the computation of the loss. This way, the optimizer makes the update of the parameters with knowledge of the constraints, since it resorts to $\frac{\partial \text{loss}(P_{\mathcal{C}_{\text{Lip}}}(\mathbf{c}_k))}{\partial \mathbf{c}_k}$. This strategy is in line with the popular spectral normalization given in [80], where the weight matrices are unconstrained and parameterized using an approximate projection.

Regarding the choice of the projection, the textbook approach would be to use the least-squares projection onto $\mathcal{C}_{\text{Lip}}$. This operation preserves the mean of $\mathbf{c}$, as shown in Appendix 6.8.2. Unfortunately, its computation is very expensive as it requires solving a quadratic program and it is not trivial to make it differentiable. As a substitute, we introduce a simpler projection SplineProj that also preserves the mean while being much faster to compute. In brief, SplineProj computes the finite differences, clips them, sums them and adds a constant to preserve the mean.

Let us denote the Moore–Penrose pseudoinverse of $\mathbf{D}$ by $\mathbf{D}^\dagger$ and the vector of ones by $\mathbf{1} \in \mathbb{R}^{M+1}$, and we define the component-wise operation

$$\text{Clip}_\Delta(x) = \begin{cases} -\Delta, & x < -\Delta \\ x, & x \in [-\Delta, \Delta] \\ \Delta, & x > \Delta. \end{cases} \tag{6.17}$$

Proposition 6.3. *The operation* SplineProj *defined as*

$$\text{SplineProj}(\mathbf{c}) = \mathbf{D}^\dagger \,\text{Clip}_\Delta(\mathbf{D}\mathbf{c}) + \frac{1}{M+1}\mathbf{1}\mathbf{1}^T\mathbf{c} \tag{6.18}$$

has the following properties:

1. *it is a projection onto* $\mathcal{C}_{\text{Lip}}$*;*
2. *it is differentiable with respect to* $\mathbf{c}$ *almost everywhere;*
3. *it preserves the mean of* $\mathbf{c}$*.*

The proof of Proposition 6.3 can be found in Appendix 6.8.2.

For our reparameterization approach, Property 2 of Proposition 6.3 is very important

as it allows us to back-propagate through SplineProj during the optimization process. To compute SplineProj efficiently, we calculate $\mathbf{D}^\dagger$ in a matrix-free fashion with a cumulative sum. The computational cost of SplineProj is negligible compared to the cost of constraining the linear layer to be 1-Lipschitz.

Scaling Parameter

We propose to increase the flexibility of our LLS activation functions by the introduction of an additional trainable scaling factor $\mu \neq 0$. Specifically, we propose the new activation function

$$\tilde{\sigma}(x) = \sigma(\mu x)/\mu. \tag{6.19}$$

With this scaling, $\tilde{\sigma}$ has breakpoints within the interval $[\nu_0/\mu, \nu_M/\mu]$ and the Lipschitz constant

$$\mathrm{Lip}(\tilde{\sigma}) = \sup_{x_1 \neq x_2 \in \mathbb{R}} \frac{|\sigma(\mu x_1)/\mu - \sigma(\mu x_2)/\mu|}{|x_1 - x_2|} = \sup_{x_1 \neq x_2 \in \mathbb{R}} \frac{|\sigma(\mu x_1) - \sigma(\mu x_2)|}{|\mu x_1 - \mu x_2|} = \mathrm{Lip}(\sigma) \tag{6.20}$$

is left unchanged. As detailed in Appendix 6.8.3, the second-order total variation is preserved as well. Basically, μ allows us to potentially decrease the training loss without violating the constraints, and it comes with a negligible computational overhead. Experimentally, we indeed found that the performance of LLS-NNs improves if we also optimize over μ. In contrast, the ReLU, AV, PReLU, GS, and HH activation functions are invariant to this parameter and do not benefit from it. In practice, the scaling parameter μ is initialized as one and can be updated via any classical stochastic gradient-based method. Throughout our experiments, one scaling factor is learned for each LLS activation function.

6.5 Illustration on Toy Problems

We evaluate the performance of LLS-NNs on a variety of tasks. In this section, we propose two toy experiments to test our framework. The image reconstruction experiments are presented in the next section.

For all experiments, we compare the performance of LLS and the five activation functions: GroupSort (GS), householder (HH), absolute value (AV), RelU and PReLU, see discussion and details on these activations in Chapter 5. For all the experiments, we tune the initialization of PReLU, the group size of GS, and the initialization, range, number of linear regions, and $\mathrm{TV}^{(2)}$ regularization of LLS for best performance. To train the respective networks, we use the Adam optimizer [43] and the default hyperparameters of its PyTorch implementation. The LLS-NNs are optimized with 3 learning rates, one for the weights (with learning rate ξ), one for the scaling parameters (with learning rate $\xi/4$) and one for parameters of the activation functions (with learning rate $\xi/40$). These

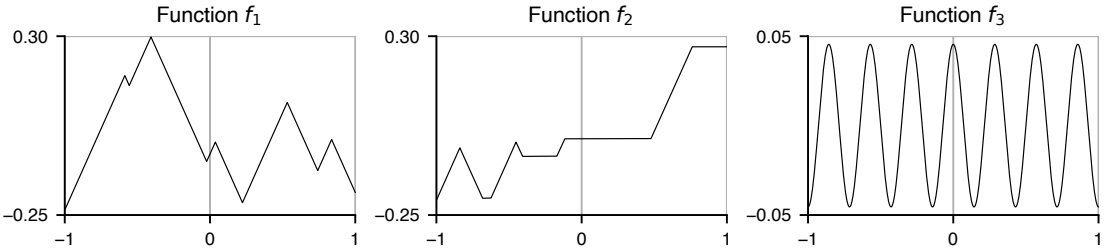

Figure 6.2: Target 1-Lipschitz functions for the first toy problem.

ratios remain fixed throughout this section and, hence, only ξ is going to be stated. Our implementation is available on Github[‡].

6.5.1 One-Dimensional Function Fitting

In this section, LW 1-Lipschitz NNs are trained to approximate 1-Lipschitz functions $f\colon \mathbb{R} \to \mathbb{R}$ on the interval $[-1, 1]$. The aim of this experiment is twofold. First, we want to probe the impact of the two methods to constrain the splines (PG-LLS and R-LLS) on the performance. Second, we want a simple but challenging experiment to compare all activation functions. The three target 1-Lipschitz functions are depicted in Figure 6.2, and we now briefly comment on the choice of these functions. The function f_1 satisfies $|f_1'| = 1$ almost everywhere. Hence, the GNP activation functions are expected to perform well and serve as a baseline against which we compare LLS activation functions. The function f_2 has both portions with $|f_2'| = 1$ and with $f_2' = 0$. It was designed to test the ability of LLS to fit functions with flat portions. Lastly, we benchmark all methods on the highly varying function $f_3 = \sin(7\pi\cdot)/(7\pi)$, which is challenging to fit under Lipschitz constraints.

For each method, we consider two common approaches to learn nonexpansive linear layers, namely orthonormalization, and spectral normalization of the weights. In both cases, we use the mean squared error (MSE) as the loss function. The train loss is computed over 1000 random points sampled uniformly from $[-1, 1]$. The test loss is computed over a uniform partition of $[-1, 1]$ with 10000 points. This experiment lets us assess the expressivity of the models and put the problem generalization aside. For each activation function, we tuned the width and the depth of the neural network for best performance leading to 10 layers and a width of 50 for ReLU-NN; 8 layers and a width of 20 for AV-, PReLU-, and HH-NN; 7 layers and a width of 20 for GS-NNs; 4 layers and a width of 10 for LLS-NNs. Concerning the activation functions, we initialized the PReLU as the absolute value, we used GS with a group size of 5, and the LLS was initialized as ReLU with range $[-0.5, 0.5]$, 100 linear regions, and we set $\eta = 10^{-7}$ for the $\mathrm{TV}^{(2)}$ regularization strength. The learning rate is set to $\xi = 2 \cdot 10^{-3}$ for LLS-NNs and for the other NNs to

[‡]https://github.com/StanislasDucotterd/Lipschitz_DSNN

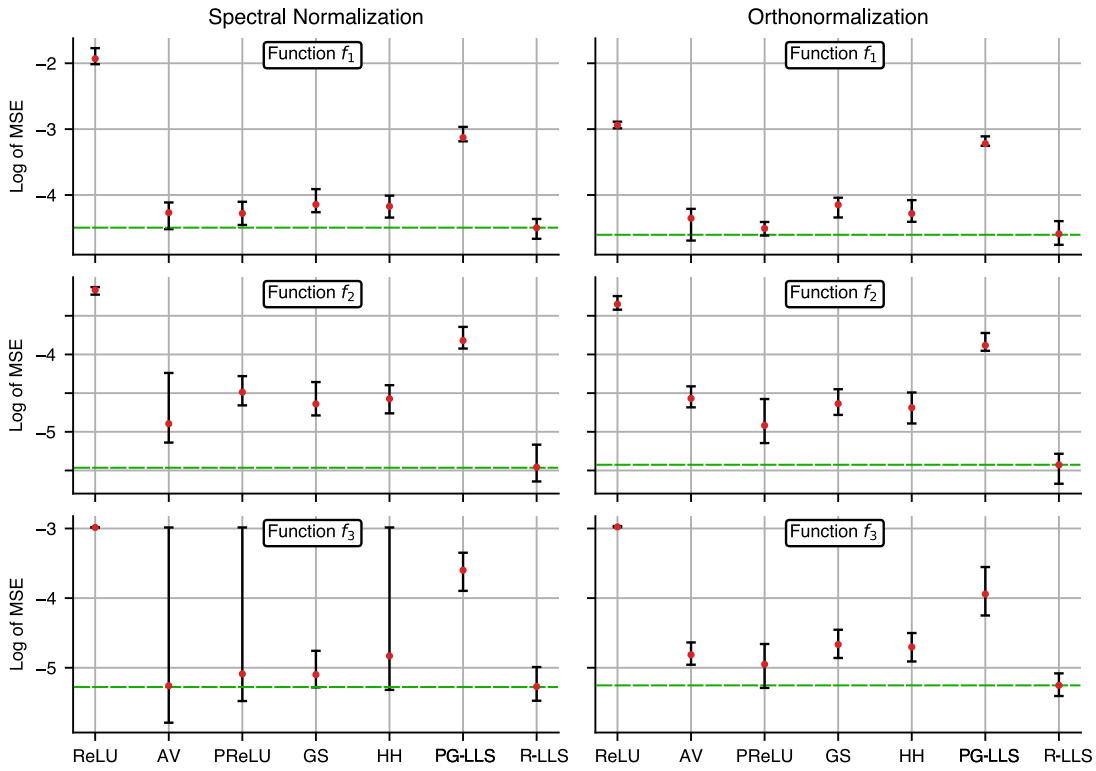

Figure 6.3: Fitting performance for the functions from Figure 6.2. The red markers represent the median performance. The black bars represent the lower and upper quartiles, respectively.

$\xi = 4 \cdot 10^{-3}$ for f_1, f_2 and $\xi = 10^{-3}$ for f_3. All weights were initialized according to the Kaiming initialization [173]. Each NN was trained for 25 independent runs with a batch size of 10 for 1000 epochs. We report the median and the two quartiles of the test losses in Figure 6.3.

For the spectral normalization, we observe that AV, PReLU, and HH tend to get stuck in local minima when fitting f_3 (the associated upper quartile of the MSE loss is quite large). In return, we observe that LLS consistently outperforms the other activation functions in all experiments. Particularly striking is the improvement of R-LLS over PG-LLS, which demonstrates the beneficial role of the reparameterization technique. Accordingly, from now on, we drop PG-LLS and only retain R-LLS.

6.5.2 Wasserstein GAN Training

As the second toy problem, we propose to train Wasserstein GANs to generate MNIST images. To this end, we use the same framework and optimization process as [82], where the discriminators have strict Lipschitz constraints instead of the commonly used gradient penalty. Note that, on the contrary, the generators themselves do not need to be Lipschitz constrained. Thus, we use ReLUs as their activation functions and only plug our different

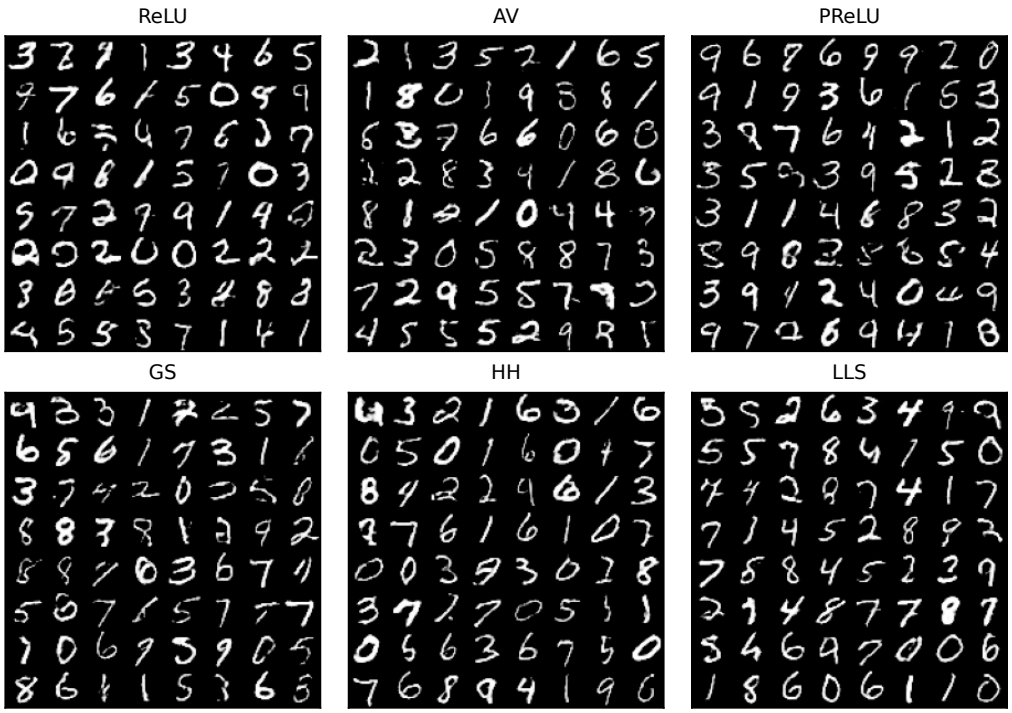

Figure 6.4: Images generated by the Wasserstein GAN models used in the experiments of Section 6.5.2.

Table 6.1: Inception scores for MNIST image generation

	ReLU	AV	PReLU	GS	HH	LLS
Inception score	1.88	2.19	2.13	2.17	2.07	**2.38**

activation functions into the discriminator. Here, after tuning, we used GS with a group size of 2, PReLU with half of the activations initialized as identity and half with absolute value, LLS initialized as the absolute value, range $[-0.1, 0.1]$, 20 linear regions, and $\eta = 10^{-6}$. Note that the spline coefficients only increase the total number of parameters in the neural network by 0.2%. The Wasserstein GANs were trained on the MNIST training set with 60000 images. We report the inception score on the MNIST test set of 10000 images using the implementation from [278] in Table 6.1.

As expected, the limited expressivity of the ReLU leads to the lowest inception score. LLS yields the best score with an improvement over the other schemes. The ability of LLS to generate realistic digits can also be appreciated visually, as illustrated in Figure 6.4.

6.6 Image Reconstruction: from Denoising to MRI and CT

6.6.1 Denoising

The state-of-the-art image denoising architectures [272, 279, 280] are not natively 1-Lipschitz. They rely on dedicated modules designed to improve performance, including skip connections, downsampling and upsampling layers, batch normalization, and attention modules. These can either make it challenging to build provably averaged denoisers or, if they can be easily constrained, their effectiveness remains to be demonstrated in a constrained setting. For this reason, we use a simple CNN architecture without batch normalization.

We train $\boldsymbol{D}$ as a 1-Lipschitz denoiser that is composed of 8 orthogonal convolutional layers parameterized with the BCOP framework [251]. For LLS-NN, we use 64 channels and, to compensate for the additional spline parameters, we use for every other model 68 channels. We use kernels of size (3×3).

The training dataset consists of 238400 patches of size (40×40) taken from the BSD500 dataset [281]. We report the results on the BSD68 test set. All images take values in $[0, 1]$. For our experiment, we add Gaussian noise with $\sigma = 5/255, 10/255, 15/255$ and build one denoiser per noise level. We train all CNNs for 50 epochs with a batch size of 128 and the MSE loss function. The PReLU activation functions were initialized as the absolute value. GS has a group size of 2. The LLS activation functions have $M = 50$ linear regions, a range $\Delta = 0.2$, and were initialized as the identity. We set $\xi = 4 \cdot 10^{-5}$ for every noise level and every model. In this experiment, we also investigated the effect of the $\mathrm{TV}^{(2)}$ regularization parameter η on the performance and the number of linear regions in all the activation functions. The performance results are provided in Table 6.2. As expected, ReLU is doing worse than the other activation functions. For each noise level, LLS is outperforming every other activation function.

The number of linear regions for the LLS activation function $\sigma_{\ell,n}$ is equal to $\frac{1}{T}\|\mathbf{L}\mathbf{c}_{\ell,n}\|_0 + 1$. This metric can lead to an overestimation of the number of linear regions due to numerical imprecisions. Instead, we define the effective number of linear regions as $(|\{1 \leq k \leq K_{\ell,n} : |(\mathbf{L}\mathbf{c}_{\ell,n})_k| > 0.01\}| + 1)$. For each LLS-NN, we report in Table 6.3 the average number of effective linear regions (AELR) of all the LLS activation functions. An AELR close to one indicates that the large majority of neurons perform an affine transform, which corresponds to a simplification of the network. Without regularization, the LLS activation functions have an AELR of 8.07 to 9.24 out of the 50 available linear regions. The $\mathrm{TV}^{(2)}$ regularization drastically sparsifies the LLS activation functions. With $\eta \in [10^{-6}, 10^{-4}]$, the AELR is between 1.07 and 1.44, which is a large decrease without degradation in the denoising performances. For $\eta = 10^{-3}$, the LLS are even further sparsified at the cost of a small loss of performance. We observe a significant loss of performance when η is increased to 10^{-2} where the network is almost an affine mapping. Notice that the AELR

is 2 for ReLU and AV, meaning that LLS outperforms them while being simpler. Another interesting observation is that, despite being very sparse on average, the LLS-NNs with $\eta \in [10^{-6}, 10^{-3}]$ have at least one activation function with at least three linear regions. This suggests that most of the common activation functions might be suboptimal as they have only two linear regions.

Table 6.2: PSNR and SSIM values for the denoising experiment.

Noise level	$\sigma = 5/255$		$\sigma = 10/255$		$\sigma = 15/255$	
Metric	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
ReLU	36.10	0.9386	31.92	0.8735	29.76	0.8203
AV	36.58	0.9499	32.33	0.8889	30.09	0.8375
PReLU	36.58	0.9498	32.25	0.8887	30.11	0.8367
GS	36.54	0.9489	32.23	0.8845	30.11	0.8346
HH	36.47	0.9476	32.25	0.8866	30.11	0.8350
LLS	**36.86**	**0.9546**	**32.55**	**0.8962**	**30.38**	**0.8479**

Table 6.3: PSNR and SSIM values for the denoising experiment with LLS-NNs and various spline regularization strength η.

	$\sigma = 5/255$			$\sigma = 10/255$			$\sigma = 15/255$		
η	PSNR	SSIM	AELR	PSNR	SSIM	AELR	PSNR	SSIM	AELR
0	36.85	0.954	9.24	**32.59**	**0.898**	8.76	30.35	0.846	8.07
10^{-6}	**36.86**	**0.955**	1.21	32.55	0.896	1.24	**30.38**	**0.848**	1.44
10^{-5}	**36.86**	0.954	1.11	32.55	0.896	1.15	30.34	0.846	1.24
10^{-4}	36.82	0.953	1.07	32.57	0.897	1.14	30.36	0.847	1.25
10^{-3}	36.63	0.950	1.02	32.47	0.892	1.06	30.31	0.844	1.10
10^{-2}	35.15	0.914	1.00	32.00	0.878	1.01	29.73	0.816	1.01

6.6.2 Medical Image Reconstruction

The Gaussian denoisers are now plugged into the PnP-FBS algorithm 6.5 to reconstruct medical images. For each setup, we use a relaxed denoiser $\tilde{\boldsymbol{D}} = \beta \boldsymbol{D} + (1 - \beta)\mathbf{Id}$, where the parameter $\beta \in [0, 1)$ is tuned[§] along with the stepsize and noise level $\sigma \in \{5/255, 10/255, 15/255\}$ on a validation set that shall be defined in the sequel. We notice that for every model the optimal β is less than 0.5, meaning that the effective denoiser $\tilde{\boldsymbol{D}}$ is half-averaged and the stability result (6.11) holds.

The goal of this chapter is to compare the various activation functions in Lipschitz-constrained PnP. For comparison with other reconstruction methods, we refer the reader to Chapters 7 and 8, which also cover the same inverse problems.

[§]The hyperparameter tuning is performed with the coarse-to-fine approach detailed in Section 7.8.1

MRI

The ground-truth images for our MRI experiments are proton-density weighted knee MR images from the fastMRI dataset [282] with fat suppression (PDFS) and without fat suppression (PD). They are generated from the fully-sampled k-space data. For each of the two categories (PDFS and PD), we create validation and test sets consisting of 10 and 50 images, respectively, where every image is normalized to have a maximum value of one. To gauge the performance of LLS-NNs in various regimes, we experiment with single- and multi-coil setups with several acceleration factors. In the single-coil setup, we simulate the measurements by masking the Fourier transform of the ground-truth image. In the multi-coil case, we consider 15 coils, and the measurements are simulated by subsampling the Fourier transforms of the multiplication of the ground-truth images with 15 complex-valued sensitivity maps (these were estimated from the raw k-space data using the ESPIRiT algorithm [283] available in the BART toolbox [284]). For both cases, the subsampling in the Fourier domain is performed with a Cartesian mask that is specified by two parameters: the acceleration $M_{\text{acc}} \in \{2, 4, 8\}$ and the center fraction $M_{\text{cf}} = 0.32/M_{\text{acc}}$. A fraction of M_{cf} columns in the center of the k-space (low frequencies) is kept, while columns in the other region of the k-space are uniformly sampled so that the expected proportion of selected columns is $1/M_{\text{acc}}$. In addition, Gaussian noise with standard deviation $\sigma_{\mathbf{n}} = 2 \cdot 10^{-3}$ is added to the real and imaginary parts of the measurements. The PSNR and SSIM values for each method are computed on the (320×320) centered ROI.

The performance in terms of PSNR and SSIM can be found in Tables 6.4 and 6.5 and are summarized in Figures 6.5 and 6.6. Examples of reconstructed images are given in Figures 6.7, 6.8, 6.9 and 6.10. We observe a clear and systematic gap between LLS and the other activation functions in all setups in terms of PSNR and SSIM. In challenging setup, e.g. $M_{\text{acc}} = 8$, it can be observed aliasing artifacts that result from the subsampling in the horizontal direction in Fourier space. These are significantly reduced in the LLS reconstruction.

Table 6.4: Test results for single-coil MRI.

	2-fold				4-fold			
	PSNR		SSIM		PSNR		SSIM	
	PD	PDFS	PD	PDFS	PD	PDFS	PD	PDFS
ReLU	38.15	37.41	0.938	0.918	30.62	31.45	0.818	0.786
AV	38.99	38.05	0.946	0.925	31.34	32.02	0.832	0.797
PReLU	38.97	38.09	0.946	0.925	31.22	32.22	0.832	0.800
GS	38.80	37.92	0.944	0.924	31.27	31.93	0.829	0.796
HH	38.72	37.89	0.944	0.924	31.22	31.94	0.830	0.796
LLS	**40.06**	**38.63**	**0.955**	**0.931**	**32.81**	**33.04**	**0.859**	**0.817**

Table 6.5: Test results for multi-coil MRI.

	4-fold				8-fold			
	PSNR		SSIM		PSNR		SSIM	
	PD	PDFS	PD	PDFS	PD	PDFS	PD	PDFS
ReLU	37.21	37.06	0.929	0.915	31.37	32.57	0.837	0.822
AV	37.81	37.48	0.935	0.919	31.82	32.95	0.845	0.829
PReLU	37.71	37.51	0.934	0.919	31.67	33.11	0.845	0.832
GS	37.76	37.41	0.933	0.919	31.79	32.9	0.843	0.829
HH	37.66	37.39	0.933	0.919	31.68	32.91	0.843	0.829
LLS	**38.68**	**37.96**	**0.943**	**0.924**	**32.75**	**33.61**	**0.859**	**0.835**

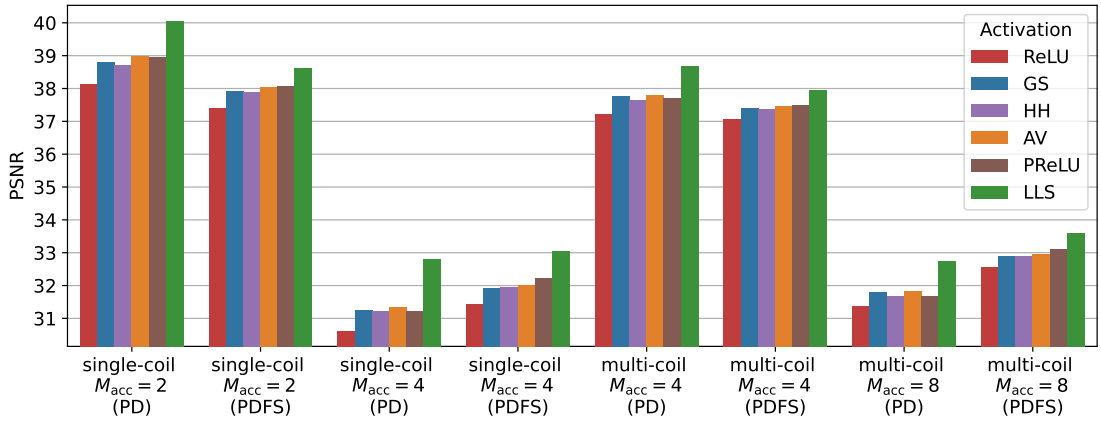

Figure 6.5: Test results for the MRI experiments (PSNR).

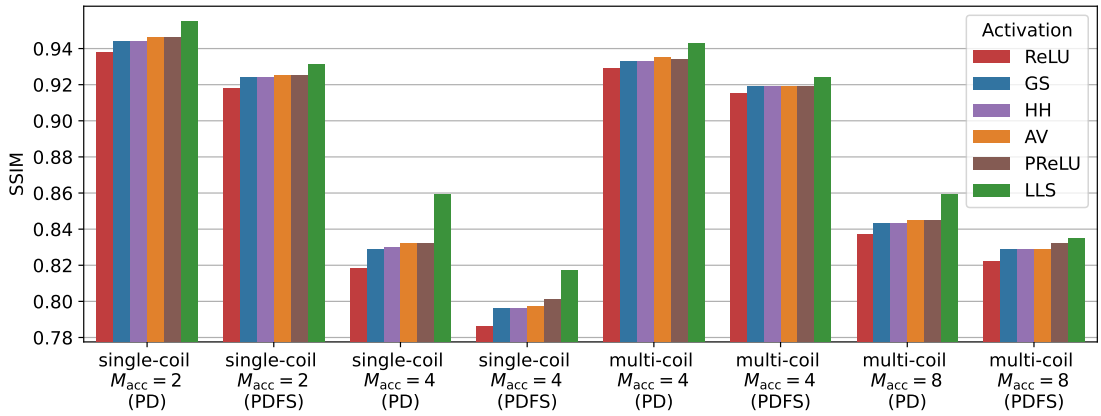

Figure 6.6: Test results for the MRI experiments (SSIM).

CT

For the CT experiment, the data consists of human abdominal CT scans for 10 patients provided by Mayo Clinic for the low-dose CT Grand Challenge [285]. The validation set

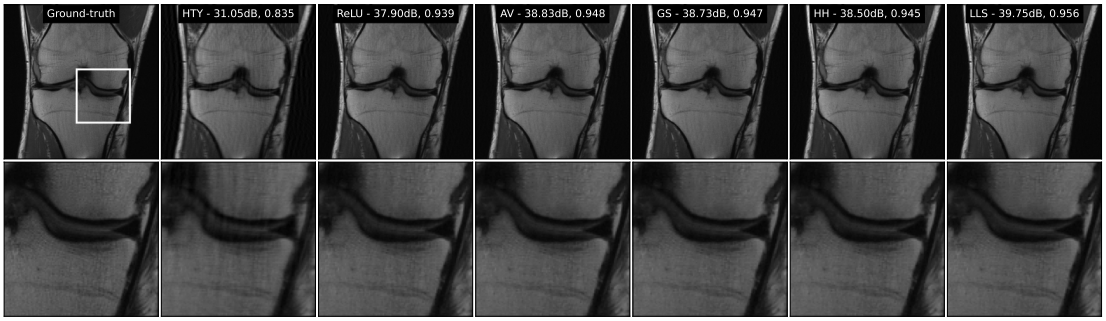

Figure 6.7: Reconstructed images for the MRI experiment (2-fold acceleration with a single coil). The reported metrics are PSNR and SSIM.

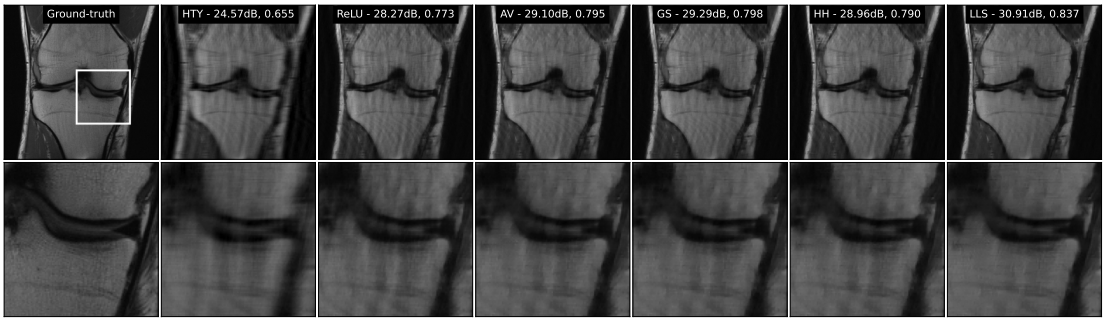

Figure 6.8: Reconstructed images for the MRI experiment (4-fold acceleration with a single coil). The reported metrics are PSNR and SSIM.

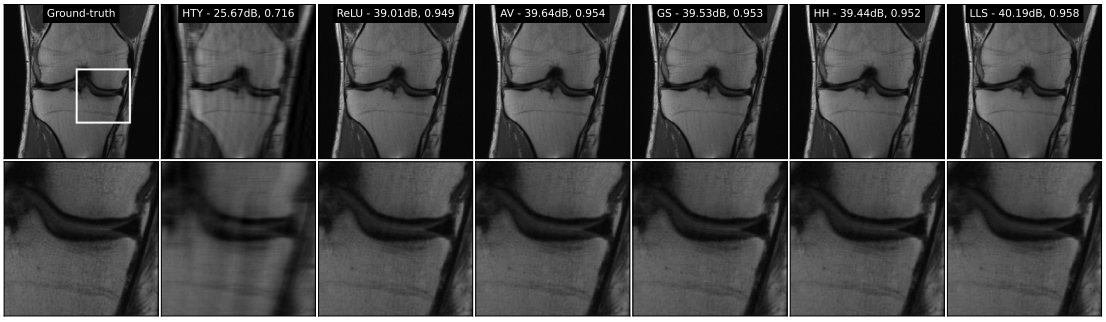

Figure 6.9: Reconstructed images for the MRI experiment (4-fold acceleration with multiple coils). The reported metrics are PSNR and SSIM.

consists of 6 images taken uniformly from the first patient, and for test, we use 128 that correspond to another patient, see [286]. All images are (512×512) and independently normalized in $[0, 1]$. The projections of the data are simulated using a parallel-beam acquisition geometry with 200 angles and 400 detectors. Lastly, Gaussian noise with standard deviation $\sigma_{\mathbf{n}} \in \{0.5, 1, 2\}$ is added to the measurements.

The quantitative results for the CT experiments can be found in Table 6.6 and in Figure 6.11. An example of a reconstructed image is given in Figure 6.12. It can be seen

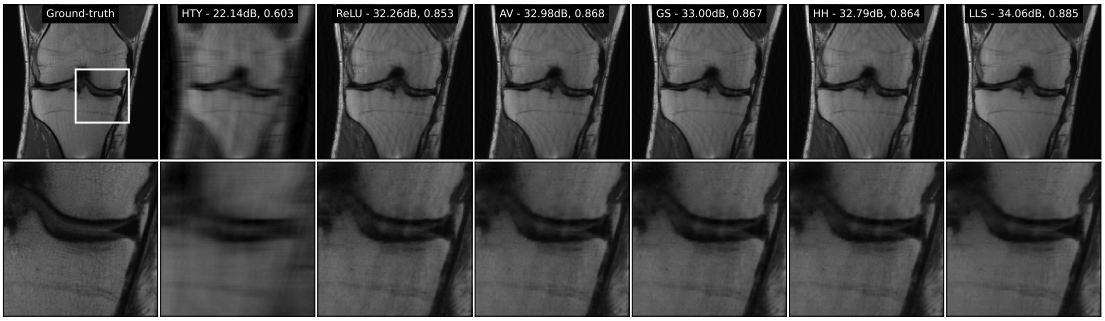

Figure 6.10: Reconstructed images for the MRI experiment (8-fold acceleration with multiple coils). The reported metrics are PSNR and SSIM.

that LLS-NN yields the best performance with some margin over competing frameworks. Visually, LLS-NNs remove more noise than NNs equipped with the other activation functions and lead to fewer aliasing artifacts in challenging setups.

Table 6.6: Test result for the CT experiments.

	$\sigma_{\mathbf{n}}$=0.5		$\sigma_{\mathbf{n}}$=1		$\sigma_{\mathbf{n}}$=2	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
FBP	32.14	0.697	27.05	0.432	21.29	0.204
ReLU	36.94	0.914	33.65	0.860	30.34	0.782
AV	37.15	0.926	34.19	0.885	31.07	0.813
PReLU	37.18	0.927	34.21	0.887	30.87	0.812
GS	36.95	0.920	33.99	0.877	30.87	0.806
HH	36.94	0.918	34.11	0.877	30.92	0.809
LLS	**38.19**	**0.931**	**35.15**	**0.897**	**31.85**	**0.844**

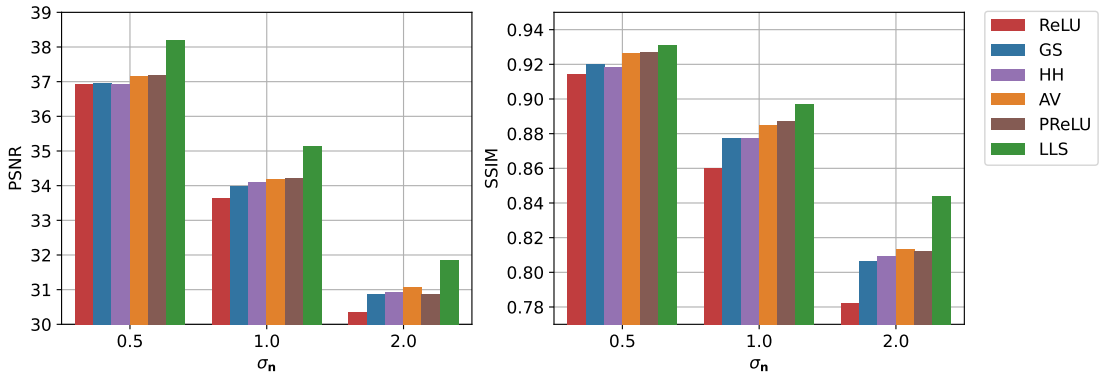

Figure 6.11: Test results for the CT experiments.

The reconstruction performances over the test set are reported in Table 6.6.

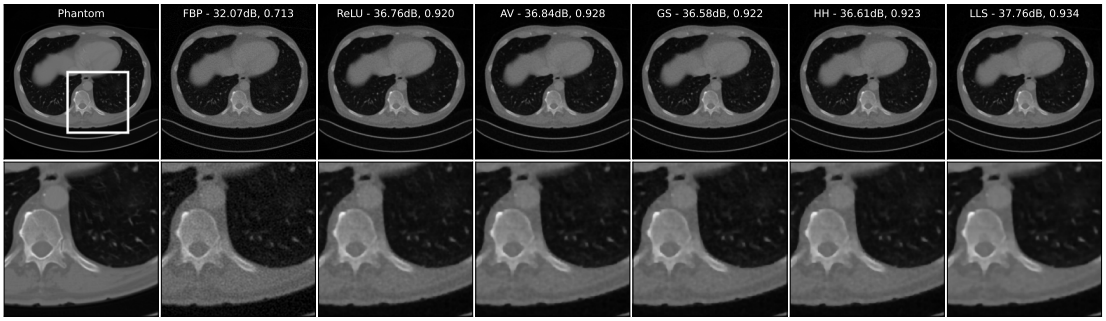

Figure 6.12: Reconstructed images for the CT experiment with $\sigma_{\mathbf{n}} = 0.5$. The reported metrics are PSNR and SSIM.

6.7 Conclusion

In this chapter, we have proposed a framework to efficiently train Lipschitz-constrained neural networks with learnable linear-spline activation functions. Our implementation embeds the Lipschitz constraint on the activation functions directly into the forward pass and adds learnable scaling factors, which preserves the Lipschitz constant of the activation functions and enhances the overall expressivity of the model. Empirically, our approach outperforms Lipschitz-constrained neural networks with activation functions such as GroupSort and Householder, which are the state of the art in this context. These observations are a starting point for the exploration of other architectural constraints and learnable non-component-wise activation functions within the framework of Lipschitz-constrained neural networks.

6.8 Appendix

6.8.1 Properties of the Reconstruction Maps

Stability of PnP-GD

Proof of Proposition 6.1. The reconstructions satisfy the fixed-point equation

$$\mathbf{H}^T\mathbf{H}\mathbf{x}_i^* - \mathbf{H}^T\mathbf{y}_i + \lambda(\mathbf{Id} - \boldsymbol{D}_\sigma)(\mathbf{x}_i^*) = 0, \tag{6.21}$$

for $i = 1, 2$, and where we use the notation $\mathbf{x}_i^* = \boldsymbol{f}_{\mathrm{GD}}(\mathbf{y}_i)$. We subtract (6.21) for $i = 1$ and $i = 2$, and take the inner product with $\mathbf{x}_2^* - \mathbf{x}_1^*$ to infer that

$$\|\mathbf{H}(\mathbf{x}_2^* - \mathbf{x}_1^*)\|_2^2 + \lambda(\|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2^2 - (\mathbf{x}_2^* - \mathbf{x}_1^*)^T(\boldsymbol{D}_\sigma(\mathbf{x}_2^*) - \boldsymbol{D}_\sigma(\mathbf{x}_1^*)) = (\mathbf{x}_2^* - \mathbf{x}_1^*)^T\mathbf{H}^T(\mathbf{y}_2 - \mathbf{y}_1) \tag{6.22}$$

To prove (6.8), we use that $\|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2^2 - (\mathbf{x}_2^* - \mathbf{x}_1^*)^T(\boldsymbol{D}_\sigma(\mathbf{x}_2^*) - \boldsymbol{D}_\sigma(\mathbf{x}_1^*)) \geq 0$ which holds since for a nonexpansive denoiser we have that

$$|(\mathbf{x}_2^* - \mathbf{x}_1^*)^T(\boldsymbol{D}_\sigma(\mathbf{x}_2^*) - \boldsymbol{D}_\sigma(\mathbf{x}_1^*))| \leq \|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2\|\boldsymbol{D}_\sigma(\mathbf{x}_2^*) - \boldsymbol{D}_\sigma(\mathbf{x}_1^*)\|_2 \leq \|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2^2. \tag{6.23}$$

Equation (6.8) then follows from the inequality $(\mathbf{x}_2^* - \mathbf{x}_1^*)\mathbf{H}^T(\mathbf{y}_2 - \mathbf{y}_1) \leq \|\mathbf{H}(\mathbf{x}_2^* - \mathbf{x}_1^*)\|_2\|\mathbf{y}_2 - \mathbf{y}_1\|_2$.

To prove (6.10), we use that

$$(\mathbf{x}_2^* - \mathbf{x}_1^*)^T(\boldsymbol{D}_\sigma(\mathbf{x}_2^*) - \boldsymbol{D}_\sigma(\mathbf{x}_1^*)) \leq \|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2\|\boldsymbol{D}_\sigma(\mathbf{x}_2^*) - \boldsymbol{D}_\sigma(\mathbf{x}_1^*)\|_2 \leq L_\sigma\|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2^2. \tag{6.24}$$

We also use that $\|\mathbf{H}(\mathbf{x}_2^* - \mathbf{x}_1^*)\|_2^2 \geq \lambda_{\min}\|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2^2$ and that $(\mathbf{x}_2^* - \mathbf{x}_1^*)\mathbf{H}^T(\mathbf{y}_2 - \mathbf{y}_1) \leq \|\mathbf{H}\|\|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2\|\mathbf{y}_2 - \mathbf{y}_1\|_2$. □

Stability of PnP-FBS

Proof of Proposition 6.2. The reconstructions satisfy the fixed-point equation

$$\boldsymbol{D}_\sigma((\boldsymbol{I} - \alpha\mathbf{H}^T\mathbf{H})\mathbf{x}_i^* + \alpha\mathbf{H}^T\mathbf{y}_i) = \mathbf{x}_i^*, \tag{6.25}$$

for $i = 1, 2$, and where we use the notation $\mathbf{x}_i^* = \boldsymbol{f}_{\mathrm{FBS}}(\mathbf{y}_i)$.

To prove (6.11), we recall that since $\boldsymbol{D}_\sigma$ is half-averaged, $2\boldsymbol{D}_\sigma - \boldsymbol{I}$ is 1-Lipschitz [274, Proposition 4.4]. We then use this property at the locations $(\boldsymbol{I} - \alpha\mathbf{H}^T\mathbf{H})\mathbf{x}_i^* + \alpha\mathbf{H}^T\mathbf{y}_i$ for $i = 1, 2$ and rely on the fixed-point equation 6.25 to find that

$$\|2(\mathbf{x}_2^* - \mathbf{x}_1^*) - (\boldsymbol{I} - \alpha\mathbf{H}^T\mathbf{H})(\mathbf{x}_2^* - \mathbf{x}_1^*) - \alpha\mathbf{H}^T(\mathbf{y}_2 - \mathbf{y}_1)\|_2^2 \tag{6.26}$$

$$\leq \|(\boldsymbol{I} - \alpha\mathbf{H}^T\mathbf{H})(\mathbf{x}_2^* - \mathbf{x}_1^*) + \alpha\mathbf{H}^T(\mathbf{y}_2 - \mathbf{y}_1)\|_2^2. \tag{6.27}$$

We then develop on both sides and, after simplification find that

$$\|\mathbf{H}(\mathbf{x}_2^* - \mathbf{x}_1^*)\|_2^2 \leq (\mathbf{x}_2^* - \mathbf{x}_1^*)\mathbf{H}^T(\mathbf{y}_2 - \mathbf{y}_1). \tag{6.28}$$

The announced result is then obtained with the Cauchy-Schwarz inequality.

To prove (6.10), we now subtract (6.25) for $i = 1$ and $i = 2$ and use the L_σ-Lipschitzness of the denoiser to find that

$$\begin{aligned} \|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2 &\leq L_\sigma \|(\boldsymbol{I} - \alpha\mathbf{H}^T\mathbf{H})(\mathbf{x}_2^* - \mathbf{x}_1^*) + \alpha\mathbf{H}^T(\mathbf{y}_2 - \mathbf{y}_1)\|_2 && (6.29)\\ &\leq L_\sigma L_\alpha \|\mathbf{x}_2^* - \mathbf{x}_1^*\|_2 + \alpha L_\sigma \|H\| \|\mathbf{y}_2 - \mathbf{y}_1\|_2. && (6.30) \end{aligned}$$

□

6.8.2 Properties of SplineProj

The Least-Square Projection onto $\mathcal{C}_{\text{Lip}}$ Preserves the Mean: Let $\mathbf{x} \in \mathbb{R}^{M+1}$ and $\mathbf{y} \in \mathcal{C}_{\text{Lip}}$. We now expressed the two vectors as $\mathbf{x} = \bar{\mathbf{x}} + \mu_x\mathbf{1}$, $\mathbf{y} = \bar{\mathbf{y}} + \mu_y\mathbf{1}$, where $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ have zero mean. It holds that

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = \|\bar{\mathbf{x}} - \bar{\mathbf{y}} + \mathbf{1}(\mu_x - \mu_y)\|_2^2 = \|\bar{\mathbf{x}} - \bar{\mathbf{y}}\|_2^2 + (\mu_x - \mu_y)^2(M+1)^2. \tag{6.31}$$

Hence, we can add $(\mu_x - \mu_y)\mathbf{1}$ to $\mathbf{y}$ and decrease the distance without violating the constraints.

SplineProj Maps $\mathbb{R}^{M+1}$ to $\mathcal{C}_{\text{Lip}}$: We have, for any $\mathbf{c} \in \mathbb{R}^{M+1}$, that

$$\|\mathbf{D}\,\text{SplineProj}(\mathbf{c})\|_\infty = \|\mathbf{D}\mathbf{D}^\dagger\,\text{Clip}_\Delta(\mathbf{D}\mathbf{c}) + \mathbf{D}\mathbf{1}\frac{1}{M+1}\mathbf{1}^T\mathbf{c}\|_\infty = \|\,\text{Clip}_\Delta(\mathbf{D}\mathbf{c})\|_\infty \leq \Delta. \tag{6.32}$$

Here, we used the fact that $\mathbf{D}\mathbf{D}^\dagger$ is the identity matrix in $\mathbb{R}^{M,M}$ and that $\mathbf{D}\mathbf{1} = \mathbf{0}$.

SplineProj is a Projection: Using the same properties as above, it holds that

$$\begin{aligned} \text{SplineProj}(\text{SplineProj}(\mathbf{c})) &= \mathbf{D}^\dagger\,\text{Clip}_\Delta\left(\mathbf{D}\mathbf{D}^\dagger\,\text{Clip}_\Delta(\mathbf{D}\mathbf{c}) + \frac{\mathbf{D}\mathbf{1}\mathbf{1}^T\mathbf{c}}{M+1}\right) + \frac{\mathbf{1}\mathbf{1}^T\mathbf{c}}{M+1}\\ &= \mathbf{D}^\dagger\,\text{Clip}_\Delta(\text{Clip}_\Delta(\mathbf{D}\mathbf{c})) + \frac{\mathbf{1}\mathbf{1}^T}{M+1}\mathbf{c}\\ &= \mathbf{D}^\dagger\,\text{Clip}_\Delta(\mathbf{D}\mathbf{c}) + \frac{\mathbf{1}\mathbf{1}^T}{M+1}\mathbf{c} = \text{SplineProj}(\mathbf{c}). \end{aligned} \tag{6.33}$$

SplineProj Preserves the Mean of c: From the properties of the Moore-Penrose inverse, we have that $\ker((\mathbf{D}^\dagger)^T) = \ker(\mathbf{D})$, therefore, $\mathbf{1}^T\mathbf{D}^\dagger = \mathbf{0}$ and

$$\mathbf{1}^T \operatorname{SplineProj}(\mathbf{c}) = \mathbf{1}^T\mathbf{D}^\dagger \operatorname{Clip}_\Delta(\mathbf{Dc}) + \mathbf{1}^T\mathbf{1}\frac{1}{M+1}\mathbf{1}^T\mathbf{c} = \mathbf{1}^T c_k. \tag{6.34}$$

SplineProj is Differentiable Almost Everywhere with Respect to c: The $\operatorname{Clip}_\Delta$ function is differentiable everywhere except at T and $-T$. Therefore, the operation $\mathbf{D}^\dagger \operatorname{Clip}_\Delta(\mathbf{Dc})$ is differentiable everywhere except on

$$S = \bigcup_{k=1}^{M} \Big\{ \mathbf{x} \in \mathbb{R}^{M+1} \colon |(\mathbf{Dx})_k| = \Delta \Big\}. \tag{6.35}$$

The set S is a union of $2M$ hyperplanes (with dimension M). Hence, S has measure zero in $\mathbb{R}^{M+1}$.

6.8.3 Scale Invariance of $\mathrm{TV}^{(2)}$

For $\mu \neq 0$ and any $\sigma \in \mathrm{BV}^{(2)}(\mathbb{R})$, it holds that $\tilde{\sigma} = \sigma(\mu\cdot)/\mu \in \mathrm{BV}^{(2)}(\mathbb{R})$ satisfies that

$$\begin{aligned}\mathrm{TV}^{(2)}(\tilde{\sigma}) &= \sup_{\varphi \in \mathcal{S}(\mathbb{R}) : \|\varphi\|_\infty \leq 1} \int_{\mathbb{R}} (\sigma(\mu x)/\mu)\varphi''(x)\mathrm{d}x = \sup_{\varphi \in \mathcal{S}(\mathbb{R}) : \|\varphi\|_\infty \leq 1} \int_{\mathbb{R}} \sigma(x)(\varphi''(x/\mu)/\mu^2)\mathrm{d}x \\ &= \sup_{\varphi \in \mathcal{S}(\mathbb{R}) : \|\varphi\|_\infty \leq 1} \int_{\mathbb{R}} \sigma(x)\varphi''(\cdot/\mu)(x)\mathrm{d}x = \mathrm{TV}^{(2)}(\sigma),\end{aligned} \tag{6.36}$$

namely, that $\mathrm{TV}^{(2)}$ is invariant with respect to this scaling.

Part III From Convex to Weakly-convex Data-driven Regularization

The previous part of this thesis was concerned with plug-and-play (PnP) methods that deploy convolutional neural network (CNN) denoisers as regularization engines to solve image reconstruction tasks. Such methods perform implicit regularization because there might not exist an underlying regularization function. In this part, we propose to move from implicit to explicit regularization through the learning of regularizers.

The frameworks introduced in this part share common features with the one of Part II. First, they comply with the plug-and-play philosophy and build a regularization prior based on an image-denoising task and, second, they rely on our learnable-spline toolbox. In Chapter 7, the learned regularizer is constrained to be convex. While it can be shown that the reconstruction algorithm is a special instance of PnP with a nonexpansive regularization operator, the interpretability is greatly improved, and, remarkably, the reconstruction quality is also enhanced. In Chapter 8, the convexity constraint is slightly relaxed to learn weakly convex regularizers. We then propose a theoretical and experimental perspective to show that such regularizers offer an excellent tradeoff between performance, number of parameters, guarantees, and interpretability when compared to other data-driven approaches.

7 A Neural-Network-Based Convex Regularizer for Image Reconstruction

The text of this chapter is adapted from the published article

A. Goujon, S. Neumayer, P. Bohra, S. Ducotterd and M. Unser "A neural-network-based convex regularizer for inverse problems", *IEEE Transactions on Computational Imaging*, volume 9, page 781-795, August 2023.

Summary

The emergence of deep-learning-based methods to solve image-reconstruction problems has enabled a significant increase in quality. Unfortunately, these new methods often lack reliability and explainability, and there is a growing interest in addressing these shortcomings while retaining the boost in performance. In this chapter, we tackle this issue by revisiting regularizers which are the sum of convex-ridge functions. The gradient of such regularizers is parameterized by a neural network that has a single hidden layer with increasing and learnable activation functions. This neural network is trained within a few minutes as a multistep Gaussian denoiser. The numerical experiments for denoising, CT, and MRI reconstruction show improvements over methods that offer similar reliability guarantees.

7.1 Introduction

7.1.1 Linear Inverse Problems

In natural science, it is common to indirectly probe an object of interest by collecting a series of linear measurements [18]. After discretization, this can be formalized as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{7.1}$$

where $\mathbf{H} \in \mathbb{R}^{m \times d}$ acts on the discrete representation $\mathbf{x} \in \mathbb{R}^d$ of the object and models the physics of the process. The vector $\mathbf{n} \in \mathbb{R}^m$ accounts for additive noise in the measurements. Given the measurement vector $\mathbf{y} \in \mathbb{R}^m$, the task is then to reconstruct $\mathbf{x}$. Many medical-imaging applications fit into this class of inverse problems [16], including magnetic resonance imaging (MRI) and X-ray computed tomography (CT).

In addition to the presence of noise, which makes the reconstruction challenging for ill-conditioned $\mathbf{H}$, it is common to have only a few measurements ($m < d$), resulting in underdetermined problems. In either case, (7.1) is ill-posed, and solving it poses serious challenges. To overcome this issue, a reconstruction $\mathbf{x}^*$ is often computed as

$$\mathbf{x}^* \in \operatorname*{arg\,min}_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + R(\mathbf{x}), \tag{7.2}$$

where $R\colon \mathbb{R}^d \to \mathbb{R}$ is a convex regularizer that incorporates prior information about $\mathbf{x}$ to counteract the ill-posedness of (7.1). Popular choices are the Tikhonov [19] or total-variation (TV) [22, 30, 264] regularizers.

7.1.2 Deep-Learning Methods

Deep-learning-based methods have emerged in the past years for the inversion of (7.1) in a variety of applications; see [287, 288] for an overview. Such approaches offer a significantly improved quality of reconstruction as compared to classical variational models of the form (7.2). Unfortunately, most of them are not well understood and lack stability guarantees [57, 289].

For end-to-end approaches, a pre-trained model outputs a reconstruction directly from the measurements $\mathbf{y}$ or from a low-quality reconstruction [33, 34, 184, 290, 291]. These approaches are often much faster than iterative solvers that compute (7.2). Their downside is that they offer no control of the data-consistency term $\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2$. In addition, they are less universal since a model is specifically trained per $\mathbf{H}$ and per noise model. End-to-end learning can also lead to serious stability issues [57].

A remedy for some of these issues is provided by the convolutional-neural-network (CNN) variants of the plug-and-play (PnP) framework [81, 89, 90, 93]. PnP methods are discussed at length in Chapter 6, and we now simply give a summary. The inspiration for PnP methods comes from the interpretation of the proximal operator

$$\operatorname{prox}_R(\mathbf{y}) = \operatorname*{arg\,min}_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 + R(\mathbf{x}) \tag{7.3}$$

used in many iterative algorithms for the computation of (7.2) as a denoiser. The idea is to replace (7.3) with a more powerful CNN-based denoiser $\boldsymbol{D}$. However, $\boldsymbol{D}$ is usually not a proper proximal operator, and the convergence of the PnP iterates is not guaranteed. It

was shown in [81] that, for an invertible $\mathbf{H}$, convergence can be ensured by constraining the Lipschitz constant of the residual operator $(\mathbf{Id} - \boldsymbol{D})$, where $\mathbf{Id}$ is the identity operator. For a noninvertible $\mathbf{H}$, this constraint, however, does not suffice. Instead, one can constrain $\boldsymbol{D}$ to be an averaged operator which, unfortunately, degrades the performance [189]. Hence, in practice, one usually only constrains $(\mathbf{Id} - \boldsymbol{D})$, even if the framework is deployed for noninvertible $\mathbf{H}$ [81, 83, 249]. While this results in good performances, it leaves a gap between theory and implementation. Following a different route, one can also ensure convergence with relaxed algorithms [94, 102]. There, $\boldsymbol{D}$ is replaced with the relaxed version $\gamma\boldsymbol{D} + (1-\gamma)\mathbf{Id}$, $\gamma \in (0, 1]$. At each iteration, γ is decreased if some condition is violated. Unfortunately, without particular constraints on $\boldsymbol{D}$, the evolution of γ is unpredictable. Hence, the associated fixed-point equation for the reconstruction is unknown a priori, which reduces the reliability of the method.

Another data-driven approach arising from (7.2) is the learning of R instead of prox_R. Pioneering work in this direction includes the *fields of experts* [292–294], where R is parameterized by an interpretable and shallow model, namely, a sum of nonlinear one-dimensional functions composed with convolutional filters. Some recent approaches rely on more sophisticated architectures with much deeper CNNs, such as with the adversarial regularization (AR) [295, 296], NETT [297], and the total-deep-variation frameworks [298], or with regularizers for which a proximal operator exists [76, 102, 273, 299]. There exists a variety of strategies to learn R, including bilevel optimization [293], unrolling [294, 298], gradient-step denoising [102, 273], and adversarial training [295, 296]. When R is convex, a global minimizer of (7.2) can be found under mild assumptions. As the relaxation of the convexity constraint usually boosts the performance [293, 300], it is consequently the most popular approach. Unfortunately, one can then expect convergence only to a critical point.

7.1.3 The Quest for Trustworthiness

In many sensitive applications such as medical imaging, there is a growing interest to improve the trustworthiness of the reconstruction methods. The available frameworks used to learn a (pseudo) proximal operator or regularizer result in a variety of neural architectures that differ in the importance attributed to the following competing properties:

- good reconstruction quality;
- independence on $\mathbf{H}$, noise model, and image domain;
- convergence guarantees and properties of the fixed points of the reconstruction algorithm;
- interpretability, which can include the existence of an explicit cost or a minimal understanding of what the regularizer is promoting.

To foster the last two properties, one usually has to impose structural constraints on the learned regularizer/proximal operator. For instance, within the PnP framework, there have been some recent efforts to improve the expressivity of averaged denoisers, either with strict Lipschitz constraints on the model, [189, 301] or with regularization of its Lipchitz constant during training [75, 76] which, in turn, improves the convergence properties of the reconstruction algorithm. In the same vein, the authors of [286, 302] proposed to learn a convex R parameterized by a deep input convex neural network (ICNN)[303] and to train it within an adversarial framework as in [295].

In the present chapter, we prioritize the reliability and interpretability of the method. Thus, we revisit the family of learnable convex-ridge regularizers [292–294, 300, 304]

$$R\colon \mathbf{x} \mapsto \sum_i \psi_i(\mathbf{w}_i^T \mathbf{x}), \tag{7.4}$$

where the profile functions $\psi_i\colon \mathbb{R} \to \mathbb{R}$ are convex, and $\mathbf{w}_i \in \mathbb{R}^d$ are learnable weights. A popular way to learn R is to solve a non-convex bilevel optimization task [305, 306] for a given inverse problem. It was reported in [293] that these learned regularizers outperform the popular TV regularizer for image reconstruction. As bilevel optimization is computationally quite intensive, it was proposed in [300] to unroll the forward-backward splitting (FBS) algorithm applied to (7.2) with a regularizer of the form (7.4). Accordingly, R is optimized so that a predefined number t of iterations of the FBS algorithm yields a good reconstruction. Unfortunately, on a denoising task with learnable profiles ψ_i, the proposed approach does not match the performance of the bilevel optimization.

To deal with these shortcomings, we introduce an efficient framework* to learn some R of the form (7.4) with free-form convex profiles. We train this R on a generic denoising task and then plug it into (7.2). This yields a generic reconstruction framework that is applicable to a variety of inverse problems. The main contributions of the present chapter are as follows.

- **Interpretable and Expressive Model:** We use a one-hidden-layer neural network (NN) with learnable increasing linear-spline activation functions to parameterize $\boldsymbol{\nabla} R$. We prove that this yields the maximal expressivity in the generic setting (7.4).
- **Embedding of the Constraints into the Forward Pass:** The structural constraints on $\boldsymbol{\nabla} R$ are embedded into the forward pass during the training. This includes an efficient procedure to enforce the convexity of the profiles, and the computation of a bound on the Lipschitz constant of $\boldsymbol{\nabla} R$, which is required for our training procedure.
- **Ultra-Fast Training:** The regularizer R is learned via the training of a multi-

*All experiments can be reproduced with the code published at https://github.com/axgoujon/convex_ridge_regularizers

gradient-step denoiser. Empirically, we observe that a few gradient steps suffice to learn a best-performing R. This leads to training within a few minutes.

- **Best Reconstruction Quality in a Constrained Scenario:** We show that our framework outperforms recent deep-learning-based approaches with comparable guarantees and constraints in two popular medical imaging modalities (CT and MRI). This includes the PnP method with averaged denoisers and a variational framework with a learnable deep convex regularizer. This even holds for a strong mismatch in the noise level used for the training and the one found in the inverse problem.

7.2 Architecture of the Regularizer

In this section, we introduce the notions required to define the convex-ridge regularizer neural network (CRR-NN).

7.2.1 General Setting

Our goal is to learn a regularizer R for the variational problem (7.2) that performs well across a variety of ill-posed problems. Similar to the PnP framework, we view the denoising task

$$\mathbf{x}^* = \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + \lambda R(\mathbf{x}) \tag{7.5}$$

as the underlying base problem for training, where $\mathbf{y}$ is the noisy image. Since we prioritize interpretability and reliability, we choose the simple convex-ridge regularizer (7.4) and use its convolutional form. More precisely, the regularity of an image x is measured as

$$R\colon x \mapsto \sum_{i=1}^{N_C} \sum_{\mathbf{k}\in\mathbb{Z}^2} \psi_i\Big((h_i * x)[\mathbf{k}]\Big), \tag{7.6}$$

where h_i is the impulse response of a 2D convolutional filter, $(h_i * x)[\mathbf{k}]$ is the value of the $\mathbf{k}$-th pixel of the filtered image $h_i * x$, and N_C is the number of channels. In the sequel, we mainly view the (finite-size) image x as the (finite-dimensional) vector $\mathbf{x} \in \mathbb{R}^d$, and since (7.6) is a special case of (7.4), we henceforth use the generic form (7.4) to simplify the notations. We use the notation $R_{\boldsymbol{\theta}}$ to express the dependence of R on the aggregated set of learnable parameters $\boldsymbol{\theta}$, which will be specified when necessary. From now on, we assume that the convex profiles ψ_i have Lipschitz continuous derivatives, i.e. $\psi_i \in C^{1,1}(\mathbb{R})$.

7.2.2 Gradient-Step Neural Network

Given the assumptions on $R_{\boldsymbol{\theta}}$, the denoised image in (7.5) can be interpreted as the unique fixed point of $\boldsymbol{T}_{R_{\boldsymbol{\theta}},\lambda,\alpha}\colon \mathbb{R}^d \to \mathbb{R}^d$ defined by

$$\boldsymbol{T}_{R_{\boldsymbol{\theta}},\lambda,\alpha}(\mathbf{x}) = \mathbf{x} - \alpha\Big((\mathbf{x}-\mathbf{y}) + \lambda \boldsymbol{\nabla} R_{\boldsymbol{\theta}}(\mathbf{x})\Big). \tag{7.7}$$

Iterations of the operator (7.7) implement a gradient descent with stepsize α, which converges if $\alpha \in (0, 2/(1+\lambda L_{\boldsymbol{\theta}}))$, where $L_{\boldsymbol{\theta}} = \mathrm{Lip}(\boldsymbol{\nabla} R_{\boldsymbol{\theta}})$ is the Lipschitz constant of $\boldsymbol{\nabla} R_{\boldsymbol{\theta}}$. In the sequel, we always enforce this constraint on α. The gradient of the generic convex-ridge expression (7.4) is given by

$$\boldsymbol{\nabla} R_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{W}^T \boldsymbol{\sigma}(\mathbf{W}\mathbf{x}), \tag{7.8}$$

where $\mathbf{W} = [\mathbf{w}_1 \cdots \mathbf{w}_p]^T \in \mathbb{R}^{p\times d}$ and $\boldsymbol{\sigma}$ is a pointwise activation function whose components $(\sigma_i = \psi_i')_{i=1}^p$ are Lipschitz continuous and increasing. In our implementation, the activation functions σ_i are shared within each channel of $\mathbf{W}$. The resulting gradient-step operator

$$\boldsymbol{T}_{R_{\boldsymbol{\theta}},\lambda,\alpha}(\mathbf{x}) = (1-\alpha)\mathbf{x} + \alpha\Big(\mathbf{y} - \lambda \mathbf{W}^T \boldsymbol{\sigma}(\mathbf{W}\mathbf{x})\Big) \tag{7.9}$$

corresponds to a one-hidden-layer convolutional NN with a bias and a skip connection. We refer to it as a *gradient-step NN*. The training of a gradient-step NN will give a CRR-NN.

7.3 Characterization of Good Profile Functions

In this section, we provide theoretical results to motivate our choice of the profiles ψ_i or, equivalently, of their derivatives $\sigma_i = \psi_i'$. This will lead us to the implementation presented in Section 7.4.

7.3.1 Existence of Minimizers and Stability of the Reconstruction

The convexity of $R_{\boldsymbol{\theta}}$ is not sufficient to ensure that the solution set in (7.2) is nonempty for a noninvertible forward matrix $\mathbf{H}$. With convex-ridge regularizers, this shortcoming can be addressed under a mild condition on the functions ψ_i (Proposition 7.1). The implications for our implementation are detailed in Section 7.4.2.

Proposition 7.1. *Let $\mathbf{H} \in \mathbb{R}^{m\times d}$ and $\psi_i\colon \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, p$, be convex functions. If $\arg\min_{t\in\mathbb{R}} \psi_i(t) \neq \emptyset$ for all $i = 1, \ldots, p$, then*

$$\emptyset \neq \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + \sum_{i=1}^{p} \psi_i(\mathbf{w}_i^T \mathbf{x}). \tag{7.10}$$

Proof. Set $S_i = \arg\min_{t\in\mathbb{R}} \psi_i(t)$. Then, each ridge $\psi_i(\mathbf{w}_i^T\cdot)$ partitions $\mathbb{R}^d$ into the three (possibly empty) convex polytopes

- $\Omega_0^i = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}_i^T\mathbf{x} \in S_i\}$;
- $\Omega_1^i = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}_i^T\mathbf{x} \leq \inf S_i\}$;
- $\Omega_2^i = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}_i^T\mathbf{x} \geq \sup S_i\}$.

Based on these, we partition $\mathbb{R}^d$ into finitely many polytopes of the form $\bigcap_{i=1}^p \Omega_{m_i}^i$, where $m_i \in \{0, 1, 2\}$. The infimum of the objective in (7.10) must be attained in at least one of these polytopes, say, $P = \bigcap_{i=1}^p \Omega_{m_i}^i$.

Now, we pick a minimizing sequence $(\mathbf{x}_k)_{k\in\mathbb{N}} \subset P$. Let $\mathbf{M}$ be the matrix whose rows are the rows of $\mathbf{H}$ and the $\mathbf{w}_i^T$ with $m_i \neq 0$. Due to the coercivity of $\|\cdot\|_2^2$, we get that $\mathbf{H}\mathbf{x}_k$ remains bounded. As the ψ_i are convex, they are coercive on the intervals $(-\infty, \inf S_i]$ and $[\sup S_i, +\infty)$ and, hence, $\mathbf{w}_i^T\mathbf{x}_k$ also remains bounded. Therefore, the sequence $(\mathbf{M}\mathbf{x}_k)_{k\in\mathbb{N}}$ is bounded and we can drop to a convergent subsequence with limit $\mathbf{u} \in \mathrm{ran}(\mathbf{M})$. The associated set

$$Q = \{\mathbf{x} \in \mathbb{R}^d \colon \mathbf{M}\mathbf{x} = \mathbf{u}\} = \{\mathbf{M}^\dagger\mathbf{u}\} + \ker(\mathbf{M}) \tag{7.11}$$

is a closed polytope. It holds that

$$\begin{aligned} \mathrm{dist}(\mathbf{x}_k, Q) &= \mathrm{dist}\Big(\mathbf{M}^\dagger\mathbf{M}\mathbf{x}_k + \mathrm{P}_{\ker(\mathbf{M})}(\mathbf{x}_k), Q\Big) \\ &\leq \mathrm{dist}(\mathbf{M}^\dagger\mathbf{M}\mathbf{x}_k, \mathbf{M}^\dagger\mathbf{u}) \to 0 \end{aligned} \tag{7.12}$$

as $k \to +\infty$ and, thus, that $\mathrm{dist}(P, Q) = 0$. The distance of the closed polytopes P and Q is 0 if and only if $P \cap Q \neq \emptyset$ [307, Theorem 1]. Note that $\psi_i(\mathbf{w}_i^T\cdot)$ is constant on P if $m_i = 0$. Hence, any $\mathbf{x} \in P \cap Q$ is a minimizer of (7.10). □

The proof of Proposition 7.1 directly exploits the properties of ridge functions. Whether it is possible to extend the result to more complex or even generic convex regularizers is not known to the authors. The assumption in Proposition 7.1 is rather weak as neither the cost function nor the one-dimensional profiles ψ_i need to be coercive. The existence of a solution for Problem (7.2) is a key step towards the stability of the reconstruction map in the measurement domain, which is given in Proposition 7.2.

Proposition 7.2. *Let* $\mathbf{H} \in \mathbb{R}^{m\times d}$ *and* $\psi_i \colon \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, p$, *be convex, continuously differentiable functions with* $\arg\min_{t\in\mathbb{R}} \psi_i(t) \neq \emptyset$. *For any* $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^m$ *let*

$$\mathbf{x}_q \in \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}_q\|_2^2 + \sum_{i=1}^p \psi_i(\mathbf{w}_i^T\mathbf{x}) \tag{7.13}$$

with $q = 1, 2$ be the corresponding reconstructions. Then,

$$\|\mathbf{H}\mathbf{x}_1 - \mathbf{H}\mathbf{x}_2\|_2 \leq \|\mathbf{y}_1 - \mathbf{y}_2\|_2. \tag{7.14}$$

Proof. Proposition 7.1 guarantees the existence of $\mathbf{x}_q$. Since the objective in (7.10) is smooth, it holds that $\mathbf{H}^T(\mathbf{H}\mathbf{x}_q - \mathbf{y}_q) + \boldsymbol{\nabla} R(\mathbf{x}_q) = \mathbf{0}$. From this, we infer that

$$\mathbf{H}^T\mathbf{H}(\mathbf{x}_1 - \mathbf{x}_2) + (\boldsymbol{\nabla} R(\mathbf{x}_1) - \boldsymbol{\nabla} R(\mathbf{x}_2)) = \mathbf{H}^T(\mathbf{y}_1 - \mathbf{y}_2). \tag{7.15}$$

Taking the inner product with $(\mathbf{x}_1 - \mathbf{x}_2)$ on both sides gives

$$\begin{aligned} &\|\mathbf{H}\mathbf{x}_1 - \mathbf{H}\mathbf{x}_2\|_2^2 + (\mathbf{x}_1 - \mathbf{x}_2)^T(\boldsymbol{\nabla} R(\mathbf{x}_1) - \boldsymbol{\nabla} R(\mathbf{x}_2)) \\ =&(\mathbf{H}(\mathbf{x}_1 - \mathbf{x}_2))^T(\mathbf{y}_1 - \mathbf{y}_2). \end{aligned} \tag{7.16}$$

To conclude, we use the fact that the gradient of a convex map is monotone, i.e. $(\mathbf{x}_1 - \mathbf{x}_2)^T(\boldsymbol{\nabla} R(\mathbf{x}_1) - \boldsymbol{\nabla} R(\mathbf{x}_2)) \geq 0$, and apply the Cauchy-Schwarz inequality to estimate

$$(\mathbf{H}(\mathbf{x}_1 - \mathbf{x}_2))^T(\mathbf{y}_1 - \mathbf{y}_2) \leq \|\mathbf{H}\mathbf{x}_1 - \mathbf{H}\mathbf{x}_2\|\|\mathbf{y}_1 - \mathbf{y}_2\|. \tag{7.17}$$

□

7.3.2 Expressivity of Profile Functions

The gradient-step NN $\boldsymbol{T}_{R_{\boldsymbol{\theta}},\lambda,\alpha}$ introduced in (7.9) is the key component of our training procedure. Here, we investigate its expressivity depending on the choice of the activation functions σ_i used to parametrize $\boldsymbol{\nabla} R_{\boldsymbol{\theta}}$.

Let $C^{0,1}_{\uparrow}(\mathbb{R})$ be the set of scalar Lipschitz-continuous and increasing functions on $\mathbb{R}$, and let $\mathcal{LS}^m_{\uparrow}(\mathbb{R})$ be the subset of increasing linear splines with at most m knots. We also define

$$\mathcal{E}(\mathbb{R}^d) = \left\{\mathbf{W}^T\boldsymbol{\sigma}(\mathbf{W}\cdot) : \mathbf{W} \in \mathbb{R}^{p\times d}, \sigma_i \in C^{0,1}_{\uparrow}(\mathbb{R})\right\} \tag{7.18}$$

and, further, for any $\Omega \subset \mathbb{R}^d$,

$$\mathcal{E}(\Omega) = \left\{\boldsymbol{f}|_{\Omega} : \boldsymbol{f} \in \mathcal{E}(\mathbb{R}^d)\right\}. \tag{7.19}$$

In the following, we set $\|\boldsymbol{f}\|_{C(\Omega)} \coloneqq \sup_{\mathbf{x}\in\Omega}\|\boldsymbol{f}(\mathbf{x})\|$ and $\|\boldsymbol{f}\|_{C^1(\Omega)} \coloneqq \sup_{\mathbf{x}\in\Omega}\|\boldsymbol{f}(\mathbf{x})\| + \sup_{\mathbf{x}\in\Omega}\|\boldsymbol{J}_{\boldsymbol{f}}(\mathbf{x})\|$.

The popular ReLU activation function is Lipschitz-continuous and increasing. Unfortunately, it comes with limited expressivity, as shown in Proposition 7.3.

Proposition 7.3. *Let $\Omega \subset \mathbb{R}^d$ be compact with a nonempty interior. Then, the set*

$$\left\{\mathbf{W}^T\mathrm{ReLU}(\mathbf{W}\cdot - \mathbf{b}) \colon \mathbf{W} \in \mathbb{R}^{p\times d}, \mathbf{b} \in \mathbb{R}^p\right\} \tag{7.20}$$

is not dense with respect to $\|\cdot\|_{C(\Omega)}$ *in* $\mathcal{E}(\Omega)$.

Proof. Since Ω has a nonempty interior, there exists $\mathbf{v} \in \mathbb{R}^d$ with $\|\mathbf{v}\|_2 = 1$, $a \in \mathbb{R}$, and $\delta > 0$ such that for $\boldsymbol{l}_{\mathbf{v}} \colon \mathbb{R} \to \mathbb{R}^d$ with $\boldsymbol{l}_{\mathbf{v}}(t) = t\mathbf{v}$, it holds that $\boldsymbol{l}_{\mathbf{v}}((a-\delta, a+\delta)) \subset \Omega$. Now, we prove the statement by contradiction. If the set (7.20) is dense in $\mathcal{E}(\Omega)$, then the set

$$\begin{aligned}&\left\{(\mathbf{W}\mathbf{v})^T \mathrm{ReLU}(\mathbf{W}\mathbf{v} \cdot - \mathbf{b}) \colon \mathbf{W} \in \mathbb{R}^{p \times d}, \mathbf{b} \in \mathbb{R}^p\right\} \\ =&\left\{\sum_{i=1}^{p} w_i \mathrm{ReLU}(w_i \cdot - b_i) \colon w_i, b_i \in \mathbb{R}\right\} \end{aligned} \tag{7.21}$$

is dense in $\mathcal{E}((a-\delta, a+\delta))$. Note that all functions f in (7.20) can be rewritten in the form

$$f(x) = \sum_{i=1}^{p_1} \mathrm{ReLU}(w_i x - b_i) + \sum_{i=1}^{p_2} (-\mathrm{ReLU}(-\tilde{w}_i x - \tilde{b}_i)), \tag{7.22}$$

where $w_i, \tilde{w}_i \in \mathbb{R}^+$, $b_i, \tilde{b}_i \in \mathbb{R}$, and $p_1 + p_2 = p$. Every summand in this decomposition is an increasing function. For the continuous and increasing function

$$g \colon t \mapsto \mathrm{ReLU}(t - a + \delta/2) - \mathrm{ReLU}(t - a - \delta/2), \tag{7.23}$$

the density implies that there exists f of the form (7.22) satisfying $\|g - f\|_{C((a-\delta,a+\delta))} \leq \delta/16$. The fact that $g(a+\delta/2) = g(a+\delta)$ implies that $(f(a+\delta) - f(a+\delta/2)) \leq \delta/8$. In addition, it holds that

$$\begin{aligned} &f(a+\delta) - f(a+\delta/2) \\ \geq &\sum_{i=1}^{p_1} \mathrm{ReLU}\Big(w_i(a+\delta) - b_i\Big) - \mathrm{ReLU}\Big(w_i(a+\delta/2) - b_i\Big) \\ \geq &\sum_{\{i : b_i \leq w_i(a+\delta/2)\}} w_i(a + \delta - a - \delta/2) \\ = &\sum_{\{i : b_i \leq w_i(a+\delta/2)\}} w_i \delta/2. \end{aligned} \tag{7.24}$$

Hence, we conclude that $\sum_{\{i : b_i \leq w_i(a+\delta/2)\}} w_i \leq 1/4$. Similarly, we can show that $\sum_{\{i : \tilde{b}_i \geq \tilde{w}_i(\delta/2-a)\}} \tilde{w}_i \leq 1/4$. Using these two estimates, we get that

$$\begin{aligned} \frac{7}{8}\delta &= g(a+\delta/2) - g(a-\delta/2) - \frac{1}{8}\delta \\ &\leq f(a+\delta/2) - f(a-\delta/2) \\ &\leq \sum_{\{i : b_i \leq w_i(a+\delta/2)\}} \delta w_i + \sum_{\{i : \tilde{b}_i \geq \tilde{w}_i(\delta/2-a)\}} \delta \tilde{w}_i \leq \frac{\delta}{2}, \end{aligned} \tag{7.25}$$

which yields a contradiction. Hence, the set (7.20) cannot be dense in $\mathcal{E}(\Omega)$. □

Remark 7.1. *Any increasing linear spline s with one knot is fully defined by the knot position t_0 and the slope on its two linear regions (s_- and s_+). This can be expressed as $s = \mathbf{u}^T \mathrm{ReLU}(\mathbf{u}(t - t_0))$ with $\mathbf{u} = (\sqrt{s_+}, -\sqrt{s_-})$. Hence, among one-knot spline activation functions, the ReLU already achieves the maximal representation power for CRR-NNs. We infer that increasing PReLU and leaky ReLU induce the same limitations as the ReLU when plugged into CRR-NNs.*

In contrast, with Proposition 7.4, the set $\mathcal{E}(\Omega)$ can be approximated using increasing linear-spline activation functions.

Proposition 7.4. *Let $\Omega \subset \mathbb{R}^d$ be compact and $m \geq 2$. Then, the set*

$$\left\{ \mathbf{W}^T \boldsymbol{\sigma}(\mathbf{W}\cdot) \colon \mathbf{W} \in \mathbb{R}^{p \times d}, \sigma_i \in \mathcal{LS}^m_\uparrow(\mathbb{R}) \right\} \tag{7.26}$$

is dense with respect to $\|\cdot\|_{C(\Omega)}$ in $\mathcal{E}(\Omega)$.

Proof. First, we consider the case $d = 1$. By rescaling and shifting, we can assume that $S \subset [0, 1]$ without loss of generality. Let $f \in C^{0,1}_\uparrow([0, 1])$, and φ_n be the linear-spline interpolator of f at locations $0, 1/2^n, \ldots, (1 - 1/2^n), 1$. Since f is increasing and φ_n is piecewise linear, φ_n is also increasing. Further, we get that

$$\|f - \varphi_n\|_{C([0,1])} \leq \max_{k \in \{1, \ldots, 2^n\}} f(k/2^n) - f((k-1)/2^n). \tag{7.27}$$

Continuous functions on compact sets are uniformly continuous, which directly implies that $\|f - \varphi_n\|_{C([0,1])} \to 0$. Now, we represent φ_n as a linear combination of increasing linear splines with 2 knots

$$\varphi_n(x) = f(0) + \sum_{k=1}^{2^n} a_{k,n} g\Big(2^n \cdot - (k-1)\Big), \tag{7.28}$$

where $a_{k,n} = (f(k/2^n) - f((k-1)/2^n))$ and g is given by

$$g(x) = \begin{cases} 0, & x \leq 0 \\ x, & 0 < x \leq 1 \\ 1, & \text{otherwise.} \end{cases} \tag{7.29}$$

Finally, (7.28) can be recast as $\varphi_n(x) = \mathbf{w}_n^T \boldsymbol{\sigma}_n(x\mathbf{w}_n)$, where each $\sigma_{n,i}$ is an increasing linear spline with 2 knots and $\mathbf{w} \in \mathbb{R}^{2^n}$. This concludes the proof for $d = 1$.

Now, we extend this result to any $d \in \mathbb{N}^+$. Let $\boldsymbol{\Phi} \colon \mathbb{R}^d \to \mathbb{R}^d$ be given by $\mathbf{x} \mapsto \mathbf{W}^T \sigma(\mathbf{W}\mathbf{x})$ with components $\sigma_i \in \mathcal{C}^0_\uparrow(\mathbb{R})$. Let $S_i = \{\mathbf{w}_i^T \mathbf{x} \colon \mathbf{x} \in \Omega\}$, where $\mathbf{w}_i \in \mathbb{R}^d$ is the ith row of $\mathbf{W}$. Using the result for $d = 1$, each σ_i can be approximated in $C(S_i)$ by a sequence of functions $(\mathbf{u}_{n,i}^T \boldsymbol{\varphi}_n(\mathbf{u}_{n,i}\cdot))_{n \in \mathbb{N}}$, where $\boldsymbol{\varphi}_n$ has components $\varphi_{n,i} \in \mathcal{LS}^2_\uparrow(\mathbb{R})$ and $\mathbf{u}_{n,i}$ are

vectors with a size that does not dependend on i. Further, the $\mathbf{u}_{n,i}$ can be chosen such that the jth component is only nonzero for a single i. Let $\mathbf{U}_n$ be the matrix whose columns are $\mathbf{u}_{n,i}$. Then, we directly have that

$$\lim_{n\to\infty} \max_{\mathbf{x}\in\{\mathbf{y}\in\mathbb{R}^d : y_i\in S_i\}} \left\| \mathbf{U}_n^T \boldsymbol{\varphi}_n(\mathbf{U}_n\mathbf{x}) - \boldsymbol{\sigma}(\mathbf{x}) \right\|_2 = 0. \tag{7.30}$$

Hence, the sequence of functions $((\mathbf{U}_n\mathbf{W})^T\boldsymbol{\varphi}_n(\mathbf{U}_n\mathbf{W}\cdot))_{n\in\mathbb{N}}$ converges to $\boldsymbol{\Phi}$ in $C(\Omega)$. This concludes the proof. □

In the end, Propositions 7.3 and 7.4 imply that using linear-spline activation functions instead of the ReLU for the σ_i enables us to approximate more convex regularizers $R_{\boldsymbol{\theta}}$.

Corollary 7.1. *Let $\Omega \subset \mathbb{R}^d$ be convex and compact with a nonempty interior. Then, the regularizers of the form* (7.4) *with Jacobians of the form* (7.26) *are dense in*

$$\left\{ \sum_{i=1}^{p} \psi_i(\mathbf{w}_i^T\mathbf{x}) : \psi_i \in C^{1,1}(\mathbb{R}) \text{ convex}, \mathbf{w}_i \in \mathbb{R}^d \right\} \tag{7.31}$$

with respect to $\|\cdot\|_{C^1(\Omega)}$. The density does not hold if we only consider regularizers with Jacobians of the form (7.20).

Proof. Let R be in (7.31). Consequently, its Jacobian is in $\mathcal{E}(\Omega)$. Due to Proposition 7.3, the regularizers with Jacobians of the form (7.20) cannot be dense with respect to $\|\cdot\|_{C^1(\Omega)}$. Meanwhile, by Proposition 7.4, we can choose $\mathbf{x}_0 \in \Omega$ and corresponding regularizers R_n of the form (7.4) with $\boldsymbol{J}_{R_n} \in$ (7.26), $\|\boldsymbol{J}_{R_n} - \boldsymbol{J}_R\|_{C(\Omega)} \to 0$ as $n \to \infty$, and $R_n(\mathbf{x}_0) = R(\mathbf{x}_0)$. Now, the mean-value theorem readily implies that $\|R_n - R\|_{C^1(\Omega)} \to 0$ as $n \to \infty$. □

Motivated by these results, we propose to parameterize the σ_i with learnable linear-spline activation functions. This results in profiles ψ_i that are splines of degree 2, being piecewise polynomials of degree 2 with continuous derivatives.

7.4 Implementation

7.4.1 Training a Multi-Gradient-Step Denoiser

Let $\{\mathbf{x}^m\}_{m=1}^M$ be a set of clean images and let $\{\mathbf{y}^m\}_{m=1}^M = \{\mathbf{x}^m + \mathbf{n}^m\}_{m=1}^M$ be their noisy versions, where $\mathbf{n}^m$ is the noise realisation. Given a loss function $\mathcal{L}$, the natural procedure to learn the parameters of $R_{\boldsymbol{\theta}}$ based on (7.5) is to solve

$$\boldsymbol{\theta}_t^*, \lambda_t^* \in \arg\min_{\boldsymbol{\theta},\lambda} \sum_{m=1}^{M} \mathcal{L}\Big(\boldsymbol{T}^t_{R_{\boldsymbol{\theta}},\lambda,\alpha}(\mathbf{y}^m), \mathbf{x}^m \Big) \tag{7.32}$$

for the limiting case $t = \infty$ and an admissible stepsize α. Here, $\boldsymbol{T}^t_{R_{\boldsymbol{\theta}},\lambda,\alpha}$ denotes the t-fold composition of the gradient-step NN given in (7.9). In principle, one can optimize the training problem (7.32) with $t = \infty$. This forms a bilevel optimization problem that can be handled with implicit differentiation techniques [78, 293, 308, 309]. However, it turns out that it is unnecessary to fully compute the fixed-point $\boldsymbol{T}^\infty_{R_{\boldsymbol{\theta}},\lambda,\alpha}(\mathbf{y}^m)$ to learn $R_{\boldsymbol{\theta}}$ in our constrained setting. Instead, we approximate $\boldsymbol{T}^\infty_{R_{\boldsymbol{\theta}},\lambda,\alpha}(\mathbf{y}^m)$ in a finite number of steps. This specifies the *t-step denoiser* NN $\boldsymbol{T}^t_{R_{\boldsymbol{\theta}},\lambda,\alpha}$, which is trained such that

$$\boldsymbol{T}^t_{R_{\boldsymbol{\theta}},\lambda,\alpha}(\mathbf{y}^m) \simeq \mathbf{x}^m \tag{7.33}$$

for $m = 1, \ldots, M$. This corresponds to a partial minimization of (7.5) with initial guess $\mathbf{y}^m$ or, equivalently, as the unfolding of the gradient-descent algorithm for t iterations with shared parameters across iterations [104, 310]. For small t, this yields a fast-to-evaluate denoiser. Since it is not necessarily a proximal operator, its interpretability is, however, limited.

Once the gradient-step NN is trained, we can plug the corresponding $R_{\boldsymbol{\theta}}$ into (7.5), and fully solve the optimization problem. This yields an interpretable *proximal denoiser*. In practice, turning a t-step denoiser into a proximal one requires the adjustment of λ and the addition of a scaling parameter, as described in Section 7.4.4. Our numerical experiments in Section 7.6.1 indicate that the number of steps t used for training the multi-gradient-step denoiser has little influence on the test performances of both the t-step and proximal denoisers. Hence, training the model within a few minutes is possible. Note that our method bears some resemblance with the variational networks (VN) proposed in [300], but there are some fundamental differences. While the model used in [300] also involves a sum of convex ridges with learnable profiles, these are parameterized by radial-basis functions and only the last step of the gradient descent is included in the forward pass. The authors of [300] observed that an increase in t deters the denoising performances, which is not the case for our architecture. More differences are outlined in Section 7.4.2.

7.4.2 Implementation of the Constraints

Our learning of the t-step denoiser is constrained as follows.

(i) The activation functions σ_i must be increasing (convexity constraint on ψ_i).

(ii) The activation functions σ_i must take the value 0 somewhere (existence constraint).

(iii) The stepsize in (7.9) should satisfy $\alpha \in (0, 2/(1 + \lambda L_{\boldsymbol{\theta}}))$ (convergent gradient-descent).

Since the methods to enforce these constraints can have a major impact on the final

performance, they must be designed carefully.

Monotonic Splines Here, we address Constraints (i) and (ii) simultaneously. Similar to [189, 236], we use learnable linear splines $\sigma_{\mathbf{c}^i}\colon \mathbb{R} \to \mathbb{R}$ with $(M+1)$ uniform knots $\nu_m = (m - M/2)\Delta$, $m = 0, \dots, M$, where Δ is the spacing of the knots. For simplicity, we assume that M is even. The learnable parameter $\mathbf{c}^i = (c_m^i)_{m=0}^M \in \mathbb{R}^{M+1}$ defines the value $\sigma_{\mathbf{c}^i}(\nu_m) = c_m^i$ of $\sigma_{\mathbf{c}^i}$ at the knots. To fully characterize $\sigma_{\mathbf{c}^i}$, we extend it by the constant value $\mathbf{c}_0^i$ on $(-\infty, \nu_0]$ and $\mathbf{c}_M^i$ on $[\nu_M, +\infty)$. This choice results in a linear extension for the corresponding indefinite integrals that appear for the regularizer $R_{\boldsymbol{\theta}}$ in (7.5). Further details on the implementation of learnable linear splines can be found in [236].

Let $\mathbf{D} \in \mathbb{R}^{M \times (M+1)}$ be the one-dimensional finite-difference matrix with $(\mathbf{D}\mathbf{c}^i)_m = c_{m+1}^i - c_m^i$ for $m = 0, \dots, (M-1)$. As $\sigma_{\mathbf{c}^i}$ is piecewise-linear, it holds that

$$\sigma_{\mathbf{c}^i} \text{ is increasing} \Leftrightarrow \mathbf{D}\mathbf{c}^i \geq 0. \tag{7.34}$$

In order to optimize over $\{\sigma_{\mathbf{c}}\colon \mathbf{D}\mathbf{c} \geq 0\}$, we reparameterize the linear splines as $\sigma_{\boldsymbol{P}_\uparrow(\mathbf{c}^i)}$, where

$$\boldsymbol{P}_\uparrow = \mathbf{C}\mathbf{D}^\dagger \mathrm{ReLU}(\mathbf{D}\,\cdot) \tag{7.35}$$

is a nonlinear projection operator onto the feasible set. There, $\mathbf{D}^\dagger$ denotes the Moore-Penrose inverse of $\mathbf{D}$ and $\mathbf{C} = (\mathbf{Id}_{M+1} - \mathbf{1}_{M+1}\mathbf{e}_{M/2+1}^T)$ shifts the output such that the $(M/2+1)$th element is zero. In effect, this projection simply preserves the nonnegative finite differences between entries in $\mathbf{c}^i$ and sets the negative ones to zero. As the associated profiles ψ_i are convex and satisfy $\psi_i'(0) = \sigma_i(0) = 0$, Proposition 7.1 guarantees the existence of a solution for Problem (7.2).

The proposed parameterization $\sigma_{\boldsymbol{P}_\uparrow(\mathbf{c}^i)}$ of the splines has the advantage of using unconstrained trainable parameters $\mathbf{c}_i$. The gradient of the objective in (7.32) with respect to $\mathbf{c}_i$ directly takes into account the constraint via $\boldsymbol{P}_\uparrow$. This approach differs significantly from the more standard projected gradient descent—as done in [300] to learn convex profiles—where the $\boldsymbol{c}_i$ would be projected onto $\{\mathbf{c}_i\colon \mathbf{D}\mathbf{c}_i \geq 0\}$ after each gradient step. While the latter routine is efficient for convex problems, we found it to perform poorly for the non-convex problem (7.32). For an efficient forward and backward pass with auto-differentiation, $\boldsymbol{P}_\uparrow$ is implemented with the `cumsum` function instead of an explicit construction of the matrix $\mathbf{D}^\dagger$, and the computational overhead is very small.

Sparsity-Promoting Regularization The use of learnable activation functions can lead to overfitting and can weaken the generalizability to arbitrary operators $\mathbf{H}$. Hence, the training procedure ought to promote simple linear splines. Here, it is natural to promote the better-performing splines with the fewest knots. This is achieved by penalizing the second-order total variation $\|\mathbf{L}\boldsymbol{P}_\uparrow(\mathbf{c}_i)\|_1$ of each spline $\sigma_{\boldsymbol{P}_\uparrow(\mathbf{c}_i)}$, where $\mathbf{L} \in \mathbb{R}^{(M-1)\times(M+1)}$

is the second-order finite-difference matrix. The final training loss then reads

$$\sum_{m=1}^{M} \mathcal{L}\Big(\boldsymbol{T}^t_{R_{\boldsymbol{\theta}},\lambda,\alpha}(\mathbf{y}^m), \mathbf{x}^m\Big) + \eta \sum_{i=1}^{p} \|\mathbf{L}\boldsymbol{P}_{\uparrow}(\mathbf{c}_i)\|_1, \tag{7.36}$$

where $\eta \in \mathbb{R}^+$ allows one to tune the strength of the regularization. We refer to [216] for more theoretical insights into second-order total-variation regularization and to [236] for experimental evidence of its relevance for machine learning.

Convergent Gradient Steps Constraint (iii) guarantees that the t-fold composition of the gradient-step NN $\boldsymbol{T}^t_{R_{\boldsymbol{\theta}},\lambda,\alpha}$ computes the actual minimizer of (7.5) for $t \to \infty$. Therefore, it should be enforced in any sensible training method. In addition, it brings stability to the training. To fully exploit the model capacity, even for small t, we need a precise upper bound for $\mathrm{Lip}(\boldsymbol{\nabla} R_{\boldsymbol{\theta}})$. The estimate that we provide in Proposition 7.5 is sharper than the classical bound derived from the sub-multiplicativity of the Lipschitz constant for compositional models. It is easily computable as well.

Proposition 7.5. *Let L_θ denote the Lipschitz constant of $\boldsymbol{\nabla} R_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{W}^T \boldsymbol{\sigma}(\mathbf{W}\mathbf{x})$ with $\mathbf{W} \in \mathbb{R}^{p\times d}$ and $\sigma_i \in \mathcal{C}^{0,1}_{\uparrow}(\mathbb{R})$. With the notation $\boldsymbol{\Sigma}_\infty = \mathbf{diag}(\|\sigma_1'\|_\infty, \ldots, \|\sigma_p'\|_\infty)$ it holds that*

$$L_\theta \leq \|\mathbf{W}^T \boldsymbol{\Sigma}_\infty \mathbf{W}\| = \|\sqrt{\boldsymbol{\Sigma}_\infty}\mathbf{W}\|^2, \tag{7.37}$$

which is tighter than the naive bound

$$L_\theta \leq L_{\boldsymbol{\sigma}} \|\mathbf{W}\|^2. \tag{7.38}$$

Proof. The bound (7.38) is a standard result for compositional models. Next, we note that the Hessian of $R_{\boldsymbol{\theta}}$ reads

$$\mathbf{H}_{R_{\boldsymbol{\theta}}}(\mathbf{x}) = \mathbf{W}^T \boldsymbol{\Sigma}(\mathbf{W}\mathbf{x})\mathbf{W}, \tag{7.39}$$

where $\boldsymbol{\Sigma}(\mathbf{z}) = \mathbf{diag}(\sigma_1'(z_1), \ldots, \sigma_p'(z_p))$. Further, it holds that $L_\theta \leq \sup_{\mathbf{x}\in\mathbb{R}^d} \|\mathbf{H}_{R_{\boldsymbol{\theta}}}(\mathbf{x})\|$. Since the functions σ_i are increasing, we have for every $\mathbf{x} \in \mathbb{R}^p$ that $\boldsymbol{\Sigma}_\infty - \boldsymbol{\Sigma}(\mathbf{W}\mathbf{x}) \succeq 0$ and, consequently,

$$\mathbf{W}^T \Big(\boldsymbol{\Sigma}_\infty - \boldsymbol{\Sigma}(\mathbf{W}\mathbf{x})\Big)\mathbf{W} \succeq 0. \tag{7.40}$$

Using the Courant-Fischer theorem, we now infer that the largest eigenvalue of $\mathbf{W}^T\boldsymbol{\Sigma}_\infty\mathbf{W}$ is greater than that of $\mathbf{W}^T\boldsymbol{\Sigma}(\mathbf{W}\mathbf{x})\mathbf{W}$. □

The bounds (7.37) and (7.38)are in agreement when the activation functions are identical, which is typically not the case in our framework. For the 14 NNs trained in Section 7.6, we found that the improved bound (7.37) was on average 3.2 times smaller than (7.38). As (7.37) depends on the parameters of the model, it is critical to embed the computation into

the forward pass. Otherwise, the training gets unstable. This is done by first estimating the normalized eigenvector $\mathbf{u}$ corresponding to the largest eigenvalue of $\mathbf{W}^T\mathbf{\Sigma}_\infty\mathbf{W}$ via the power-iteration method in a non-differentiable way, for instance under the `torch.no_grad()` context-manager. Then, we directly plug the estimate $L_\theta \simeq \|\mathbf{W}^T\mathbf{\Sigma}_\infty\mathbf{W}\mathbf{u}\|$ in our model and hence embed it in the forward pass. This approach is inspired by the spectral-normalization technique proposed in [80], which is a popular and efficient way to enforce Lipschitz constraints on fully connected linear layers. Note that a similar simplification is also proposed and studied in the context of deep equilibrium models [311]. In practice, the estimate $\mathbf{u}$ is stored so that it can be used as a warm start for the next computation of L_θ.

7.4.3 From Gradients to Potentials

To recover the regularizer R from its gradient $\boldsymbol{\nabla} R$, one has to determine the profiles ψ_i, which satisfy $\psi_i' = \sigma_{\boldsymbol{P}_\uparrow(\mathbf{c}^i)}$. Hence, each ψ_i is a piecewise polynomial of degree 2 with continuous derivatives, i.e. a spline of degree two. These can be expressed as a weighted sum of shifts of the rescaled causal B-spline of degree 2 [126], more precisely as

$$\psi_i = \sum_{k\in\mathbb{Z}} d_k^i \beta_+^2 \left(\frac{\cdot - k}{\Delta}\right). \tag{7.41}$$

To determine the coefficients $(d_k^i)_{k\in\mathbb{Z}}$, we use the fact that $(\beta_+^2)'(k) = (\delta_{1,k} - \delta_{2,k})$, where δ is the Kronecker delta, see [126] for details. Hence, we obtain that $d_k^i - d_{k-1}^i = (\boldsymbol{P}_\uparrow(\mathbf{c}^i))_k$, which defines $(d_k^i)_{k\in\mathbb{Z}}$ up to a constant. This constant can be set arbitrarily as it does not affect $\boldsymbol{\nabla} R$. Due to the finite support of β_+^2, one can efficiently evaluate ψ_i and then R.

7.4.4 Boosting the Universality of the Regularizer

The learned $R_{\boldsymbol{\theta}}$ depends on the training task (denoising) and on the noise level. To solve a generic inverse problem, in addition to the regularization strength λ, we propose to incorporate a tunable scaling parameter $\mu \in \mathbb{R}^+$ and to compute

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\arg\min}\ \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda/\mu R_{\boldsymbol{\theta}}(\mu\mathbf{x}). \tag{7.42}$$

While the scaling parameter is irrelevant for homogeneous regularizers such as the Tikhonov and TV, it is known to boost the performance within the PnP framework when applied to the input of the denoiser [270]. During the training of t-step denoisers, we also learn a scaling parameter μ by letting the gradient step NN (7.7) become

$$\boldsymbol{T}_{R_{\boldsymbol{\theta}},\lambda,\mu,\alpha}(\mathbf{x}) = \mathbf{x} - \alpha\Big((\mathbf{x} - \mathbf{y}) + \lambda\boldsymbol{\nabla} R_{\boldsymbol{\theta}}(\mu\mathbf{x})\Big), \tag{7.43}$$

with now $\alpha < 2/(1 + \lambda\mu\mathrm{Lip}(\boldsymbol{\nabla} R_{\boldsymbol{\theta}}))$.

Algorithm 1 FISTA [28] to solve (7.42)

Input: $\mathbf{x}_0 \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^m$, $\lambda \geq 0$, $\mu > 0$
Set $k = 0$, $\mathbf{z}_0 = \mathbf{x}_0$, $\alpha = 1/(\mu\lambda \text{Lip}(\boldsymbol{\nabla} R) + \|\mathbf{H}\|^2)$, $t_0 = 1$
while tolerance not reached **do**
$\mathbf{x}_{k+1} = (\mathbf{z}_k - \alpha(\mathbf{H}^T(\mathbf{H}\mathbf{z}_k - \mathbf{y}) + \lambda \boldsymbol{\nabla} R(\mu \mathbf{z}_k)))_+$
$t_{k+1} = (1 + \sqrt{4t_k^2 + 1})/2$
$\mathbf{z}_{k+1} = \mathbf{x}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}_{k+1} - \mathbf{x}_k)$
$k \leftarrow k + 1$
Output: $\mathbf{x}_k$

Table 7.1: Properties of different regularization frameworks.

	Explicit cost	Provably convergent	Universal	Shallow	Smooth reg.
TV	✓	✓	✓	✓	✗
ACR	✓	✓	✗	✗	✗
DnICNN	✓	✓	✓	✗	✓
PnP-βCNN	✗	✓	✓	✗	-
PnP-DnCNN	✗	✗	✓	✗	-
CRR-NN	✓	✓	✓	✓	✓

7.4.5 Reconstruction Algorithm

The objective in (7.42) is smooth with Lipschitz-continuous gradients. Hence, a reconstruction can be computed through gradient-based methods. We found the fast iterative shrinkage-thresholding algorithm (FISTA, Algorithm 1) to be well-suited to the problem while it also allows us to enforce the positivity of the reconstruction. Other efficient algorithms for CRR-NNs include the adaptive gradient descent (AdaGD) [312] and its proximal extension [313]; both benefit from a stepsize based on an estimate of the local Lipschitz constant of $\boldsymbol{\nabla} R$ instead of a more conservative global one.

7.5 Connections to Deep-Learning Approaches

Our proposed CRR-NNs have a single nonlinear layer, which is rather unusual in the era of deep learning. To further explore their theoretical properties, we briefly discuss two successful deep-learning methods, namely, the PnP and the explicit design of convex regularizers, and state their most stable and interpretable versions. This will clarify the notions of strict convergence, interpretability, and universality. All the established comparisons are synthesized in Table 7.1.

7.5.1 Plug-and-Play and Averaged Denoisers

Convergent Plug-and-Play *For a more general discussion on convergent PnP methods, see the introduction of Chapter 6.* The training procedure proposed for CRR-NNs leads to a convex regularizer $R_{\boldsymbol{\theta}}$, whose proximal operator (7.5) is a good denoiser. Conversely, the proximal operator can be replaced by a powerful denoiser $\boldsymbol{D}$ in proximal algorithms, which is referred to as PnP. In the PnP-FBS algorithm derived from (7.2) [28, 265], the reconstruction is carried out iteratively via

$$\mathbf{x}_{k+1} = \boldsymbol{D}\Big(\mathbf{x}_k - \alpha \mathbf{H}^T(\mathbf{H}\mathbf{x}_k - \mathbf{y})\Big), \tag{7.44}$$

where α is the stepsize and $\boldsymbol{D}\colon \mathbb{R}^d \to \mathbb{R}^d$ is a generic denoiser. A standard set of sufficient conditions[†] to guarantee convergence of the iterations (7.44) is that

(i) $\boldsymbol{D}$ is averaged, namely $\boldsymbol{D} = \beta \boldsymbol{N} + (1-\beta)\mathbf{Id}$ where $\beta \in (0,1)$ and $\boldsymbol{N}\colon \mathbb{R}^n \to \mathbb{R}^n$ is a nonexpansive mapping;

(ii) $\alpha \in [0, 2/\|\mathbf{H}\|^2)$;

(iii) the update operator in (7.44) has a fixed point.

In general, Condition (i) is not sufficient to ensure that $\boldsymbol{D}$ is the proximal operator of some convex regularizer R. Hence, its interpretability is still limited. Further, Condition (ii) implies that $\mathbf{x} \mapsto \big(\mathbf{x} - \alpha \mathbf{H}^T(\mathbf{H}\mathbf{x} - \mathbf{y})\big)$ is averaged. Hence, as averagedness is preserved through composition, the iterates are updated by the application of an averaged operator (see [83] for details). With Condition (iii), the convergence of the iterations (7.44) follows from Opial's convergence theorem. Beyond convergence, it is known that averaged denoisers with $\beta \leq 1/2$ yield a stable reconstruction map in the measurement domain [314], in the same sense as given in Proposition 7.2 for CRR-NNs.

The nonexpansiveness of $\boldsymbol{D}$ is also commonly assumed to prove the convergence of other PnP schemes. This includes, for instance, gradient-based PnP [78]. There, the gradient $\boldsymbol{\nabla} R$ of the regularizer used in reconstruction algorithms is replaced with a learned monotone operator $\boldsymbol{F} = \mathbf{I} - \boldsymbol{D}$. The operator $\boldsymbol{D}$ can be interpreted as a denoiser and is assumed to be nonexpansive to prove convergence.

Constraint vs Performance As discussed in [76, 93], the performance of the denoiser $\boldsymbol{D}$ is in direct competition with its averagedness. A simple illustration of this issue is provided in Figure 7.1. Unsurprisingly, Condition (i) is not met by any learned state-of-the-art denoiser, and it is usually also relaxed in the PnP literature.

[†]Here, $\mathbf{H}$ can be noninvertible; otherwise, weaker conditions exist [81].

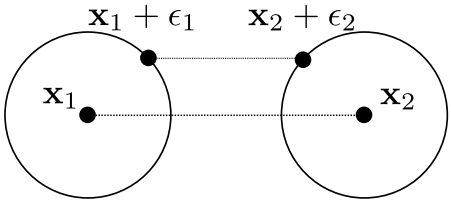

Figure 7.1: The distance between the two noisy images $(\mathbf{x}_1 + \epsilon_1)$ and $(\mathbf{x}_2 + \epsilon_2)$ can be smaller than that between their clean versions $\mathbf{x}_1$ and $\mathbf{x}_2$. This limits the performance of a nonexpansive denoiser $\boldsymbol{D}$ since $\|\boldsymbol{D}(\mathbf{x}_1 + \epsilon_1) - \boldsymbol{D}(\mathbf{x}_2 + \epsilon_2)\| \leq \|\mathbf{x}_1 + \epsilon_1 - (\mathbf{x}_2 + \epsilon_2)\| < \|\mathbf{x}_1 - \mathbf{x}_2\|$ in the scenario depicted.

For instance, it is common to use non-1-Lipschitz learning modules, such as batch normalization [81], or to only constrain the residual $(\mathbf{Id} - \boldsymbol{D})$ to be nonexpansive, which enables one to train a nonexpansive NN in a residual way [81, 83, 101], with the caveat that $\mathrm{Lip}(\boldsymbol{D})$ can be as large as 2. Another recent approach consists of penalizing during training either the norm of the Jacobian of $\boldsymbol{D}$ at a finite set of locations [75, 76] or of another local estimate of the Lipschitz constant [77, 78]. Interestingly, even slightly relaxed frameworks usually yield significant improvements in the reconstruction quality. However, they do not provide convergence guarantees for ill-posed inverse problems, which is problematic for sensitive applications such as biomedical imaging.

Averaged Deep NNs To leverage the success of deep learning, $\boldsymbol{N}$ is typically chosen as a deep CNN of the form[‡]

$$\boldsymbol{N} = \boldsymbol{C}_K \circ \boldsymbol{\sigma} \circ \cdots \circ \boldsymbol{C}_2 \circ \boldsymbol{\sigma} \circ \boldsymbol{C}_1, \tag{7.45}$$

where $\boldsymbol{C}_k$ are learnable convolutional layers and $\boldsymbol{\sigma}$ is the activation function [80, 81, 189]. To meet Condition (i), $\boldsymbol{N}$ must be nonexpansive, which one usually achieves by constraining $\boldsymbol{C}_k$ and $\boldsymbol{\sigma}$ to be nonexpansive. This is predicated on the sub-multiplicativity of the Lipschitz constant with respect to composition; as in $\mathrm{Lip}(\boldsymbol{f} \circ \boldsymbol{g}) \leq \mathrm{Lip}(\boldsymbol{f})\mathrm{Lip}(\boldsymbol{g})$. Unfortunately, this bound is not sharp and may grossly overestimate $\mathrm{Lip}(\boldsymbol{f} \circ \boldsymbol{g})$. For deep models, this overestimation aggravates since the bound is used sequentially. Therefore, for averaged NNs, the benefit of depth is unclear because the gain of expressivity brought by the many layers is reduced by a potentially very pessimistic Lipschitz-constant estimate. Put differently, these CNNs can easily learn the zero function while they struggle to generate mappings with a Lipschitz constant close to one. For the same reason, the learning process is also prone to vanishing gradients in this constrained setting. Under Lipschitz constraints, the zero-gradient region of the popular ReLU activation function causes provable limitations [82, 211, 252]. Some of these can be resolved by the use of other activation functions instead, including PReLU, GroupSort (GS) and learnable linear splines (LLS), see Part II for a detailed discussion.

[‡]The benefit of standard skip connections combined with the preservation of the nonexpansiveness of the NN is unclear.

In this chapter, CRR-NNs are compared against two variants of PnP.

- **PnP-DnCNN** corresponds to the popular implementation given in [81]. The denoiser is a DnCNN with 1-Lipschitz linear layers (the constraints are therefore enforced on the residual map only) and unconstrained batch-normalization modules. Hence this method has no convergence and stability guarantees, especially for ill-posed inverse problems.
- **PnP-βCNN** corresponds to PnP equipped with a provably averaged denoiser. This method comes with similar guarantees as CRR-NNs but less interpretability. It is included to convey the message that the standard way of enforcing Lipschitz constraints affects expressivity as reported for instance in [315], and even makes it hard to improve upon TV. With that in mind, CRR-NNs provide a way to overcome this limitation. We shall include averaged denoisers equipped with various activation functions, including ReLU, PReLU, GroupSort (GS), and learnable linear splines (LLS) as introduced and discussed in Chapters 5 and 6.

Construction of Averaged Denoisers from CRR-NNs The training of CRR-NNs offers two ways to build averaged denoisers. Since proximal operators are half-averaged, we directly get that the proximal denoiser (7.5) is an averaged operator. For the t-step denoiser, the following holds.

Proposition 7.6. *The t-step denoiser* (7.33) *is averaged for $\alpha \in [0, 2/(2 + \lambda L_{\boldsymbol{\theta}})]$ with $L_{\boldsymbol{\theta}} = \mathrm{Lip}(\boldsymbol{\nabla} R_{\boldsymbol{\theta}})$.*

Proof. The t-step denoiser is built from the gradient-step operator $\boldsymbol{T}_{R_{\boldsymbol{\theta}},\lambda,\alpha}$. Here, we use the more explicit notation

$$\boldsymbol{T}(\mathbf{x}, \mathbf{y}) = \mathbf{x} - \alpha((\mathbf{x} - \mathbf{y}) + \lambda \boldsymbol{\nabla} R_\theta(\mathbf{x})). \tag{7.46}$$

This makes explicit the dependence on $\mathbf{y}$ and, for simplicity, the dependence on $R_{\boldsymbol{\theta}}$, λ, and α are omitted. It is known that $\boldsymbol{T}$ is averaged with respect to $\mathbf{x}$ for $\alpha \in (0, 2/(1 + \lambda L_{\boldsymbol{\theta}}))$. This ensures convergence of gradient descent, but it does not characterize the denoiser itself. The t-step denoiser depends on the initial value $\mathbf{x}_0 = \mathbf{y}$ and is determined by the recurrence relation $\mathbf{x}_{k+1} = \boldsymbol{T}(\mathbf{x}_k, \mathbf{y})$. For the map $\boldsymbol{L}_k \colon \mathbf{y} \mapsto \mathbf{x}_k$, it holds that $\boldsymbol{L}_{k+1} = \boldsymbol{U} \circ \boldsymbol{L}_k + \alpha \mathbf{Id}$, where $\boldsymbol{U} = \mathbf{Id} - \alpha(\mathbf{Id} + \lambda \boldsymbol{\nabla} R_\theta)$. The Jacobian of $\boldsymbol{U}$ reads $\mathbf{J}_{\boldsymbol{U}} = \mathbf{I} - \alpha(\mathbf{I} + \lambda \mathbf{H}_{R_\theta})$ and satisfies that $((1 - \alpha) - \alpha\lambda L_{\boldsymbol{\theta}})\mathbf{I} \preceq \mathbf{J}_{\boldsymbol{U}} \preceq (1 - \alpha)\mathbf{I}$. From this, we infer that

$$\mathrm{Lip}(\mathbf{U}) \leq \max\Big(\alpha\lambda L_{\boldsymbol{\theta}} - (1 - \alpha), 1 - \alpha\Big). \tag{7.47}$$

Since $\alpha \leq 2/(2 + \lambda L_{\boldsymbol{\theta}})$, we then get that $\mathrm{Lip}(\mathbf{U}) \leq (1 - \alpha)$. Hence, $\mathrm{Lip}(\mathbf{U} \circ \boldsymbol{L}_k) \leq (1 - \alpha)\mathrm{Lip}(\boldsymbol{L}_k)$. Since $\boldsymbol{L}_0 = \mathbf{Id}$ is averaged, the same holds by induction for all the t-step denoisers $\boldsymbol{L}_t$. □

Note that for $\alpha \in (2/(2+\lambda L_{\boldsymbol{\theta}}), 2/(1+\lambda L_{\boldsymbol{\theta}}))$, the 1-step denoiser is also averaged but, for $1 < t < +\infty$, it remains an open question. The structure of t-step and proximal denoisers differs radically from averaged CNNs as in (7.45). For instance, the t-step denoiser uses the noisy input $\mathbf{y}$ in each layer. Remarkably, these skip connections preserve the averagedness of the mapping. While constrained deep CNNs struggle to learn mappings that are not too contractive, both proximal and t-step denoisers can easily reproduce the identity by choosing $R_{\boldsymbol{\theta}} = 0$. This seems key to account for the fact that the proposed denoisers outperform averaged deep NNs, while they can be trained two orders of magnitude faster, see Section 7.6.

7.5.2 Deep Convex Regularizers

Another approach to leverage deep-learning-based priors with stability and convergence guarantees consists of learning a deep convex regularizer R. These priors are typically parameterized with an ICNN, which is an NN with increasing and convex activation functions along with positive weights for some linear layers [303]. There exist various strategies to train the ICNN.

The adversarial convex regularizer (ACR) framework [286, 302] relies on the adversarial training proposed in [295]. The regularizer is learned by minimizing its value on clean images and maximizing its value on unregularized reconstructions. This allows for learning non-smooth R and also avoids bilevel optimization. A key difference with CRR-NNs and PnP methods is that ACR is modality-depend (it is not universal). In addition, with R being non-smooth, it is challenging to exactly minimize the cost function, but the authors of [286, 302] did not find any practical issues in that matter using gradient-based solvers. To boost the performance of R, they also added a sparsifying filter bank to the ICNN, namely, a convex term of the form $\|\mathbf{U}\mathbf{x}\|_1$, where the linear operator $\mathbf{U}$ is made of convolutions learned conjointly with the ICNN.

In [273], the regularizer is trained so that its gradient step is a good blind Gaussian denoiser. There, the authors use ELU activations in the ICNN[§] to obtain a smooth R.

The aforementioned ICNN-based frameworks [273, 286, 302] have major differences with CRR-NNs: (i) they typically require orders of magnitude more parameters; (ii) the computation of $\boldsymbol{\nabla} R$, used to solve inverse problems, requires one to back-propagate through the deep CNN which is time-consuming; (iii) the role of each parameter is not interpretable because of the depth of the model (see Section 7.6.4). As we shall see, CRR-NNs are much faster to train and tend to perform better (see Section 7.6).

[§]The authors also explore non-convex regularization but they offer no guarantees on computing the global minimum.

7.6 Experiments

7.6.1 Training of CRR-NNs

The CRR-NNs are now trained on a Gaussian-denoising task with noise levels $\sigma \in \{5/255, 25/255\}$. The same procedure as in [81, 272] is used to form 238,400 patches of size (40×40) from 400 images of the BSD500 dataset [281]. For validation, the same 12 images as in [81, 272] are used. The weights $\mathbf{W}$ in $R_{\boldsymbol{\theta}}$ are parameterized as the composition of two zero-padded convolutions with kernels of size (7×7) and with 8 and 32 output channels, respectively. This composition of two linear components, although not more expressive theoretically, facilitates the patch-based training of CRR-NNs. For inference, the convolutional layer can then be transformed back to a single convolution. Similar to [293], the kernels of the convolutions are constrained to have zero mean. Lastly, the linear splines have $M+1 = 21$ equally distant knots with $\Delta = 0.01$, and the sparsifying regularization parameter is $\eta = 2 \cdot 10^{-3}(255\sigma)$. We initially set $\mathbf{c}_i = \mathbf{0}$.

The CRR-NNs are trained for 10 epochs with $t \in \{1, 2, 5, 10, 20, 30, 50\}$ gradient steps. For this purpose, the ℓ_1 loss is used for $\mathcal{L}$ along with the Adam optimizer with its default parameters $(\beta_1, \beta_2) = (0.9, 0.999)$, and the batch size is set to 128. The learning rates are decayed with rate 0.75 at each epoch and initially set to 0.05 for the parameters λ and μ, to 10^{-3} for $\mathbf{W}$, and to $5 \cdot 10^{-5}$ for $\mathbf{c}_i$.

Recall that for a given t, the training yields two denoisers.

- ***t*-Step Denoiser**: This corresponds to $\boldsymbol{T}^t_{R_{\boldsymbol{\theta}},\lambda,\alpha}$ and is the denoiser optimized during training. It is natural to compare it to properly constrained PnP methods based on averaged deep denoisers as in [189, 301], which in general also do not correspond to minimizing an energy.
- **Proximal Denoiser**: The learned regularizer $R_{\boldsymbol{\theta}}$ is plugged into (7.42) with $\mathbf{H} = \mathbf{I}$, and the solution is computed using Algorithm 1 with small tolerance (10^{-6} for the relative change of norm between consecutive iterates). The parameters λ and μ are tuned on the validation dataset with the coarse-to-fine method given in Appendix 7.8.1. This important step enables us to compensate for the gap between (i) gradient-step training and full minimization, and (ii) training and testing noise levels, if different.

7.6.2 Denoising: Comparison with Other Methods

Although not the final goal, image denoising yields valuable insights into the training of CRR-NNs. It also enables us to compare CRR-NNs to the related methods given in Table 7.2 on the standard BSD68 test set.

Now, we briefly give the implementation details of the various frameworks. CRR-

Table 7.2: Convex models and averaged denoisers tested on BSD68.

	$\sigma = 5/255$	$\sigma = 25/255$
TV*,‡ [316]	36.41	27.48
Higher-order MRFs*,‡ [293]	NA	28.04
VN^{1,t†}[300]	NA	27.69
LLS-NN$_\sigma{}^\ddagger$ (see Chapter 6)	36.86	27.93
D$_{\text{ISTA}}{}^\ddagger$ [301]	36.54	NA
GS-DnICNN†[273]	36.85	27.76
D$_{\text{ADMM}}{}^\ddagger$[301]	36.62	NA
CRR-NN-ReLU (t-step)†,‡	35.50	26.75
CRR-NN (t-step)†,‡	**36.97**	**28.12**
CRR-NN (proximal)*,‡	<u>36.96</u>	<u>28.11</u>

* Full minimization of a convex function

† Partial minimization of a convex function

‡ Stable steps (Lipschitz-constrained)

NN-ReLU models are trained in the same way as CRR-NNs but with ReLU activation functions (with learnable biases) instead of linear splines. To emulate [273], we train a DnICNN with the same architecture (ELU activations, 6 layers, and 128 channels per layer, 745 344 parameters) as a gradient step denoiser for 200 epochs, separately for $\sigma \in \{5/255, 25/255\}$, and refer to it as GS-DnICNN. We also compare the performance with the nonexpansive denoiser LLS-NN$_\sigma$ introduced in Chapter 6 and trained on the same denoising task as the CRR-NNs with $\sigma \in \{5/255, 25/255\}$. The other reported frameworks do not provide public implementations. Therefore, the numbers are taken from the corresponding papers. Lastly, the TV denoising is performed with the algorithm proposed in [316]. The results for all models are presented in Table 7.2 and Figure 7.2.

- ***t*-Step/Averaged Denoisers**: The CRR-NN-ReLU models perform poorly and confirms that ReLU is not well-suited to our setting. This limitation of ReLU was also observed experimentally in [189] in the context of 1-Lipschitz denoisers. Our models improve over the gradient-step denoisers parameterized with ICNNs, even though the latter has many more parameters. The CRR-NN implementation improves over the special instance VN^{1,t} of variational-network denoisers proposed in [300], which also partially minimizes a convex cost. With a convex model similar to CRR-NNs (see Section 7.4 for a discussion), it is shown that an increase in t decreases the performance (reported as VN$^{1,t}_{24}$ in [300, Figure 5]). The model VN^{1,t} cannot compete with the proximal denoiser trained with bilevel optimization in [293]. By contrast, for $\sigma = 25/255$ we obtain an improvement over VN^{1,t} of 0.2dB for $t = 1$, and more than 0.6dB as t increases. Note that, in [300], the layers of the t-step VN^{1,t} denoiser are not guaranteed to be averaged. Our models also improves over the averaged LLS-NN$_\sigma$ (+0.1dB for $\sigma = 5$, +0.15dB for $\sigma = 25/255$), and the two averaged denoisers D$_{\text{ISTA}}$ and D$_{\text{ADMM}}$ [301] (+0.4/+0.3dB for $\sigma = 5/255$). In

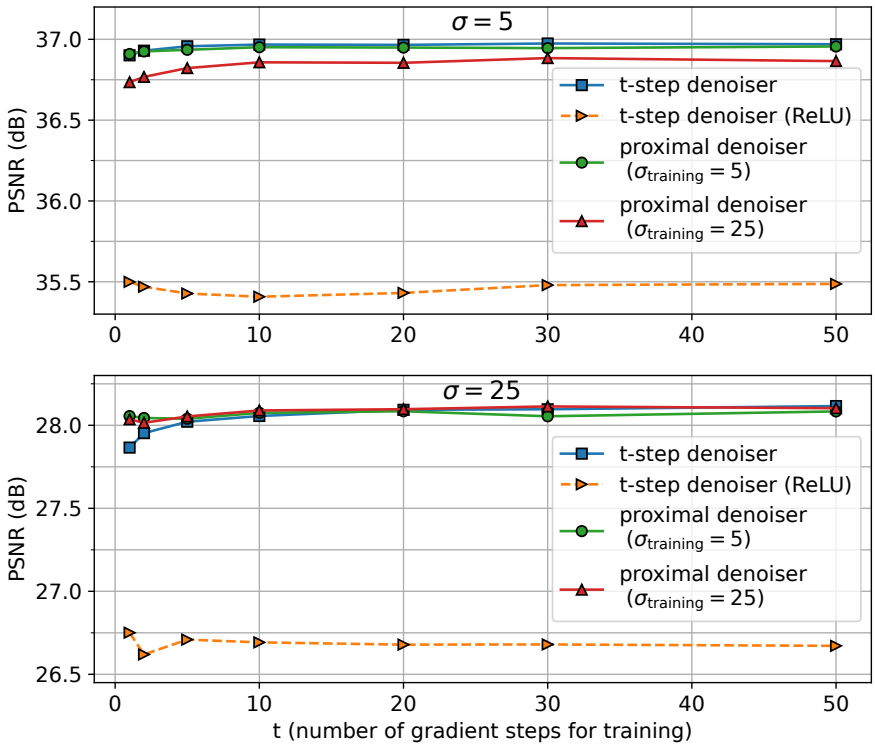

Figure 7.2: Test denoising performance of CRR-NNs for noise level $\sigma = 5/255$ and $\sigma = 25/255$ versus the number of gradient steps used for training, the denoiser type (t-step vs proximal), and the noise level used for training.

their simplest form, the latter are built with fixed linear layers (patch-based wavelet transforms) and learnable soft-thresholding activation functions.

- **Proximal Denoisers**: Our models yield slight improvements over the higher-order Markov random field (MRF) model in the pioneering work [293] (28.04dB vs 28.11dB for $\sigma = 25/255$). With a similar architecture—but with fixed smoothed absolute value ψ_i—the latter approach involves a computationally intensive bilevel optimization with second-order solvers. Here, we show that a few gradient steps for training already suffice to be competitive. This leads to ultrafast training and bridges the gap between higher-order MRF models and VN denoisers. Lastly, we remark that our proximal denoisers are robust to a mismatch in the training and testing noise levels.

7.6.3 Biomedical Image Reconstruction

The six CRR-NNs trained on denoising with $t \in \{1, 10, 50\}$ and $\sigma \in \{5/255, 25/255\}$ are used to solve the MRI and CT inverse problems that we now describe. Note that the setups explored for both modalities are identical to those of Chapter 6.

Table 7.3: Single-coil MRI.

	2-fold				4-fold			
	PSNR		SSIM		PSNR		SSIM	
	PD	PDFS	PD	PDFS	PD	PDFS	PD	PDFS
Zero-fill	33.32	34.49	0.871	0.872	27.40	29.68	0.729	0.745
TV	39.22	37.73	0.947	0.917	32.44	32.67	0.833	0.781
PnP-βCNN (ReLU)	38.15	37.41	0.938	0.918	30.62	31.45	0.818	0.786
PnP-βCNN (PReLU)	38.97	38.09	0.946	0.925	31.22	32.22	0.832	0.800
PnP-βCNN (GS)	38.80	37.92	0.944	0.924	31.27	31.93	0.829	0.796
PnP-βCNN (LLS)	40.06	38.63	0.955	0.931	32.81	33.04	0.859	0.817
CRR-NN	**40.95**	38.91	**0.961**	0.934	33.99	33.75	0.880	0.831
PnP-DnCNN [81]	40.52	**39.02**	0.956	**0.935**	**35.24**	**34.63**	**0.884**	**0.840**

MRI The ground-truth images for our MRI experiments are proton-density weighted knee MR images from the fastMRI dataset [282] with fat suppression (PDFS) and without fat suppression (PD). They are generated from the fully-sampled k-space data. For each of the two categories (PDFS and PD), we create validation and test sets consisting of 10 and 50 images, respectively, where every image is normalized to have a maximum value of one. To gauge the performance of CRR-NNs in various regimes, we experiment with single-coil and multi-coil setups with several acceleration factors. In the single-coil setup, we simulate the measurements by masking the Fourier transform of the ground-truth image. In the multi-coil case, we consider 15 coils, and the measurements are simulated by subsampling the Fourier transforms of the multiplication of the ground-truth images with 15 complex-valued sensitivity maps (these were estimated from the raw k-space data using the ESPIRiT algorithm [283] available in the BART toolbox [284]). For both cases, the subsampling in the Fourier domain is performed with a Cartesian mask that is specified by two parameters: the acceleration $M_{\text{acc}} \in \{2, 4, 8\}$ and the center fraction $M_{\text{cf}} = 0.32/M_{\text{acc}}$. A fraction of M_{cf} columns in the center of the k-space (low frequencies) is kept, while columns in the other region of the k-space are uniformly sampled so that the expected proportion of selected columns is $1/M_{\text{acc}}$. In addition, Gaussian noise with standard deviation $\sigma_{\mathbf{n}} = 2 \cdot 10^{-3}$ is added to the real and imaginary parts of the measurements. The PSNR and SSIM values for each method are computed on the (320×320) centered ROI.

CT To provide a fair comparison with the ACR method, we now target the CT experiment proposed in [286]. The data consists of human abdominal CT scans for 10 patients provided by the Mayo Clinic for the low-dose CT Grand Challenge [285]. The validation set consists of 6 images taken uniformly from the first patient of the training set from [286]. We use the same test set as [286], more precisely, 128 slices with size (512×512) that correspond to one patient. The projections of the data are simulated

Table 7.4: CRR-NN: Single-coil MRI versus training setup.

			2-fold				4-fold			
			PSNR		SSIM		PSNR		SSIM	
image	σ_{train}	t	PD	PDFS	PD	PDFS	PD	PDFS	PD	PDFS
BSD	5/255	1	40.55	38.71	0.959	0.932	33.32	33.37	0.866	0.819
BSD	5/255	10	40.52	38.69	0.959	0.932	33.30	33.36	0.865	0.817
BSD	5/255	50	40.50	38.67	0.958	0.931	33.29	33.32	0.865	0.816
BSD	25/255	1	40.75	38.84	0.960	0.934	33.62	33.60	0.875	0.828
BSD	25/255	10	40.78	38.81	0.960	0.933	33.63	33.59	0.875	0.826
BSD	25/255	50	40.71	38.77	0.960	0.932	33.57	33.54	0.872	0.824
MRI	5/255	10	40.95	38.91	0.961	0.934	33.99	33.75	0.880	0.831
MRI	25/255	10	40.61	38.73	0.959	0.932	33.93	33.71	0.878	0.830

Table 7.5: Multi-coil MRI.

	4-fold				8-fold			
	PSNR		SSIM		PSNR		SSIM	
	PD	PDFS	PD	PDFS	PD	PDFS	PD	PDFS
$\mathbf{H}^T\mathbf{y}$	27.71	29.94	0.751	0.759	23.80	27.19	0.648	0.681
TV	38.06	37.31	0.935	0.914	32.77	33.38	0.850	0.824
PnP-βCNN (ReLU)	37.21	37.06	0.929	0.915	31.37	32.57	0.837	0.822
PnP-βCNN (PReLU)	37.71	37.51	0.934	0.919	31.67	33.11	0.845	0.832
PnP-βCNN (GS)	37.76	37.41	0.933	0.919	31.79	32.9	0.843	0.829
PnP-βCNN (LLS)	38.68	37.96	0.943	0.924	32.75	33.61	0.859	0.835
CRR-NN	<u>39.54</u>	<u>38.29</u>	**0.950**	<u>0.927</u>	<u>34.29</u>	<u>34.50</u>	**0.881**	<u>0.852</u>
PnP-DnCNN [81]	**39.55**	**38.52**	<u>0.947</u>	**0.929**	**35.11**	**35.14**	**0.881**	**0.858**

Table 7.6: CRR-NN: Multi-coil MRI versus training setup.

			4-fold				8-fold			
			PSNR		SSIM		PSNR		SSIM	
image	σ_{train}	t	PD	PDFS	PD	PDFS	PD	PDFS	PD	PDFS
BSD	5/255	1	39.15	38.09	0.947	0.925	33.82	34.22	0.873	0.846
BSD	5/255	10	39.14	38.08	0.946	0.925	33.82	34.20	0.873	0.845
BSD	5/255	50	39.14	38.05	0.946	0.924	33.78	34.16	0.872	0.844
BSD	25/255	1	39.34	38.21	0.948	0.926	34.02	34.35	0.876	0.849
BSD	25/255	10	39.33	38.19	0.948	0.926	34.01	34.34	0.876	0.848
BSD	25/255	50	39.29	38.15	0.948	0.926	33.96	34.29	0.876	0.847
MRI	5/255	10	39.54	38.29	0.950	0.927	34.29	34.50	0.881	0.852
MRI	25/255	10	39.33	38.14	0.947	0.925	34.22	34.40	0.878	0.849

Table 7.7: CT.

	$\sigma_{\mathbf{n}}$=0.5		$\sigma_{\mathbf{n}}$=1		$\sigma_{\mathbf{n}}$=2	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
FBP	32.14	0.697	27.05	0.432	21.29	0.204
TV	36.38	0.936	34.11	0.906	31.57	0.863
PnP-βCNN (ReLU)	36.94	0.914	33.65	0.860	30.34	0.782
PnP-βCNN (PReLU)	37.18	0.927	34.21	0.887	30.87	0.812
PnP-βCNN (GS)	36.95	0.920	33.99	0.877	30.87	0.806
PnP-βCNN (LLS)	38.19	0.931	35.15	0.897	31.85	0.844
ACR [286, 302]	38.06	<u>0.943</u>	35.12	0.911	32.17	0.868
CRR-NN	**39.30**	**0.947**	<u>36.29</u>	<u>0.916</u>	<u>33.16</u>	<u>0.878</u>
PnP-DnCNN [81]	<u>38.93</u>	0.941	**36.49**	**0.921**	**33.52**	**0.897**

Table 7.8: CRR-NN: CT versus training setup.

			$\sigma_{\mathbf{n}}$=0.5		$\sigma_{\mathbf{n}}$=1		$\sigma_{\mathbf{n}}$=2	
image	σ_{train}	t	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
BSD	5/255	1	38.84	0.943	35.70	0.907	32.48	0.860
BSD	5/255	10	38.90	0.943	35.73	0.908	32.49	0.860
BSD	5/255	50	38.82	0.940	35.64	0.904	32.47	0.855
BSD	25/255	1	39.01	0.945	35.91	0.913	32.72	0.867
BSD	25/255	10	39.07	0.945	35.95	0.911	32.71	0.867
BSD	25/255	50	39.04	0.944	35.89	0.912	32.71	0.860
CT	5/255	10	39.30	0.947	36.29	0.916	33.15	0.873
CT	25/255	10	38.89	0.945	36.11	0.917	33.16	0.878

using a parallel-beam acquisition geometry with 200 angles and 400 detectors. Lastly, Gaussian noise with standard deviation $\sigma_{\mathbf{n}} \in \{0.5, 1, 2\}$ is added to the measurements.

Reconstruction Frameworks A reconstruction with isotropic TV regularization is computed with FISTA [28], in which prox_R is computed as in [27] to enforce positivity. We also consider reconstructions obtained with the PnP method with (i) provably averaged denoisers βCNN_σ ($\sigma = 5, 15$)—the one designed and trained in Chapter 6 with various activation functions, including ReLU, PReLU, GS and LLS—and (ii) the popular pertained DnCNNs [81] ($\sigma = 5, 15, 40$). The latter are residual denoisers with 1-Lipschitz convolutional layers and batch normalization modules, which yield a non-averaged denoiser with no convergence guarantees for ill-posed problems. To adapt the strength of the denoisers, in addition to the training noise level, we use relaxed denoisers $\boldsymbol{D}_\gamma = \gamma \boldsymbol{D} + (1-\gamma)\mathbf{Id}$ for all denoisers $\boldsymbol{D}$, where $\gamma \in (0, 1]$ is tuned along with the stepsize α given in (7.44). We only report the performance of the best-performing setting. The ACR framework [286, 302] yields a convex regularizer for (7.2) that is specifically designed

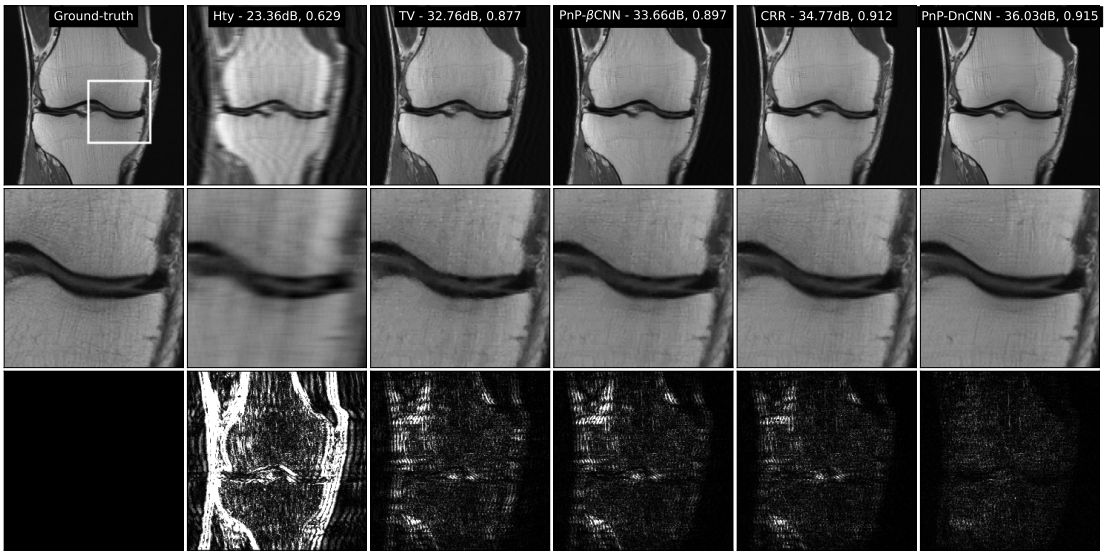

Figure 7.3: Reconstructed images for the 4-fold accelerated multi-coil MRI experiment. The reported metrics are PSNR and SSIM. The last row shows the squared differences between the reconstructions and the ground-truth image.

for the described CT problem. To be consistent with [286, 302], we apply 400 iterations of gradient descent, even though the objective is nonsmooth, and tune the stepsize and λ. The results are consistent with those reported in [286, 302].

To assess the dependence of CRR-NNs on the image domain, we also train models for Gaussian denoising of CT and MRI images ($t = 10$, $\sigma \in \{5/255, 25/255\}$). The training procedure is the same as for BSD image denoising, but a larger kernel size of 11 was required to saturate the performance. The learned filters and activations are included in Appendix 7.8.3.

The hyperparameters for all these methods are tuned to maximize the average PSNR over the validation set with the coarse-to-fine method given in Appendix 7.8.1.

Results and Discussion For each modality, a reconstruction example is given for each framework in Figures 7.3 and 7.4, and additional illustrations are given in Appendix 7.8.4 Material. For PnP with average denoiser, we only show the reconstruction obtained with the LLS activation, since it is always of superior quality than with the other activations. The PSNR and SSIM values for the test set given in Tables 7.3, 7.5, and 7.7 attest that CRR-NNs consistently outperform the other frameworks with comparable guarantees. It can be seen from Tables 7.4, 7.6, and 7.8 that the improvements hold for all setups explored to trained CRR-NNs. The training of CRR-NNs on the target image domain allows for an additional small performance boost. The performances of CRR-NNs are close to the ones of PnP-DnCNN, which has however no guarantees and little interpretability. PnP-DnCNN

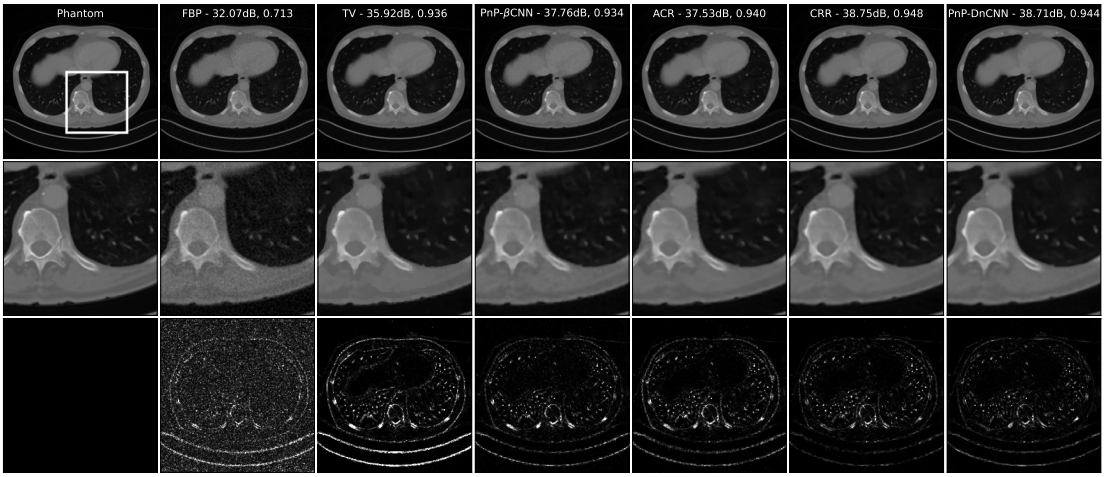

Figure 7.4: Reconstructed images for the CT experiment with $\sigma_{\mathbf{n}} = 0.5$. The reported metrics are PSNR and SSIM. The last row shows the squared differences between the reconstructions and the ground-truth image.

typically yields artifact-free reconstructions but is more prone to over-smoothing (Figure 7.3) or even to exaggeration of some details in rare cases (see Figures in Appendix 7.8.4). Lastly, observe that the properly constrained PnP-βCNN developed in Chapter 6 is not competitive with CRR-NNs, which illustrates the benefit of using explicit regularization in this setting. This confirms the difficulty of training provably 1-Lipchitz CNN, which is also reported for MRI image reconstruction in [315]. Convergence curves for CRR-NNs can be found in Appendix 7.8.2.

7.6.4 Under the Hood of the Learnt Regularizers

The filters and activation functions for learned CRR-NNs with $\sigma \in \{5/255, 25/255\}$ and $t = 5$ are shown in Figures 7.5 and 7.6.

Filters

The impulse responses of the filters vary in orientation and frequency response. This indicates that the CRR-NN decouples the frequency components of patches. The learned kernels typically come in groups that are reminiscent of 2D steerable filters [317, 318]. Interestingly, their support is wider when the denoising task is carried out for $\sigma = 25/255$ than for $\sigma = 5/255$.

Activation Functions

The linear splines converge to simple functions throughout the training. The regularization (7.36) leads to even simpler ones without a compromise in performance. Most of them end up with 3 linear regions, with their shape being reminiscent of the clipping function $\mathrm{Clip}(x) = \mathrm{sign}(x)\min(|x|, 1)$. The learned regularizer is closely related to ℓ_1-norm based regularization as many of the learned convex profiles ψ_i resemble some smoothed version of the absolute-value function.

Pruning CRR-NNs

Since the NN has a simple architecture, it can be efficiently pruned before inference by removal of the filters associated with almost-vanishing activation functions. This yields models with typically between 3000 and 5000 parameters and offers a clear advantage over deep models, which can usually not be pruned efficiently.

A Signal-Processing Interpretation

Given that the gradient-step operator $\mathbf{x} \mapsto (\mathbf{x} - \alpha \mathbf{W}^T \boldsymbol{\sigma}(\mathbf{W}\mathbf{x}))$ of the learned regularizer is expected to remove some noise from $\mathbf{x}$, the 1-hidden-layer CNN $\mathbf{W}^T\boldsymbol{\sigma}(\mathbf{W}\cdot)$ is expected to extract noise. The response of $\mathbf{x}$ to the learned filters forms the high-dimensional representation $\mathbf{W}\mathbf{x}$ of $\mathbf{x}$. The clipping function preserves the small responses to the filters, while it cuts the large ones. Hence, the estimated noise $\mathbf{W}^T\boldsymbol{\sigma}(\mathbf{W}\mathbf{x})$ is reconstructed by essentially removing the components of $\mathbf{x}$ that exhibit a significant correlation with the kernels of the filters. All in all, the learning of the activation functions leads closely to wavelet- or framelet-like denoising. Indeed, the proximal operator of $\mathbf{x} \mapsto \|\mathrm{DWT}(\mathbf{x})\|_1$ is given by

$$\begin{aligned} \mathrm{prox}_{\|\mathrm{DWT}(\cdot)\|_1}(\mathbf{x}) &= \mathrm{IDWT}(\mathrm{soft}(\mathrm{DWT}(\mathbf{x}))) \\ &= \mathbf{x} - \mathrm{IDWT}(\mathrm{clip}(\mathrm{DWT}(\mathbf{x}))), \end{aligned} \tag{7.48}$$

where $\mathrm{soft}(\cdot)$ is the soft-thresholding function, DWT and IDWT are the orthogonal discrete wavelet transform and its inverse, respectively. The equivalent formulation with the clipping function follows from $\mathrm{IDWT}(\mathrm{DWT}(\mathbf{x})) = \mathbf{x}$ and $\mathrm{soft}(\mathbf{x}) = (\mathbf{x} - \mathrm{clip}(\mathbf{x}))$. The soft-thresholding function is used for direct denoising while the clipping function is tailored to residual denoising. Note that the given analogy is, however, limited since the learned filters are not orthonormal ($\mathbf{W}^T\mathbf{W} \neq \mathbf{I}$).

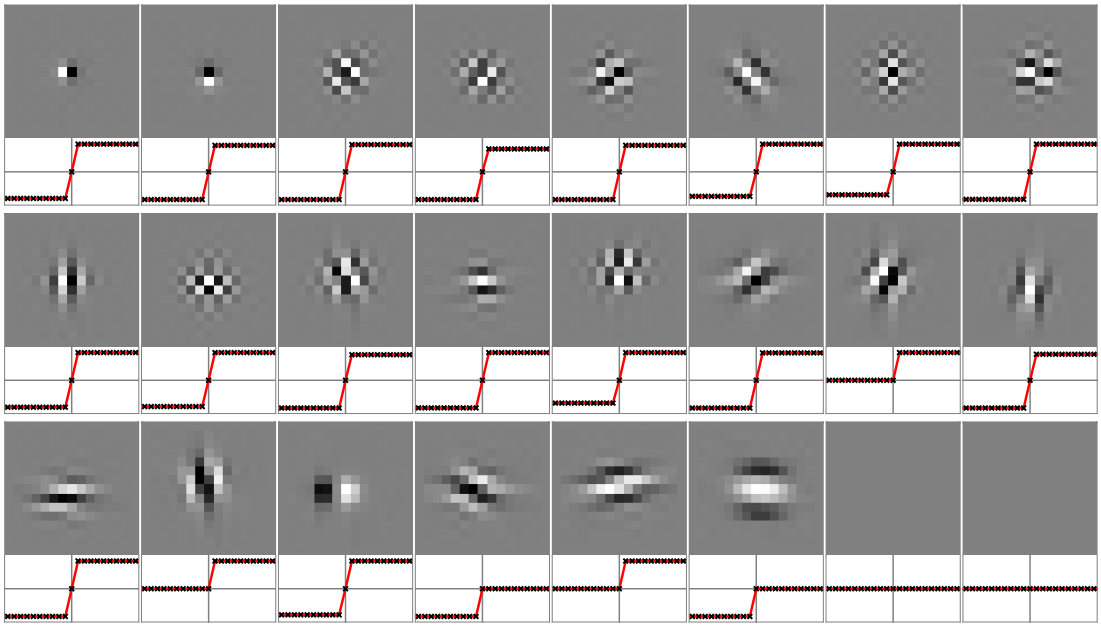

Figure 7.5: Impulse response of the filters and activation functions of the CRR-NN trained with $\sigma = 5$. The crosses indicate the knots of the splines. For the 8 missing filters, the associated activation functions were numerically identically zero.

Role of the Scaling Factor

To clarify the role of the scaling factor μ introduced in (7.42), we investigate a toy problem on the space of one-dimensional signals. Since these can be interpreted as images varying along a single direction, a signal regularizer R_1 can be obtained from $R_{\boldsymbol{\theta}}$ by replacing the 2D convolutional filters with 1D convolutional filters whose kernels are the ones of $R_{\boldsymbol{\theta}}$ summed along a direction. Next, we seek a compactly supported signal with fixed mass that has minimum regularization cost, as in

$$\hat{\mathbf{c}} = \underset{\mathbf{c}\in\mathbb{R}^d}{\arg\min}\, R_1(\mu\mathbf{c}) \;\text{ s.t. }\; \begin{cases} \mathbf{1}^T\mathbf{c} = 1, \\ \mathbf{c}_k = 0, \qquad \forall k \notin [k_1, k_2]. \end{cases} \tag{7.49}$$

The solutions for various values of μ are shown in Figure 7.7. Small values of μ promote smooth functions in a way reminiscent of the Tikhonov regularizer applied to finite differences. Large values of μ promote functions with constant portions and, conjointly, allow for sharp jumps, which is reminiscent of the TV regularizer. This reasoning is in agreement with the shape of the activation functions shown in Figures 7.5 and 7.6. Indeed, an increase in μ allows one to enlarge the region where the regularizer has constant gradients, while a decrease of μ allows one to enlarge the region where the regularizer has linear gradients.

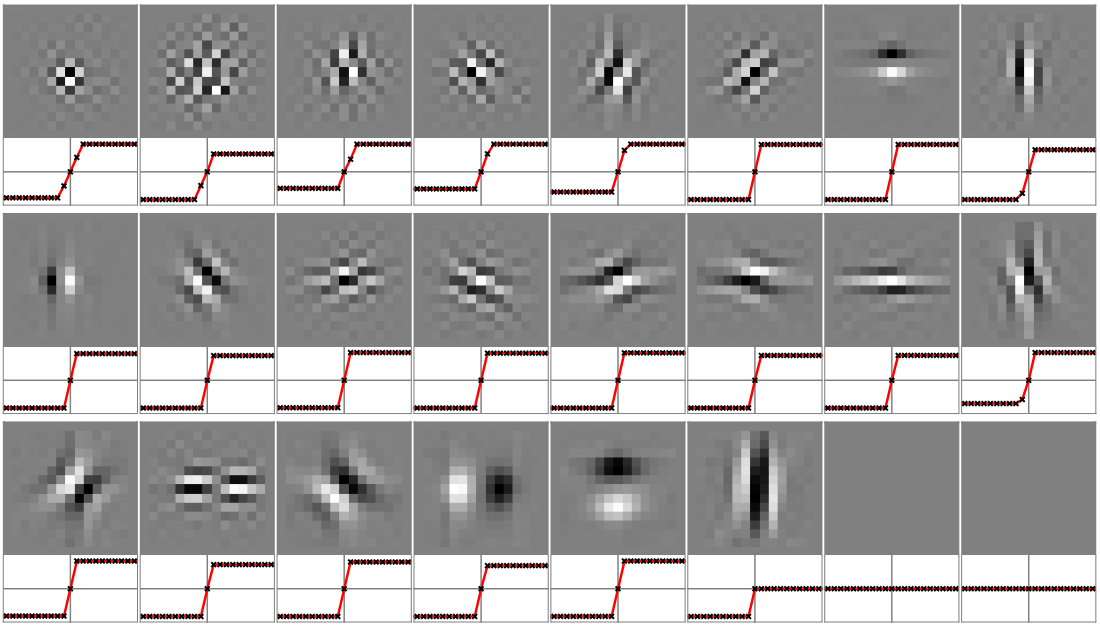

Figure 7.6: Impulse response of the filters and activation functions of the CRR-NN trained with $\sigma = 25/255$.

7.7 Conclusion

We have proposed a framework to learn universal convex-ridge regularizers with adaptive profiles. When applied to inverse problems, it is competitive with those recent deep-learning approaches that also prioritize the reliability of the method. Not only CRR-NNs are faster to train, but they also offer improvements in image quality. The findings raise the question of whether shallow models such as CRR-NNs, despite their small number of parameters, already offer optimal performance among methods that rely either on a learnable convex regularizer or on the PnP framework with a provably averaged denoiser. In the future, CRR-NNs could be fine-tuned on specific modalities via the use of $\mathbf{H}$ for training. This could further improve the reconstruction quality, as observed when shifting from PnP to deep unrolled algorithms while maintaining the guarantees.

7.8 Appendix

7.8.1 Hyperparameter Tuning

The parameters λ and μ used in (7.42) can be tuned with a coarse-to-fine approach. Given the performance on the 3×3 grid $\{(\gamma_\lambda)^{-1}\lambda, \lambda, \gamma_\lambda\lambda\} \times \{(\gamma_\mu)^{-1}\mu, \mu, \gamma_\mu\mu\}$, we identify the best values λ^* and μ^* on this subset and move on to the next iteration as follows:

- if $\lambda^* = \lambda$, we refine the search grid by reducing γ_μ to $(\gamma_\mu)^\zeta$, $\zeta < 1$;

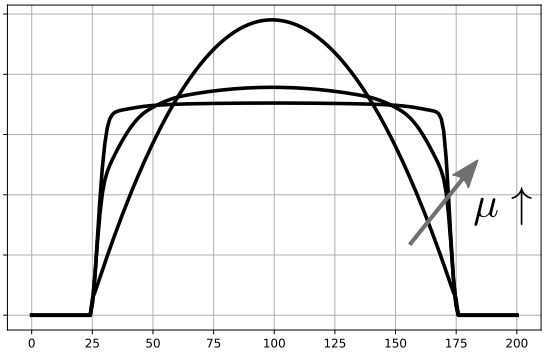

Figure 7.7: Solutions of the one-dimensional problem (7.49) for increasing values of μ. The plotted functions are supported in $[25, 175]$ and minimize the learned regularizer given a unit sum of their values.

- otherwise, λ is updated to λ^*.

A similar update is performed for the scaling parameter. The search is terminated when both γ_λ and γ_μ are smaller than a threshold, typically, 1.01. In practice, we initialized $\gamma_\lambda = \gamma_\mu = 4$ and set $\zeta = 0.5$. The method usually requires between 50 and 100 evaluations on tuples (λ, μ) on the validation set before it terminates. The proposed approach is predicated on the observation that the optimization landscape in the (λ, μ) domain is typically well-behaved. The same principles apply to tune a single hyperparameter, as found in the TV and the PnP-βCNN methods. Let us remark that the performance was found to change only slowly with the scaling parameter μ for the MRI and CT experiments. Hence, in practice, it is enough to tune μ very coarsely.

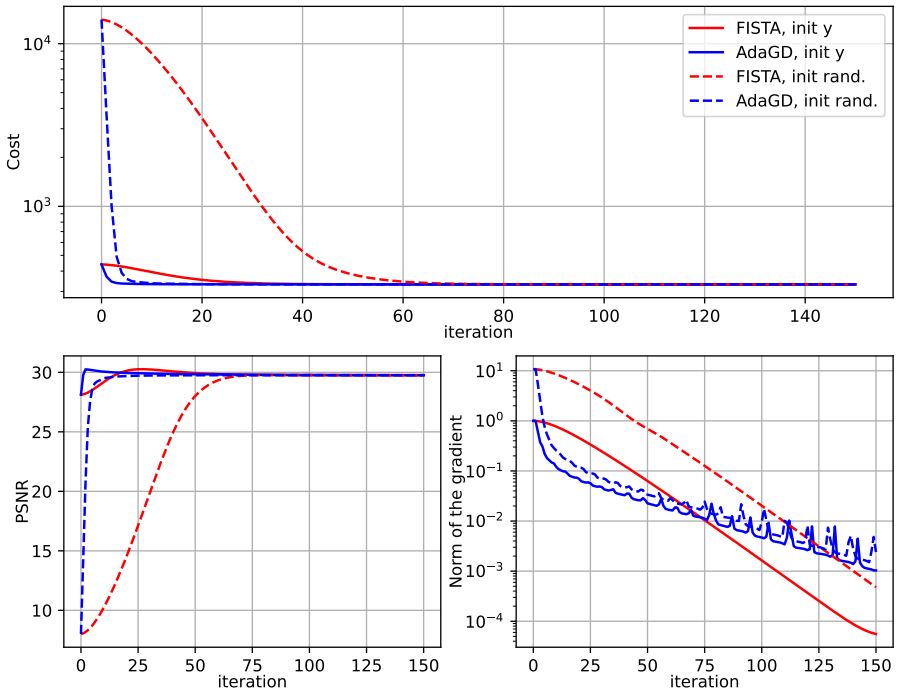

Figure 7.8: Example of convergence curves (denoising).

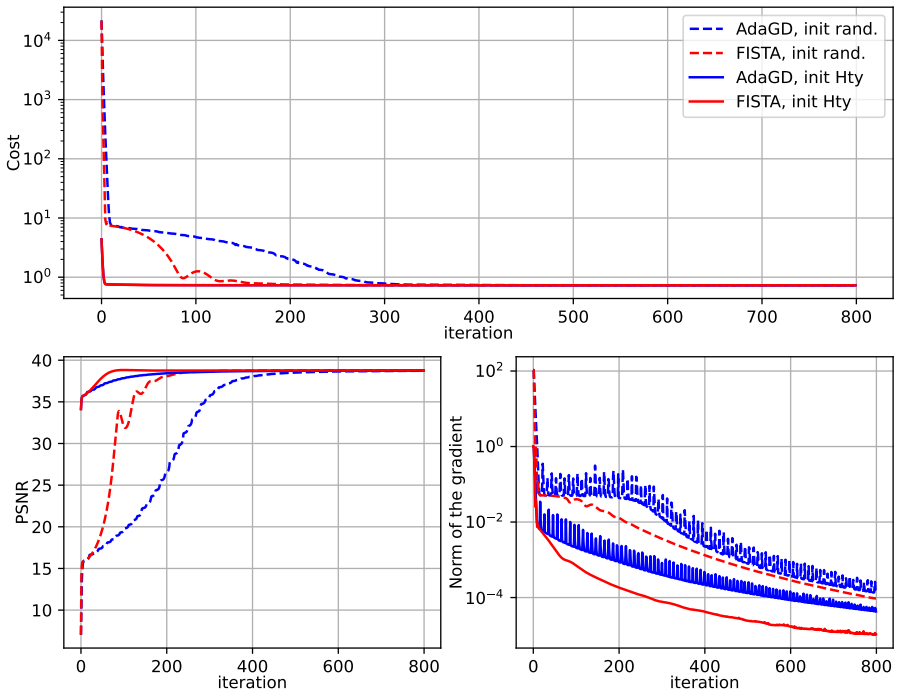

Figure 7.9: Example of convergence curves (MRI).

7.8.2 Convergence Curves

In this section, we present convergence curves for image denoising (Figure 7.8), MRI reconstruction (Figure 7.9), and CT reconstruction (Figure 7.10) with CRR-NNs. The underlying objective is minimized with FISTA[5] [6][28] and AdaGD[5][312], which both converge generally fast. Depending on the task and the desired accuracy, one or the other might be faster. The observed gradient-norm oscillations for AdaGD are typical for this method and unrelated to CRR-NNs [312]. Finally, note that the initialization affects the convergence speed, but does not impact the reconstruction quality. This differs significantly from PnP methods that deploy loosely constrained denoisers.

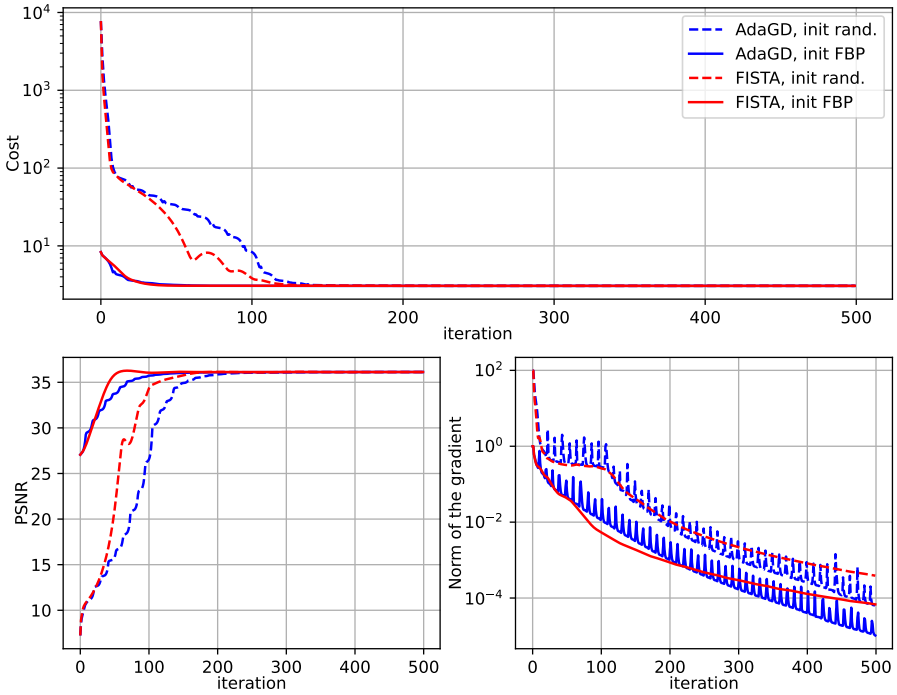

Figure 7.10: Example of convergence curves (CT).

7.8.3 Activations and Filters

We provide the filters and activations of a CRR-NN trained for the denoising of CT images (Figure 7.11) and of MRI images (Figure 7.12). Compared to the training on the BSD500 dataset, larger kernel sizes were needed to saturate the performances.

[5]For the plots, the positivity constraint is dropped, otherwise, the gradient does not necessarily vanish at the minimum.

[6]For denoising, the problem is 1-strongly convex. Hence, we use Nesterov's rule $(1-\sqrt{L})/(1+\sqrt{L})$ instead of $(t_k-1)/t_{k+1}$ for extrapolation [319].

7.8.4 Reconstructed images

MRI In Figures 7.13 and 7.14, we present reconstructions from multi- and single-coil MRI measurements, and report their PSNR and SSIM as metrics. The reconstruction task in Figure 7.14 is particularly challenging. In this regime, it can be observed that the loosely constrained PnP-DnCNN exaggerates some structures, even though the metrics remain acceptable.

CT In Figures 7.15 and 7.16, we provide reconstructions for the CT experiments with noise levels $\sigma_{\mathbf{n}} = 1, 2$ in the measurements. The reported metrics are PSNR and SSIM.

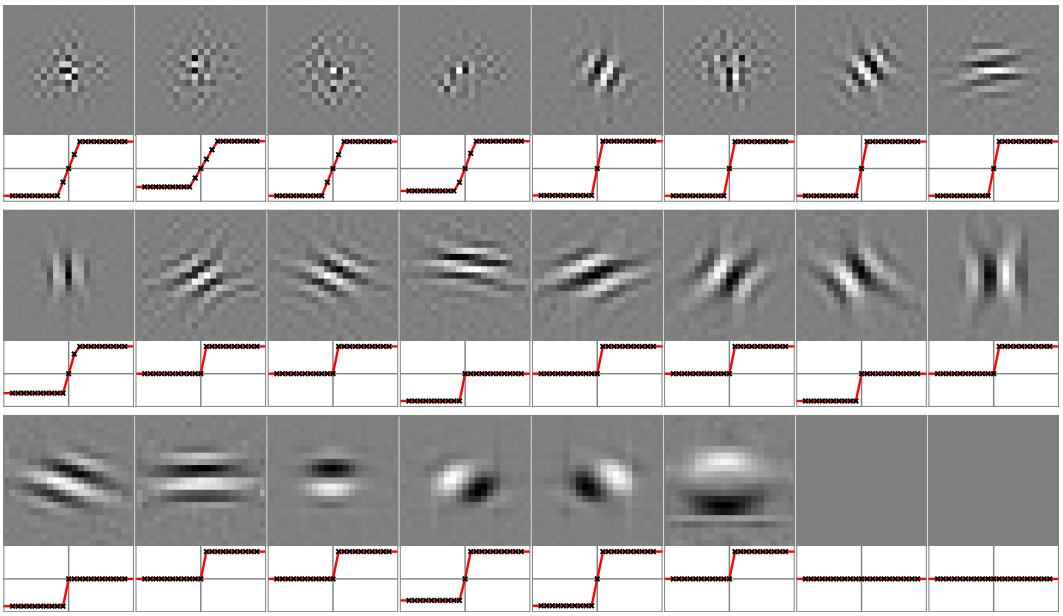

Figure 7.11: Impulse response of the filters and activation functions of the CRR-NN trained to denoise CT images.

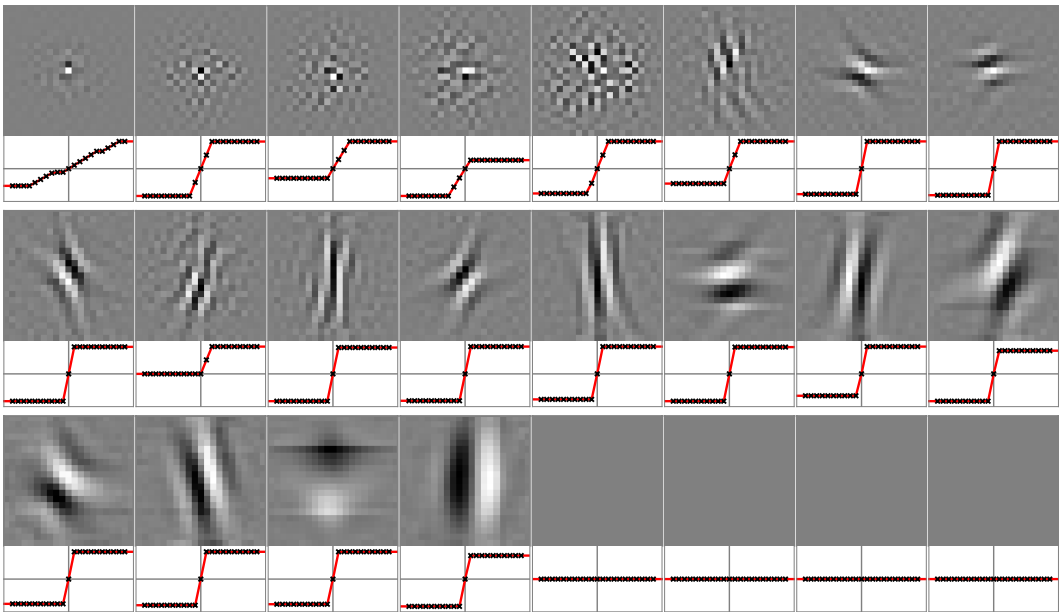

Figure 7.12: Impulse response of the filters and activation functions of the CRR-NN trained to denoise MRI images.

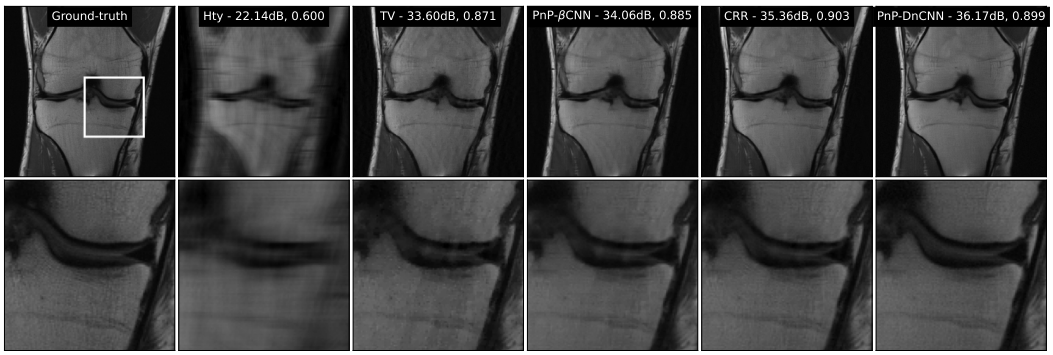

Figure 7.13: Reconstructions for the 8-fold accelerated multi-coil MRI experiment.

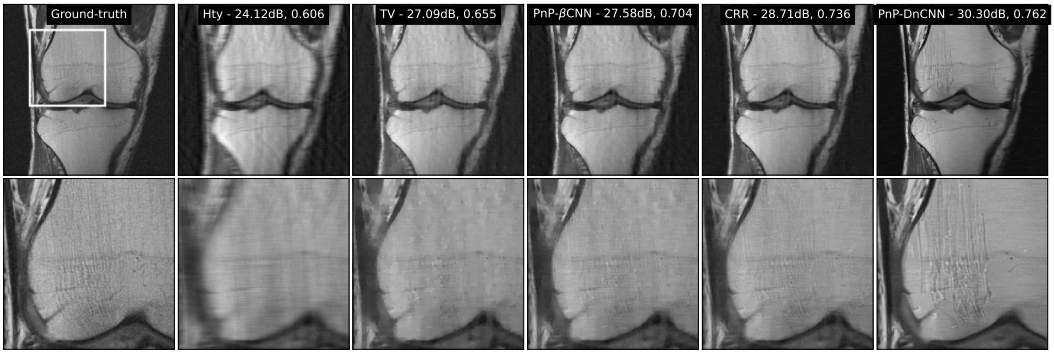

Figure 7.14: Reconstructions for the 4-fold accelerated single-coil MRI experiment. Note the unexpected behavior of DnCNN.

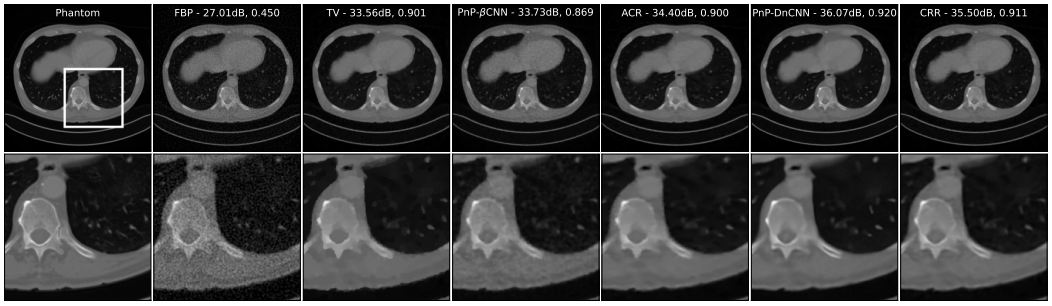

Figure 7.15: Reconstructed images for the CT experiment with $\sigma_{\mathbf{n}} = 1.0$.

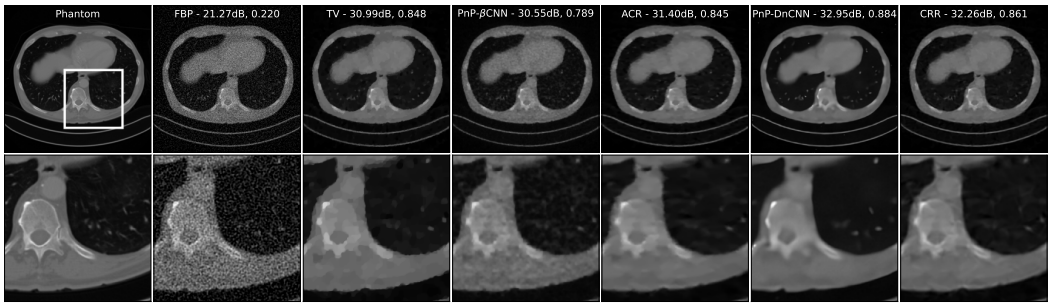

Figure 7.16: Reconstructed images for the CT experiment with $\sigma_{\mathbf{n}} = 2.0$.

8 Learning Weakly Convex Regularizers for Convergent Image-Reconstruction Algorithms

The text of this chapter is adapted from the in press paper

A. Goujon, S. Neumayer and M. Unser "Learning weakly convex regularizers for convergent image-reconstruction algorithms", accepted in *SIAM Journal on Imaging Sciences*, 2023.

8.1 Summary

We propose to learn non-convex regularizers with a prescribed upper bound on their weak-convexity modulus. Such regularizers give rise to variational denoisers that minimize a convex energy. They rely on a few parameters (less than 15,000) and offer a signal-processing interpretation as they mimic classical sparsity-promoting regularizers. Through numerical experiments, we show that such denoisers outperform convex-regularization methods as well as the popular BM3D denoiser. Additionally, the learned regularizer can be deployed to solve inverse problems with iterative schemes that provably converge. For both CT and MRI reconstruction, the regularizer generalizes well and offers an excellent tradeoff between performance, number of parameters, guarantees, and interpretability when compared to other data-driven approaches.

8.2 Introduction

8.2.1 Linear Inverse Problems

Linear inverse problems are ubiquitous in imaging, with applications in medical imaging [16], including magnetic resonance imaging (MRI) and X-ray computed tomography (CT). In a discretized linear inverse problem [18], the goal is to reconstruct an (unknown) image of interest $\mathbf{x} \in \mathbb{R}^d$ from a given noisy observation

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \in \mathbb{R}^m, \tag{8.1}$$

where $\mathbf{H} \in \mathbb{R}^{m \times d}$ denotes the measurement operator and $\mathbf{n} \in \mathbb{R}^m$ is a noise term. To overcome a possibly ill-conditioned $\mathbf{H}$ and the presence of noise, it is standard to compute the reconstruction $\hat{\mathbf{x}}$ as a solution of the variational problem

$$\hat{\mathbf{x}} = \operatorname*{arg\,min}_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + R(\mathbf{x}), \tag{8.2}$$

where the regularizer $R \colon \mathbb{R}^d \to \mathbb{R}$ incorporates prior information about $\mathbf{x}$. Convex regularizers, such as the Tikhonov [19] or total-variation (TV) [22, 30] ones, are popular as they allow one to efficiently solve (8.2). Unfortunately, such regularizers do not yield state-of-the-art reconstructions and have known limitations. For instance, they typically struggle to preserve textures in the image $\mathbf{x}$ [320].

8.2.2 The Convex Non-Convex Framework for Denoising

The reliance on a well-chosen non-convex R leads to improved performance [292, 293], with the caveat that finding a global minimum of (8.2) becomes intractable in general. A possible remedy is provided by the convex non-convex (CNC) framework. It consists in the deployment of a non-convex R_{CNC} such that the global objective

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2} \|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + R_{\mathrm{CNC}}(\mathbf{x}) \tag{8.3}$$

is convex, see [321] for an overview. Over the past years, the use of CNC approaches has led to improved results in various settings, including dictionary learning [322], plug-and-play (PnP) algorithms [76, 323], and matrix completion [324].

For the case of image denoising, namely $\mathbf{H} = \mathbf{I}$, the data-fidelity term $\frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2$ is 1-strongly convex. Hence, to ensure the convexity of $\mathcal{J}$, the regularizer R_{CNC} needs to be 1-weakly convex from the definition of weak convexity (see Section 8.3). Various strategies have been proposed to design a weakly convex R_{CNC}.

Explicit Design The commonly used $\|\cdot\|_1$-norm for sparse regularization can be replaced by a non-convex penalty function that better mimics the behavior of the $\|\cdot\|_0$-norm while ensuring that one remains within the CNC framework. This includes properly scaled versions of the logarithm and the minimax concave penalty [325]. Although non-convex, these functions are quasi-convex. In particular, they are such that large values are more penalized than smaller ones. The potentials are then combined with convolutional filters. This yields, for instance, TV-like regularizers [326, 327], which extend and improve upon their convex counterparts.

Implicit Design with Moreau Envelopes There exists a systematic method to convert any convex regularizer into a nonconvex but still 1-weakly convex one, utilizing its (generalized) Moreau envelope [321, 324]. Such a regularizer, however, does not admit a closed form, and the existing algorithms to solve (8.2) involve a computationally intensive bilevel optimization task.

Implicit Design via the Learning of Proximal Operators Although R_{CNC} is non-convex, its proximity operator $\mathrm{prox}_{R_{\mathrm{CNC}}}$ is well-defined under mild conditions [97]. In [76], the authors propose to directly learn $\mathrm{prox}_{R_{\mathrm{CNC}}}$ such that it is a good Gaussian denoiser. To do so, they explicitly parameterize the proximal operator, in line with the recently introduced gradient-step denoisers [102, 273]. More precisely, they express the residual map $(\mathrm{prox}_{R_{\mathrm{CNC}}} - \mathrm{Id})$ as the gradient of a deep convolutional neural network (CNN), and require that the residual is contractive by enforcing that it has a Lipschitz constant smaller than 1. This yields excellent performance, with the caveat that it is challenging to enforce strict Lipschitz constraints on the gradient of a CNN. For this reason, the authors of [76] propose to regularize the spectral norm of the Jacobian of $(\mathrm{prox}_{R_{\mathrm{CNC}}} - \mathrm{Id})$ at a finite number of locations. This method works well in practice but does not offer any provable guarantee on the weak-convexity property of the underlying (implicit) objective $\mathcal{J}$.

8.2.3 Extension to Ill-Posed Inverse Problems

The design of CNC models is difficult when the forward matrix $\mathbf{H}$ is noninvertible. Since the data term $\frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2$ is not strongly convex anymore, the 1-weak convexity of R is not sufficient. Then, the condition on R depends on $\mathbf{H}$, and CNC models are therefore usually tailored to a specific problem. One can partially circumvent this limitation by combining a proximal algorithm with a generic weakly convex regularizer, for which the proximal operator is well-defined. The convergence to stationary points of the objective is established in [76, 98] for the forward-backward splitting [28] based on the very general convergence result for functions with the Kurdyka-Łojasiewicz (KL) property given in [96]. When R is differentiable, as will be assumed in our setting, similar results can be obtained for gradient descent applied to the non-convex objective (8.2), see [96]. From a stochastic perspective, it is known that such first-order methods do not get trapped into strict saddle points of the objective [328]. This is a possible explanation for the good empirical performance of non-convex reconstruction frameworks.

8.2.4 Other Deep-Learning-Based Variational Methods with Some Guarantees

The emergence of deep-learning-based methods has led to significant improvements in the quality of reconstruction for inverse problems. Yet, due to the black-box nature of

deep NNs, this often comes with a loss of interpretability and reliability. Thus, there is a growing interest to mitigate these limitations, see [329] for a survey. In the following, we briefly comment on works that rely on the variational formulation (8.2) with a learned regularizer R but that are not directly within the CNC framework. To provide maximal theoretical guarantees within iterative image reconstruction, it was proposed in [286] to learn a convex R based on a deep CNN, and shown in [330] that a shallow model, namely, a convex ridge regularizer NN (CRR-NN) with few parameters, was sufficient. The latter offers the opportunity to learn a collection of filters and sparsity-promoting profile functions to build R. This is inspired by the Fields-of-Experts (FoE) framework [292] and its many variants, such as [293], to design and learn a non-convex R. While [293] yields good performance, it does not guarantee that the objective is convex. Another popular extension of FoE is trainable nonlinear reaction diffusion (TNRD) [331]. There, the minimization scheme associated with (8.2) is unrolled and different filters and potential functions are learned at each step. This improves the performance over [293] but does not correspond to an energy minimization anymore. More recently, all these frameworks have been unified in the context of variational networks [300]. The combination of these with recent findings in deep CNN research and early stopping techniques has then led to the total deep variation framework [298]. Although this model has several layers, some interpretability remains possible through an eigenfunction analysis. Another deep-learning-based variational method with convergence guarantees and a regularization scheme is found in [297].

8.2.5 Outline and Main Contributions

In this chapter, we propose a framework to learn a 1-weakly convex regularizer that yields an interpretable proximal denoiser. The general framework is introduced in Section 8.3. Then, the principal contributions are as follows.

- **Denoising**: In Section 8.4, we propose a scheme for the training of weakly-convex-ridge-regularizer neural networks (WCRR-NN), with a significant increase in performance over their convex counterparts but with the same guarantees and interpretability. Based on a condition introduced in Proposition 8.2, the associated denoising problem is convex, which allows for global minimization. Numerical experiments indicate that the learning of both the profiles and the filters leads to a sparsity prior that is state-of-the-art in the CNC framework across various noise levels for the BSD68 test set. In particular, it is the first convex-energy-based model that outperforms BM3D [332], which has been one of the most popular benchmarks for nearly 15 years now.

- **Inverse Problems**: In Section 8.5, we deploy the learned regularizer to solve generic inverse problems by minimizing (8.2) with an accelerated gradient-descent (AGD) scheme that is tailored to our weakly convex regularizer (Algorithm 2). Further,

we prove that the algorithm reaches some critical point of the objective (Theorem 8.1). Numerical experiments for CT and MRI demonstrate that the regularizer empirically generalizes well. We find that it outperforms several energy-based reconstruction methods that come with convergence guarantees.

Finally, conclusions are drawn in Section 8.6. The full implementation will be released before publication and pre-trained models can be accessed upon request.

8.3 Weakly Convex Regularizers

Our goal is to construct a regularizer R for the variational reconstruction model (8.2) that performs well across a variety of inverse problems while maintaining the theoretical guarantees and interpretability of classical schemes. A particularly promising direction is given by the CNC framework, where one can efficiently find a global minimum of the objective in (8.2). As commonly done in practice, our strategy is to design and train the regularizer based on the denoising task

$$\hat{\mathbf{x}} = \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + R(\mathbf{x}), \tag{8.4}$$

where $\mathbf{y}$ is a noisy version of a clean image. The minimization of (8.2) for generic inverse problems and weakly convex regularizers is then discussed in Section 8.5.

To obtain a CNC model in (8.4), R needs to be 1-weakly convex so that the overall objective remains convex.

Definition 8.1. *A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is*

i) convex *if $f(\lambda\mathbf{x}+(1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y})$ for all $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$ and $\lambda\in[0,1]$;*

ii) ρ-strongly convex *if $(f - \frac{\rho}{2}\|\cdot\|^2)$ is convex with $\rho \geq 0$;*

iii) and ρ-weakly convex *if $f + \frac{\rho}{2}\|\cdot\|^2$ is convex with $\rho \geq 0$.*

Note that a ρ-weakly convex function is also μ-weakly convex for any $\mu \geq \rho$. A convex R is ρ-weakly convex for any $\rho \geq 0$ and, in particular, 0-weakly convex. For a differentiable R, convexity is equivalent to the monotonicity of ∇R. Hence, a differentiable R is ρ-weakly convex iff

$$(\nabla R(\mathbf{y}) - \nabla R(\mathbf{x}))^T(\mathbf{y}-\mathbf{x}) \geq -\rho\|\mathbf{y}-\mathbf{x}\|_2^2, \tag{8.5}$$

for any $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$. Given a twice-differentiable R, ρ-weak convexity is equivalent to

$$H_R(\mathbf{x}) \succeq -\rho\mathbf{I} \tag{8.6}$$

for any $\mathbf{x} \in \mathbb{R}^d$, where $H_R(\mathbf{x})$ denotes the Hessian of R at $\mathbf{x}$. In other words, the Hessian of a ρ-weakly convex function has all its eigenvalues in the range $[-\rho, +\infty)$.

Remark 8.1. *Any differentiable function R with L-Lipschitz gradient is L-weakly convex. This estimate is, however, not necessarily tight, in the sense that R might also be ρ-weakly convex for some $0 \leq \rho \ll L$ all the way to zero. For instance, any convex R with L-Lipschitz gradient is L-weakly convex but it is also trivially 0-weakly convex because it is equivalent to being convex.*

Weak convexity provides more flexibility, while still maintaining most of the desirable properties of usual convex-regularization frameworks. In particular, the proximal operator

$$\operatorname{prox}_R(\mathbf{y}) = \underset{\mathbf{x}\in\mathbb{R}^d}{\arg\min} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + R(\mathbf{x}) \tag{8.7}$$

is well-defined for any ρ-weakly convex R with $\rho < 1$. Indeed, the objective in (8.7) is $(1-\rho)$-strongly convex, which ensures the existence of a unique minimizer. The properties of the proximal operator in a generic non-convex setting are characterized in detail in [97]. The main implication here is the Lipschitz continuity of our denoiser (8.7) (Proposition 8.1).

Proposition 8.1 ([97])**.** *For any ρ-weakly convex regularizer R with $\rho < 1$, there exists a convex lower semi-continuous potential $g\colon \mathbb{R}^d \to \mathbb{R}$ such that $\operatorname{prox}_R(\mathbf{x}) \in \partial g(\mathbf{x})$ holds for every $\mathbf{x} \in \mathbb{R}^d$. Conversely, the subgradient of any such g coincides with prox_R for some R that is 1-weakly convex on any convex subset of its domain. Furthermore, prox_R is $(\frac{1}{1-\rho})$-Lipschitz, in the sense that*

$$\|\operatorname{prox}_R(\mathbf{y}_2) - \operatorname{prox}_R(\mathbf{y}_1)\|_2 \leq \frac{1}{1-\rho}\|\mathbf{y}_2 - \mathbf{y}_1\|_2 \tag{8.8}$$

for any $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^d$. More generally, C^{k+1} regularity of the potential g leads to C^k regularity of prox_R. Finally, prox_R is invertible on its range in this setting.

For a convex regularizer R, $\rho = 0$ and, hence, prox_R is non-expansive (1-Lipschitz). For a non-convex but weakly convex R, namely $\rho > 0$, this is not necessarily the case anymore. We conjecture that this is key to the boost in performance since non-expansive denoisers have intrinsic limitations, see for instance [330, Fig. 1].

8.4 Design of a Learnable and Provably 1-Weakly Convex Regularizer for Denoising

In this section, we discuss the construction and training of R and compare it with several other variational frameworks.

8.4.1 Regularizer Architecture

The weakly convex regularizer R is chosen as the sum of convolutional ridges

$$R\colon x[\cdot] \mapsto \sum_{i=1}^{N_C} \sum_{\mathbf{k}\in\mathbb{R}^2} \psi_i\Big((h_i * x)[\mathbf{k}]\Big), \tag{8.9}$$

where $x[\cdot]$ represents a 2D image, $(h_i[\cdot])_{i=1}^{N_C}$ are the impulse responses of a collection of linear and shift-invariant filters, and $(\psi_i)_{i=1}^{N_C}$ are potential functions with Lipschitz continuous derivative. In practice, the finite-size input images are zero-padded so that the outputs of the convolutions have the same spatial size as the input image. We also choose potential functions with a shared profile ψ, so that $\psi_i = \alpha_i^{-2}\psi(\alpha_i\cdot)$ with $\alpha_i > 0$. As the $h_i[\cdot]$ can absorb the α_i in the definition of ψ_i, this is just a different parameterization for adding weights α_i^{-2} in front of the profile ψ. The advantage of our parameterization is that $\mathrm{Lip}(\psi_i')$ does not depend on α_i, which will simplify the reasoning throughout this section. The number N_C of filters is also referred to as the number of channels or feature maps of the model. The motivations behind our choice are threefold.

- **Interpretability.** The model (8.9) includes many traditional compressed-sensing regularizers. Consequently, it has a simple signal-processing interpretation, see Section 8.4.5. The parameters for deeper CNN-based regularizers are usually much harder to interpret than those in (8.9).
- **Control of ρ.** The weak-convexity modulus of (8.9) can be upper-bounded using Proposition 8.2. This is far less obvious for deeper CNN architectures. There, weak convexity is usually promoted via regularization during training [76]. While this works qualitatively, it does not generate provably ρ-weakly convex maps for some prescribed ρ.
- **Model Expressivity.** There is evidence that, in constrained settings, (8.9) has good expressive power. For instance, when learning convex regularizers for (8.2), architectures of the form (8.9) are on par with deep CNNs such as the input convex NN (ICNN), all the while depending on much fewer parameters [330].

To simplify the notation in the sequel, the regularizer (8.9) is written whenever needed in the generic form

$$R\colon \mathbf{x} \mapsto \sum_{j=1}^{d\times N_C} \psi_j(\mathbf{w}_j^T\mathbf{x}), \tag{8.10}$$

where $\mathbf{x} = (x[\mathbf{k}])_{\mathbf{k}\in\Omega} \in \mathbb{R}^d$ is the vectorized representation of $x[\cdot]$, the $\mathbf{w}_j \in \mathbb{R}^d$ correspond to shifted versions of the filter kernels, and j indexes at the same time along the channels and the 2D shifts of the kernels. The gradient of this differentiable regularizer R reads

$$\boldsymbol{\nabla} R(\mathbf{x}) = \mathbf{W}^T\,\boldsymbol{\varphi}(\mathbf{W}\mathbf{x}), \tag{8.11}$$

where $\mathbf{W} = [\mathbf{w}_1 \cdots \mathbf{w}_{dN_C}]^T \in \mathbb{R}^{dN_C \times d}$ and $\boldsymbol{\varphi}$ is the pointwise *activation function* given by $\boldsymbol{\varphi}(\mathbf{z}) = (\psi_j'(z_j))_{j=1}^{dN_C} = (\alpha_j^{-1}\psi'(\alpha_j z_j))_{j=1}^{dN_C}$. Note that $\mathbf{W}\mathbf{x}$ is a multichannel filtered version of the image $\mathbf{x}$. Since ψ can absorb the spectral norm of $\mathbf{W}$, we enforce that $\|\mathbf{W}\| = 1$, where $\|\cdot\|$ denotes the spectral norm, in order to remove some redundancy from the model and simplify the explanations.

In the following, we use that the Lipschitz continuity of ψ' implies differentiability of ψ' almost everywhere (Rademacher's theorem) and that the essential infimum $\operatorname{ess\,inf}_{t\in\mathbb{R}} \psi''(t)$ is well-defined and satisfies $|\operatorname{ess\,inf}_{t\in\mathbb{R}} \psi''(t)| \leq \operatorname{Lip}(\psi')$.

Lemma 8.1. *Let $\psi\colon \mathbb{R} \to \mathbb{R}$ have Lipschitz continuous derivative. Then ψ is ρ-weakly convex for any $\rho \geq s_{\text{inf}} = \max(0, -\operatorname{ess\,inf}_{t\in\mathbb{R}} \psi''(t))$.*

Proof. The Lipschitz continuity of ψ' implies that $\psi'(t_2) - \psi'(t_1) = \int_{t_1}^{t_2} \psi''(t)\mathrm{d}t$ for any $t_1, t_2 \in \mathbb{R}$. From this, we infer that $(\psi(t_2) - \psi(t_1))(t_2 - t_1) \geq (\operatorname{ess\,inf}_{t\in\mathbb{R}} \psi''(t))(t_2 - t_1)^2$, which is precisely condition (8.5). □

Proposition 8.2. *Any R of the form (7.4) with $\|\mathbf{W}\| = 1$ and a ρ-weakly convex ψ is ρ-weakly convex. In particular, assuming that ψ' is Lipschitz continuous, this holds for any $\rho \geq s_{\text{inf}}$ as defined in Lemma 8.1.*

Proof. Since $\alpha_i > 0$ and $\psi_i = \alpha_i^{-2}\psi(\alpha_i \cdot)$, the convexity of $t \mapsto \psi(t) + \frac{\rho}{2}t^2$ implies the convexity of $t \mapsto \psi_i(t) + \frac{\rho}{2}\alpha_i^{-2}(\alpha_i t)^2$. Thus, $\mathbf{x} \mapsto \psi_j(\mathbf{w}_j^T\mathbf{x}) + \frac{\rho}{2}(\mathbf{w}_j^T\mathbf{x})^2$ and $\mathbf{x} \mapsto R(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{W}\mathbf{x}\|_2^2$ are also convex. Since $\|\mathbf{W}\| = 1$ and $\rho > 0$, $\mathbf{x} \mapsto \frac{\rho}{2}(\|\mathbf{x}\|_2^2 - \|\mathbf{W}\mathbf{x}\|_2^2)$ is convex, and we infer that $\mathbf{x} \mapsto R(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x}\|_2^2$ is convex. □

Hence, we can obtain a 1-weakly regularizer R by enforcing that $s_{\text{inf}} \leq 1$.

Remark 8.2. *The ridge decomposition (8.10) of R is also used within the CRR-NN framework [330], which involves the learning of a convex-ridge regularizer R with learnable spline potentials ψ_j. For CRR-NNs, $s_{\text{inf}} = 0$ is enforced to ensure that the ψ_j are convex. On the contrary, the present WCRR-NN model with $s_{\text{inf}} \in [0, 1]$ has more freedom and therefore extends upon [330].*

Proposition 8.3 (Existence of a minimizer)**.** *Let $\mathbf{H} \in \mathbb{R}^{m\times d}$ and $\psi_i\colon \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, p$, be positive, continuous and piecewise-polynomial functions with finitely many pieces. Then,*

$$\emptyset \neq \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + \sum_{i=1}^{p} \psi_i(\mathbf{w}_i^T\mathbf{x}). \tag{8.12}$$

Proof. Each ψ_i partitions $\mathbb{R}$ into finitely many closed[1] intervals $(I_i^\ell)_{\ell=1}^{L_i}$ on which it is a polynomial, and hence $\psi_i(\mathbf{w}_i^T\cdot)$ partitions $\mathbb{R}^d$ into L_i closed convex regions $\Omega_i^\ell =$

[1] Such a partition with closed interval exists because the profile functions are continuous.

$\{\mathbf{x}\colon \mathbf{w}_i^T\mathbf{x} \in I_i^\ell\}$. Based on theses, we partition $\mathbb{R}^d$ into finitely many polytopes of the form $\cap_{i=1}^p \Omega_i^{\ell_i}$, where $\ell_i \in \{1, \dots, L_i\}$. Since the partition is finite, the infimum of the objective is the infimum of the objective on (at least) one of these closed polytopes, say $P = \cap_{i=1}^p \Omega_i^{\ell_i}$.

Now, we pick a minimizing sequence $(\mathbf{x}_k)_{k\in\mathbb{N}} \subset P$. Due to the coercivity of $\|\cdot\|_2^2$, we get that the sequence $(\mathbf{H}\mathbf{x}_k)_{k\in\mathbb{N}}$ remains bounded. We now note that if $I_i^{\ell_i}$ is bounded, i.e. $1 < \ell_i < L_i$, then $(\mathbf{w}_i^T\mathbf{x}_k)_{k\in\mathbb{N}}$ is bounded. Otherwise, the interval $I_i^{\ell_i}$ is unbounded, and since ψ_i is a positive polynomial on it, it is either coercive or constant[2]. Let $\mathbf{M}$ be the matrix whose rows are the rows of $\mathbf{H}$ and the $\mathbf{w}_i^T$ with i such that ψ_i is not constant on $I_i^{\ell_i}$. Given the previous arguments, the sequence $(\mathbf{M}\mathbf{x}_k)_{k\in\mathbb{N}}$ is bounded. The situation is now the same as in the second part of the proof of Proposition 7.1 given in Chapter 8, and one can follow the same steps to conclude.

□

The present parameterization of R, is greatly inspired by [330]. However, instead of its single (non-decreasing) spline non-linearity used in [330], we decompose the activation $\varphi = \psi'$ into the difference of two splines as

$$\varphi = \mu\,\varphi_+ - \varphi_-, \tag{8.13}$$

where $\mu \in \mathbb{R}_{\geq 0}$ is a learnable parameter and the φ_+, φ_- are trainable, non-decreasing, non-expansive linear splines. Although theoretically equivalent to the use of a single linear spline φ with $s_{\text{inf}} \leq 1$, we found the decomposition (8.13) to be more effective for the training. Theoretical motivations for using splines in a constrained NN have been proposed in [211], and a discussion of the expressivity of the resulting NN architecture can be found in [333]. Our choice ensures that the following properties are met:

- R is 1-weakly convex, which follows from Proposition 8.2;
- the Lipschitz constant is bounded as

$$\text{Lip}(\boldsymbol{\nabla} R) \leq \|\mathbf{W}\|^2 \,\text{Lip}(\varphi) \leq \max(\mu, 1). \tag{8.14}$$

In the following, we provide more parameterization details regarding the parameterization.

Parameterization of Learnable Linear Splines Both linear splines φ_+ and φ_- are parameterized in the same way with our spline toolbox [236, 314]. In the sequel, we abbreviate their respective learnable parameters $\mathbf{c}_+$ and $\mathbf{c}_-$ by $\mathbf{c}$. We use $\varphi_{\mathbf{c}}\colon \mathbb{R} \to \mathbb{R}$ with knots $\tau_m = (m - M/2)\Delta$, $m = 0, \dots, M$, where Δ is the spacing. For simplicity,

[2] A nonconstant polynomial cannot have a finite limit at $\pm\infty$.

we assume that M is even. The learnable parameter $\mathbf{c} = (c_m)_{m=0}^M \in \mathbb{R}^{M+1}$ defines the values $\varphi_{\mathbf{c}}(\tau_m) = c_m$ of $\varphi_{\mathbf{c}}$ at the knots. To fully characterize $\varphi_{\mathbf{c}}$, we extend it by the constant value c_0 on $(-\infty, \tau_0]$ and c_M on $[\tau_M, +\infty)$. Consequently, any primitive ψ of $\varphi_{\mathbf{c}}$ is piecewise quadratic on $[\tau_0, \tau_M]$ with affine extensions.

Constraints on the Linear Splines To ensure that the φ_i are non-decreasing and non-expansive, we follow the strategy introduced in [330]. Let $\mathbf{D} \in \mathbb{R}^{M\times(M+1)}$ be the one-dimensional finite-difference matrix with $(\mathbf{Dc})_m = (c_{m+1} - c_m)$ for $m = 1, \dots, M$. As $\varphi_{\mathbf{c}}$ is piecewise-linear, it holds that

$$\varphi_{\mathbf{c}} \text{ is non-decreasing and non-expansive} \Leftrightarrow 0 \leq (\mathbf{Dc})_m \leq \Delta,\ m = 1, \dots, M. \tag{8.15}$$

To optimize over $\{\, \varphi_{\mathbf{c}} \colon 0 \leq (\mathbf{Dc})_m \leq \Delta,\ m = 1, \dots, M\}$, we reparameterize the linear splines as $\varphi_{\boldsymbol{P}_\uparrow(\mathbf{c})}$, where

$$\boldsymbol{P}_\uparrow(\mathbf{c}) = \mathbf{S}\mathrm{Clip}_{[0,\Delta]}(\mathbf{Dc}) + \mathbf{1}^T\mathbf{c} \tag{8.16}$$

is a nonlinear projection onto the feasible set (8.15). In (8.16), $\mathrm{Clip}_{[0,\Delta]}$ is the pointwise clipping operation with $\mathrm{Clip}_{[0,\Delta]}(t) = \min(\max(0, t), \Delta)$, and $\mathbf{S}$ denotes the cumulative-sum operation with $(\mathbf{Sd})_{m+1} = \sum_{k=1}^m d_k$ for $m = 0, \dots, M$ and any $\mathbf{d} \in \mathbb{R}^m$. In words, $\boldsymbol{P}_\uparrow$ clips the finite differences between entries in $\mathbf{c}$ that are either greater than Δ or negative and sets them to the closest admissible value, while it preserves the mean due to the additional term $\mathbf{1}^T\mathbf{c}$.

Further, we enforce that φ_+ and φ_- are odd, which is natural for imaging as it results in even potentials. To get this symmetry while still satisfying (8.15), we use the change of variable $\mathbf{c} \to \frac{1}{2}(\boldsymbol{P}_\uparrow(\mathbf{c}) - \texttt{reverse}(\boldsymbol{P}_\uparrow(\mathbf{c}))$, where `reverse` flips the order of the entries of $\mathbf{c}$. Hence, all constraints are embedded into the parameterization, and the parameter $\mathbf{c}$ that is learned remains unconstrained.

Parameterization of Convolutional Filters The learnable convolution layer $\mathbf{W}$ is required to be of unit norm. Hence, we parameterize $\mathbf{W}$ as $\mathbf{W} = \mathbf{U}/\|\mathbf{U}\|$, where $\mathbf{U}$ represents a convolutional layer with the same dimensions as $\mathbf{W}$. The computation of the spectral norm $\|\mathbf{U}\|$ will be described in Section 8.4.3. To efficiently explore a large field of view, see also [330], we decompose $\mathbf{U}$ into a composition of three zero-padded convolutions with kernels of size $(k_s \times k_s)$, k_s odd, and an increasing number of output channels. Similar to [293], the convolution kernels are constrained to have zero means. The equivalent (up to boundary effects) *single-convolution* layer would have a kernel of size $(K_s \times K_s)$ with $K_s = 3k_s - 2$.

8.4.2 Multi-Noise-Level Denoiser

So far, we only introduced a generic R that is not adapted to diverse noise levels. To obtain a denoiser for various noise levels σ, a common approach is to incorporate an adjustable parameter $\lambda_\sigma \in \mathbb{R}$ as in

$$\hat{\mathbf{x}} = \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + \lambda_\sigma R(\mathbf{x}). \tag{8.17}$$

In principle, this leads to a noise-level-dependent regularizer $R_\sigma = \lambda_\sigma R$, but this dependence on σ turns out to be too simple to ensure good performance across multiple noise levels. Another limitation in this setting is that R_σ is λ_σ-weakly convex. Hence, for $\lambda_\sigma > 1$, one is not guaranteed to remain within the CNC framework and, for $\lambda_\sigma < 1$, we might not exploit the full freedom given by CNC models. Therefore, we instead express the parameters α_i[3] introduced in Section 8.4.1 as functions of the noise level

$$\alpha_i(\sigma) = \mathrm{e}^{s_{\alpha_i}(\sigma)}/(\sigma+\epsilon), \tag{8.18}$$

where we set $\epsilon = 1\cdot 10^{-5}$ to prevent instabilities for small σ. Here, s_{α_i} is a learnable linear spline with underlying parameter $\mathbf{c}_\alpha^i$, which is parameterized similarly to $\varphi_\pm$ but without constraints. The exponential parameterization in (8.18) allows for efficiently exploring a large range at training and such a scheme is quite common in learning, e.g. in the popular TNRD framework [331]. The scaling by σ in (8.18) allows for normalizing the noise distribution before the activation and was found to be very helpful in practice. Ultimately, our noise-level-dependent profile functions $\psi_i(t,\sigma) = (1/\alpha_i(\sigma))^2\psi(\alpha_i(\sigma)t)$ satisfy

$$\frac{\partial^2\psi_i}{\partial t^2}(t,\sigma) = \mu\varphi_+'(\alpha_i(\sigma)t) - \varphi_-'(\alpha_i(\sigma)t) \in [-1,+\infty). \tag{8.19}$$

Remark 8.3. *The bound on the weak-convexity modulus given in Proposition 8.2 does not depend on the parameters α_i. Consequently, the addition of the α_i as learnable parameters does not compromise the weak-convexity guarantees on R.*

In the remainder of this chapter, $\boldsymbol{\theta}$ represents the aggregated set of learnable parameters (as detailed in Section 8.4.3) and we use the notation $R_{\boldsymbol{\theta}}$ whenever an explicit reference to the parameters is needed. Likewise, with a slight abuse of notation, we use $R_{\boldsymbol{\theta}(\sigma)}$ to denote the regularizer at noise level σ. This noise-dependent regularizer $R_{\boldsymbol{\theta}(\sigma)}$ then yields the proximal denoiser

$$D_{\boldsymbol{\theta}(\sigma)}(\mathbf{y}) = \operatorname{prox}_{R_{\boldsymbol{\theta}(\sigma)}}(\mathbf{y}) = \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + R_{\boldsymbol{\theta}(\sigma)}(\mathbf{x}). \tag{8.20}$$

In general, $D_{\boldsymbol{\theta}(\sigma)}$ does not have a closed-form expression but, due to the convexity and smoothness of the underlying objective, $D_{\boldsymbol{\theta}(\sigma)}(\mathbf{y})$ can be computed efficiently with gradient-

[3] In preliminary investigations, we also attempted the learning of the parameter μ as a function of the noise level but it did not improve performance. Hence, μ is chosen to be constant across noise levels.

based solvers. In practice, we use AGD [334] combined with the standard gradient-based restart technique introduced in [335]. The stepsize is chosen as $1/(1+\max(1,\mu))$, which ensures convergence to a global minimizer as a consequence of the Lipschitz bound (8.14).

8.4.3 Training Procedure

In this section, we detail how the parameters $\boldsymbol{\theta}$ are learned so that $D_{\boldsymbol{\theta}(\sigma)}(\mathbf{y})$ is a good Gaussian denoiser across multiple noise levels.

Training Problem

Let $\{\mathbf{x}^m\}_{m=1}^M$ be a set of clean images. Each image $\mathbf{x}^m$ is corrupted as $\mathbf{y}^m = \mathbf{x}^m + \sigma^m \mathbf{n}^m$ with Gaussian noise $\mathbf{n}^m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and a noise level $\sigma^m \sim \mathcal{U}[0, \sigma_{\max}]$. Then, we define the following multi-noise-level training problem

$$\hat{\boldsymbol{\theta}} \in \operatorname*{arg\,min}_{\boldsymbol{\theta}} \sum_{m=1}^{M} \mathbb{E}_{(\mathbf{n}^m, \sigma^m)} \Big(\| D_{\boldsymbol{\theta}(\sigma^m)}(\mathbf{y}^m) - \mathbf{x}^m \|_1 \Big). \tag{8.21}$$

Here, the ℓ_1 loss is chosen because it is known to be robust and well-performing for the training of CNNs [300, 336].

Optimization

For clarity, we briefly recall the various parameters contained in $\boldsymbol{\theta}$ before outlining the actual optimization procedure.

Profile-Related Parameters The linear splines φ_+ and φ_- are parameterized by $\mathbf{c}_+$ and $\mathbf{c}_-$ via the constrained coefficients $\tilde{\mathbf{c}}_\pm = \frac{1}{2}(\boldsymbol{P}_\uparrow(\mathbf{c}_\pm) - \texttt{reverse}(\boldsymbol{P}_\uparrow(\mathbf{c}_\pm)))$ so that they are odd, non-decreasing, and non-expansive. Together with $\mu > 0$, (8.13) then leads to the linear-spline activation function φ. Recall that its primitive defines the profile ψ. The parameters $\mathbf{c}_\alpha^i$ specify the linear-spline functions $\alpha_i(\sigma)$, which rescale the profile ψ across the channels in (8.10) and across the noise levels.

Spectral Normalization The convolution operation represented by $\mathbf{W}$ is parameterized as $\mathbf{W} = \mathbf{U}/\|\mathbf{U}\|$, $\mathbf{U}$ consisting in the composition of 3 zero-padded convolutions. Here, $\|\mathbf{U}\|$ is computed as follows.

- For an efficient optimization, it is necessary to embed the estimation of $\|\mathbf{U}\|$ into the forward pass. By assuming circular boundary conditions instead of zero-padding, we can consider $\mathbf{U}$ as a *single-convolution* layer. Then, $\mathbf{U}^T\mathbf{U}$ encodes a 2D convolution

from a one-channel input to a one-channel output. Hence, it can be represented by a kernel $\mathbf{K}_{\mathbf{U}^T\mathbf{U}} \in \mathbb{R}^{(2K_s-1)\times(2K_s-1)}$. In this setting, the spectrum of $\mathbf{U}^T\mathbf{U}$ can be computed using the 2D discrete Fourier transform (DFT) [210] as

$$\operatorname{spec}(\mathbf{U}^T\mathbf{U}) = \Big\{ |\operatorname{DFT}(\operatorname{Pad}_{\sqrt{d}}(K_{\mathbf{U}^T\mathbf{U}}))_{k_1k_2}| \colon 1 \leq k_1, k_2 \leq \sqrt{d} \Big\}, \tag{8.22}$$

where $\operatorname{Pad}_{\sqrt{d}}$ zero-pads $\mathbf{K}_{\mathbf{U}^T\mathbf{U}}$ into a $(\sqrt{d} \times \sqrt{d})$ image. We rely on (8.22) to estimate $\|\mathbf{U}\| \simeq \max(\operatorname{spec}(\mathbf{U}^T\mathbf{U}))$ during training since it is efficient to compute.

- Subsequently, when evaluating a trained WCRR-NN model, the true $\|\mathbf{U}\|$ is computed with high precision using the power method (1000 steps). This firm normalization guarantees the 1-weak convexity of the underlying R (up to numerical imprecision).

Implicit-Differentiation The learning of the proximal denoiser comes with the challenge that $D_{\boldsymbol{\theta}(\sigma)}$ depends implicitly on $\boldsymbol{\theta}$. As shown in the deep-equilibrium (DEQ) framework [308], it is possible to compute the Jacobian $J_{\boldsymbol{\theta}} D_{\boldsymbol{\theta}(\sigma)}$ of the denoiser with respect to the parameters via implicit differentiation. For this purpose, two steps are required.

- **Image Denoising.** First, given a noisy input $\mathbf{y}^m$, one needs to perform the forward pass, which consists in the computation of $\hat{\mathbf{x}} = D_{\boldsymbol{\theta}(\sigma^m)}(\mathbf{y}^m)$. The deployed AGD is run until the relative change of norm between consecutive iterates is lower than 10^{-4}.

- **Gradient Computation.** We use the DEQ implementation introduced in [308] and now briefly discuss the general concept within our setting. The differentiability of R implies that the denoised images satisfy

$$\hat{\mathbf{x}}(\boldsymbol{\theta}) - \mathbf{y} + \boldsymbol{\nabla}_{\mathbf{x}} R(\boldsymbol{\theta}, \hat{\mathbf{x}}(\boldsymbol{\theta})) = \mathbf{0}, \tag{8.23}$$

where the dependence on σ is dropped for clarity and the dependence on $\boldsymbol{\theta}$ is made explicit. The application of the implicit-function theorem for (8.23) leads to

$$(\mathbf{I} + \mathbf{H}_R(\boldsymbol{\theta}, \hat{\mathbf{x}}(\boldsymbol{\theta}))) J_{\boldsymbol{\theta}} \hat{\mathbf{x}}(\boldsymbol{\theta}) = J_{\boldsymbol{\theta}} (\boldsymbol{\nabla}_{\mathbf{x}} R)(\boldsymbol{\theta}, \hat{\mathbf{x}}(\boldsymbol{\theta})). \tag{8.24}$$

Hence, we evaluate the matrix-vector products with $(J_{\boldsymbol{\theta}} D_{\boldsymbol{\theta}}(\mathbf{y}))^T = (J_{\boldsymbol{\theta}} \hat{\mathbf{x}}(\boldsymbol{\theta}))^T$ (which are required for computing the gradients of (8.21) within the backpropagation algorithm) by solving a simple linear system. This is carried out with the Anderson routine given in [308]. While deriving $J_{\boldsymbol{\theta}}(\boldsymbol{\nabla}_{\mathbf{x}} R)$ is cumbersome and usually left to automatic differentiation, we use the explicit expression

$$\mathbf{H}_R(\hat{\mathbf{x}})\mathbf{u} = \mathbf{W}^T \Big(\boldsymbol{\varphi}'(\mathbf{W}\hat{\mathbf{x}}) \odot (\mathbf{W}\mathbf{u}) \Big), \tag{8.25}$$

where $\odot$ is the Hadamard product and the piecewise-constant function φ' is analytically derived from the B-spline representation of the linear spline φ. This yields the same results as automatic differentiation but was found to be more efficient.

Optimization The non-convex training problem in (8.21) is solved with the stochastic Adam optimizer [43], where we sample for each batch the $\mathbf{x}^m$, the corresponding noise-level σ^m, and the noise $\mathbf{n}^m$. Note that, within each batch, images are corrupted with different noise levels and, likewise, in different epochs, different noise levels can be applied to the same $\mathbf{x}^m$.

8.4.4 Training and Denoising Performance

The proposed weakly convex regularizer is learned over the Gaussian-denoising task described in Section 8.4.3, with $\sigma_{\max} = 30/255$. The same procedure as in [189] is used to form 238,400 grayscale patches[4] of size (40×40) from 400 images of the BSD500 data set [281], while 12 other images are kept for validation. In accordance with the ablation study reported in Tables 8.2 and 8.3, the three filters in $\mathbf{U}$ have kernels with $k_s = 5$ and 4, 8, and 60 output channels, respectively. The linear spline φ_i have $M + 1 = 101$ equally distant knots with $\Delta = 2 \cdot 10^{-3}$. We initially set $\mathbf{c}_+ = \mathbf{0}$ and $(\mathbf{c}_-)_m = \tau_m$, which was found to be important to help the training. Intuitively, this choice helps the regularizer to use weak convexity, which is only permitted through $\mathbf{c}_-$. The linear splines parameterizing $\alpha_i(\sigma)$ have 11 equally distant knots in the range $[0, \sigma_{\max}]$ and are initialized with the constant value 5. Our model is trained with the Adam optimizer for 6000 steps with batches of size 128, which takes less than 2 hours on a Tesla V100 GPU. The learning rates are initially set to $5 \cdot 10^{-2}$ for μ, $5 \cdot 10^{-3}$ for $\mathbf{U}$, and $\mathbf{c}_\alpha^i$, and to $5 \cdot 10^{-4}$ for $\mathbf{c}_+$ and $\mathbf{c}_-$. Then, they are decayed by 0.75 every 500 batches. For evaluation, the denoising (8.20) is performed with AGD and a tolerance of 10^{-4} for the relative change of norm between consecutive iterates. An example of convergence curves is provided in Figure 8.1.

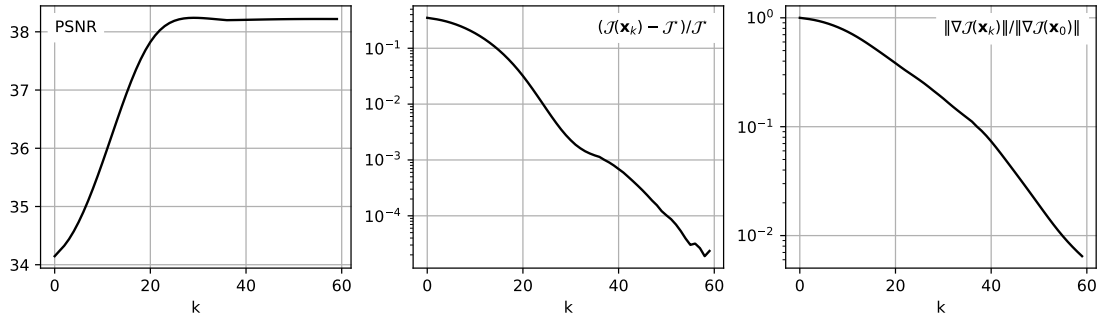

Figure 8.1: Example of convergence curves for denoising with the WCRR-NN and AGD.

[4]WCRR-NNs are fully convolutional and can process input of any spatial size.

Table 8.1: Denoising performance on the BSD68 test set.

		$\sigma = 5/255$	$\sigma = 15/255$	$\sigma = 25/255$
Convex	TV[1,2,3][22]	36.41	29.90	27.48
	Higher-order MRFs convex[1,2,3] [293]	-	30.45	28.04
	CRR-NN[1,2,3][330]	36.96	30.55	28.11
Provably CNC	TV CNC[1,2]	36.53	29.92	27.49
	WCRR-NN[1,2]	37.68	31.22	28.69
Approx. CNC	Prox-DRUNet	37.98	31.70	29.18
Others	Higher-order MRFs[1] [293]	-	31.22	28.70
	BM3D [332]	37.54	31.11	28.60

[1]Ridge-based regularizer, [2]Minimization of convex functional, [3]Convex regularizer

The numerical evaluation of our WCRR-NNs and several other methods on the BSD68 test set is provided in Table 8.1. The task is non-blind, in the sense that the noise level is used either directly as an input (as in BM3D) or indirectly via a regularization parameter that is tuned on a corresponding validation set (as in TV). The first important observation is that WCRR-NNs, which implement a convex energy, outperform the popular BM3D denoiser [332]. To the best of our knowledge, this is the first time a (learnable) convex model surpasses BM3D. A visual comparison of BM3D and (W)CRR-NNs is provided in Figure 8.2. The results obtained with the 2nd and 6th methods in Table 8.1 on the same image can be found in the original paper [293]. Next, we discuss in more depth the frameworks from Table 8.1 that are close in spirit to WCRR-NNs.

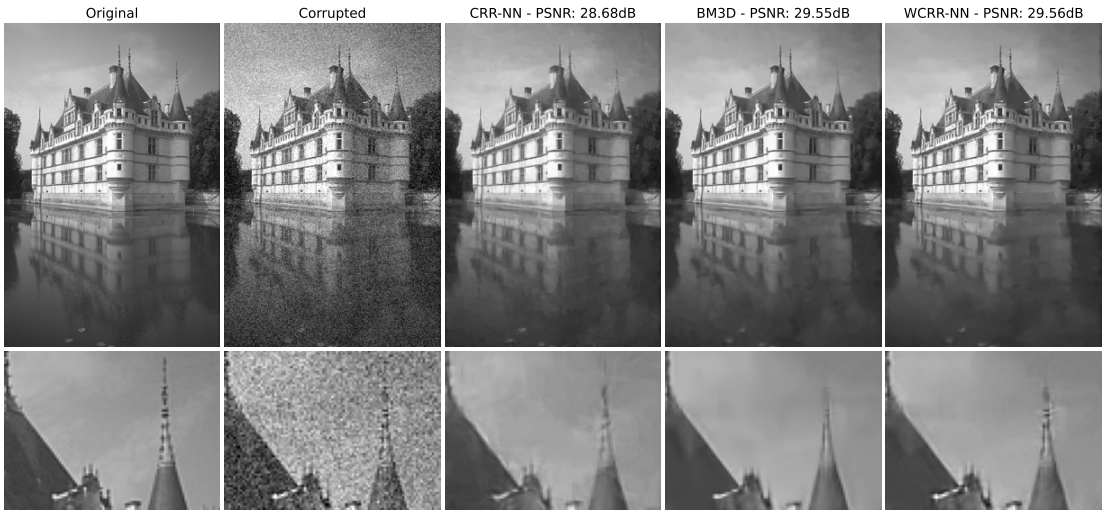

Figure 8.2: Denoising of the “castle" image from the BSD68 test set for noise level $\sigma = 25$.

Table 8.2: WCRR-NN: PSNR on BSD68 vs number of filters.

N_c	10	20	40	60	80
$\sigma = 5/255$	37.43	37.59	37.63	37.66	37.66
$\sigma = 15/255$	30.85	31.15	31.19	31.21	31.20
$\sigma = 25/255$	28.16	28.62	28.67	28.68	28.69

Table 8.3: WCRR-NN: PSNR on BSD68 vs kernel size.

k_s	3	5	7
K_s	5	13	19
$\sigma = 5/255$	37.64	37.66	37.65
$\sigma = 15/255$	31.14	31.21	31.21
$\sigma = 25/255$	28.56	28.68	28.68

CNC-Based Total Variation The WCRR-NN model is inspired by earlier works that extend TV denoising to the CNC framework using non-convex potential functions [326, 327]. The publicly available implementations outperform TV for specific classes of images, typically cartoon-like ones with sharp edges. However, we did not observe any significant improvements in the denoising of the natural images in BSD68. Hence, in our comparison, we used our version of CNC-TV, which was obtained by training a WCRR-NN with two fixed filters, namely, the horizontal and vertical finite differences. This corresponds to an anisotropic TV denoising with learned profiles—the CNC counterpart of the standard anisotropic TV denoising model. As reported in Table 8.1, this only yields marginal improvements over TV.

Field of Experts and Higher-Order MRFs The FoE approach corresponds to learning the filters associated with a regularizer of the form (8.10) with hand-picked profile functions [292]. It was successfully applied in [293], with both convex and non-convex profiles. A key difference with WCRNNs lies in the theoretical guarantees: The non-convex profiles are unconstrained in [292, 293]. Hence, the objective function is not provably convex. This means that the optimization is delicate and the convergence to a global optimum cannot be guaranteed. Interestingly, WCRR-NN offer the same performance as in [293] while minimizing a convex energy.

CRR-NNs This chapter extends upon the convex regularizers learned with CRR-NNs [330]. The substitution of weak convexity for convexity makes a significant difference as it yields a gain of at least 0.5dB for all noise levels (see Table 8.1 and Figure 8.3). In contrast with the simpler 2-filter TV setting, the improvement is substantial. This indicates that the learning of sufficiently many filters is necessary to fully exploit the additional freedom provided by weak convexity.

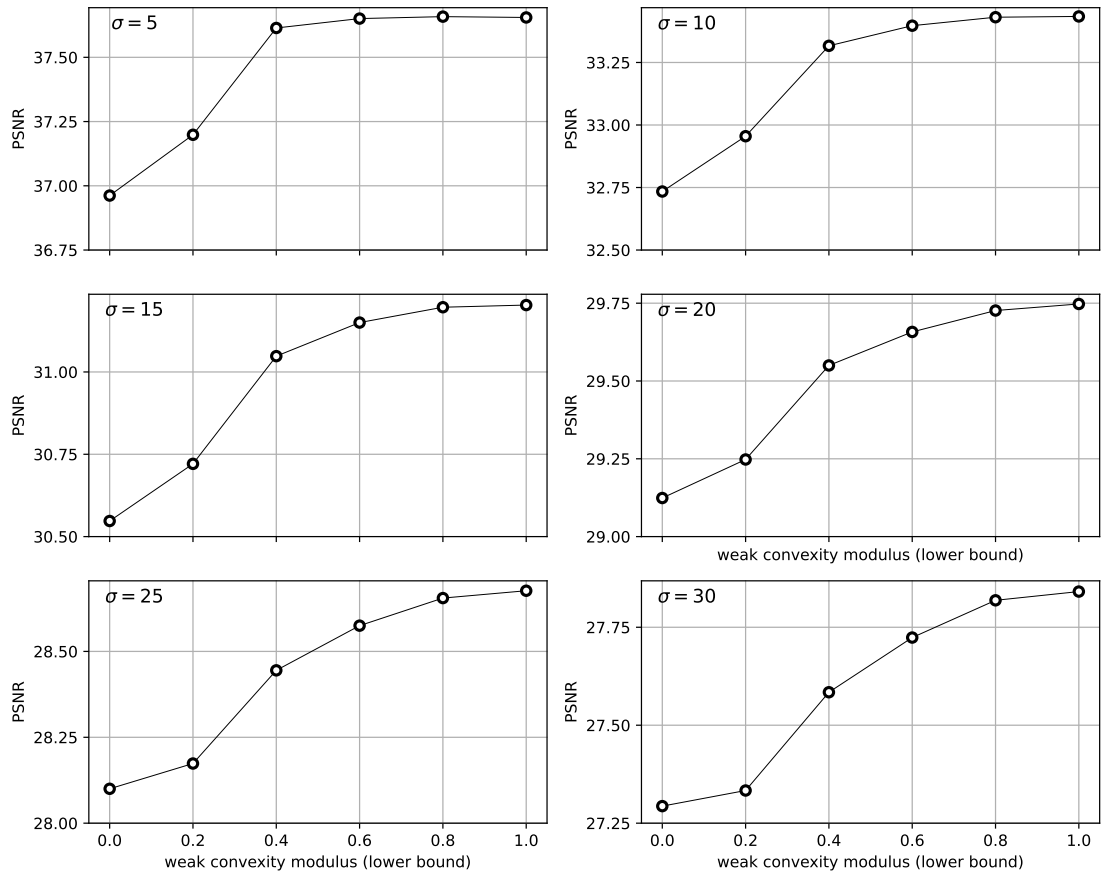

Figure 8.3: Denoising test performance of WCRR-NNs vs weak convexity bound s_{inf}. Note that for $s_{\text{inf}} = 0$ the regularizer is convex and we recover the performance of CRR-NNs.

Gradient-Step Denoisers Proposition 8.1 allows one to (implicitly) construct non-convex regularizers R by learning their proximal operator. This result is exploited in [76], where $\text{prox}_R = \boldsymbol{\nabla}\psi$ is parameterized through the potential $\psi = \frac{1}{2}\|\cdot\|^2 + g$, where $\boldsymbol{\nabla} g$ must be contractive. To leverage the power of deep-learning, the authors choose $g = \frac{1}{2}\|\cdot - \text{DRUNet}(\cdot)\|_2^2$, where DRUNet [337] is a deep CNN with ~ 17 million parameters. As there are currently no efficient methods to globally bound the Lipschitz constant of the gradient of a deep CNN, they propose to instead regularize the norm of the Jacobian of $\boldsymbol{\nabla} g$ at finitely many locations during training. This yields the Prox-DRUNet denoiser, which performs very well in practice (see Table 8.1[5]). Note that Prox-DRUNet only approximately satisfies the conditions to be a truly CNC method because $\|\mathbf{H}_g(\mathbf{x})\|$ can be greater than 1 for some $\mathbf{x}$, meaning that $\boldsymbol{\nabla} g$ is not contractive (as already reported

[5]The Prox-DRUNet denoiser given in [76] is trained on color images. For grayscale denoising, we plug the image into all three color channels and average the output across the channels, and tune the denoising strength parameter σ to optimize performance. As expected, the obtained metrics are on par with DnCNN for $\sigma = 25$ and with the gradient-step denoiser for $\sigma = 5$ in [102], which indicates the appropriateness of the usage.

in [76]). On noisy BSD68 images, we found[6] that $\|\mathbf{H}_g\|$ can be as large as 1.07 ($\sigma = 5$), 1.08 ($\sigma = 15$), 1.18 ($\sigma = 25$), and on a set of 68 random images (iid uniformly distributed pixels in $[0, 1]$) as large as 1.69 ($\sigma = 5$), 1.20 ($\sigma = 15$), 1.43 ($\sigma = 25$). Overall, we believe that WCRR-NNs and Prox-DRUNet offer a very complementary perspective. In fact, the good performance of Prox-DRUNet suggests that there could even be some room for further improvements with provably CNC methods.

8.4.5 Interpretation as Sparsity Prior

The filters and profile functions learned for our WCRR-NNs are shown in Figures 8.4 and 8.6, respectively.

Filters The impulse responses of the filters in $\mathbf{W}$ present patterns akin to wavelets and Gabor filters, in that they come in various modulations, orientations, and scales. In addition, the kernel $\mathbf{K}$ corresponding to the convolution $\mathbf{W}^T\mathbf{W}$ is very close to the 2D discrete Kronecker impulse, meaning that $\mathbf{W}$ is almost a Parseval frame ($\mathbf{W}^T\mathbf{W} \simeq \mathbf{I}$). A key difference, however, is that $\mathbf{K}$ is zero-mean. We also observed that more filters than in the convex setting of CRR-NNs are needed to reach the maximal performance. The payoff is that the filters are now able to capture more complicated patterns.

Profile Functions The learned profiles ψ_i are shared among the filters and then individually rescaled with the α_i so that the ψ_i have the same shape. Hence, only their prototype ψ is discussed here. The latter converges to a quasi-convex function (i.e., sub-level sets are intervals) even without us explicitly imposing this constraint. Moreover, ψ fully exploits the 1-weak convexity of the regularizer R in the sense that $\min_t \psi''(t) = -1$. Hence, this is an active constraint since R would not satisfy it by default. Overall, ψ closely resembles the minimax concave penalty function [325]. The influence of the weak-convexity parameter s_{inf} on the shape of the profile is shown in Figure 8.6.

To extend our model, we also experimented with learning a different ψ_i for each filter. This led to less interpretable profiles (not necessarily quasi-convex and with some oscillations), while it only offered a negligible gain in performance: less than 0.05dB on the denoising experiment for noise levels $\sigma \in \{5/255, 15/255, 25/255\}$.

Noise-Dependent Scaling The only part of R that depends on the noise level σ are the profiles ψ_i, which depend on σ through the $\alpha_i(\sigma)$. As can be seen in Figure 8.5, the $1/\alpha_i$ are on average linear functions of σ. Loosely speaking, most profiles will roughly have the form $\psi_i(t, \sigma) \simeq \sigma^2\psi(t/\sigma)$. To verify that such a simple dependence

[6] We computed $\|\mathbf{H}_g\|$ with a precise power method (300 iterations).

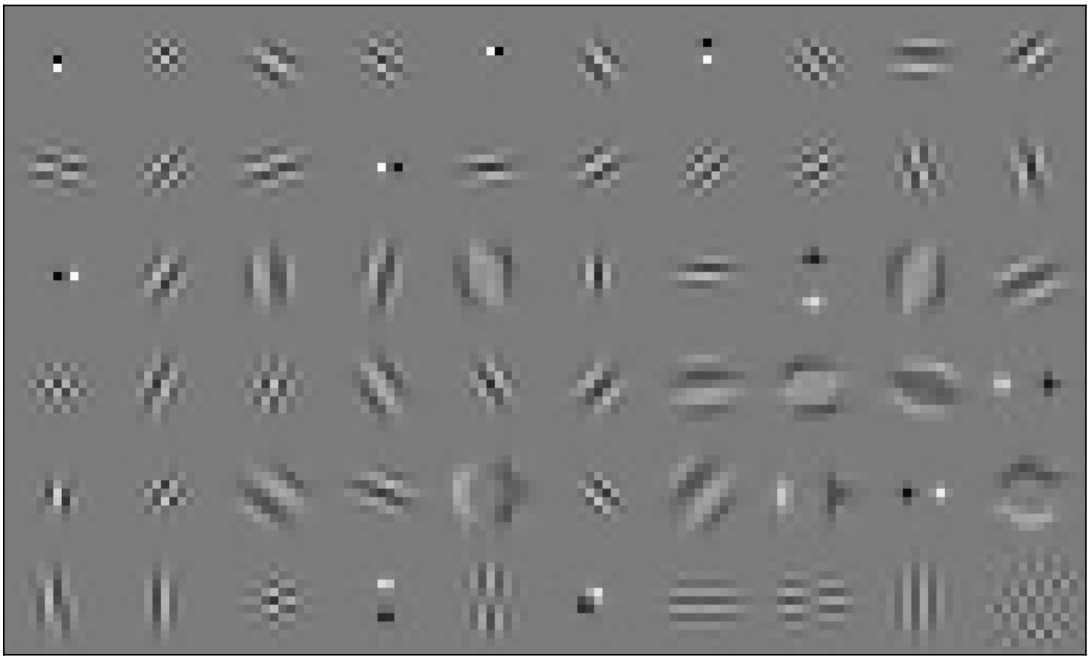

Figure 8.4: Impulse response of the filters in the learned WCRR-NN.

of R on σ is sufficient, 3 WCRR-NNs were trained to denoise at a single noise level ($\sigma \in \{5/255, 15/255, 25/255\}$). As these models do not outperform the multi-noise-level WCRR-NN on BSD68, the simple rescaling of the profiles appears to suffice.

Signal-Processing Perspective The regularizer R is trained to promote natural images. The corresponding gradient-descent step[7] $\mathbf{x} \mapsto \mathbf{x} - \boldsymbol{\nabla} R(\mathbf{x})/\mathrm{Lip}(\boldsymbol{\nabla} R) = \mathbf{x} - \mathbf{W}^T \boldsymbol{\varphi}(\mathbf{W}\mathbf{x})/\|\varphi'\|_\infty$, which should increase the regularity of images, is therefore expected to remove features considered as noise in natural images. In turn, we then expect that $\mathbf{x} \mapsto \mathbf{W}^T \boldsymbol{\varphi}(\mathbf{W}\mathbf{x})/\|\varphi'\|_\infty$ extracts some noise. Due to its shape, see Figure 8.6, the function $\varphi/\|\varphi'\|_\infty$ preserves the small responses $\mathbf{W}\mathbf{x}$ to the filters (it is almost the identity for small inputs), and cuts the large ones (it is almost the zero function for large inputs). Hence, one reconstructs the estimated noise $\mathbf{W}^T \boldsymbol{\varphi}(\mathbf{W}\mathbf{x})/\|\varphi'\|_\infty$ by essentially removing the components of $\mathbf{x}$ that exhibit a significant correlation with the kernels. This allows for a more efficient noise extraction than done by the monotonic clipping function learned in the convex regularization framework of CRR-NNs, see [330, Figs. 5 and 6]. While the monotonic clipping also preserves the small inputs, it is unable to fully remove the large responses because of the monotonicity constraint stemming from the convexity of the underlying potential.

In addition to the above perspective, we can make a link with wavelet- or framelet-like

[7] In our setting with no biases and where φ has a maximum slope at the origin, it can be shown that $\mathrm{Lip}(\mathbf{W}^T \boldsymbol{\varphi}(\mathbf{W}\cdot)) = \|\varphi'\|_\infty$.

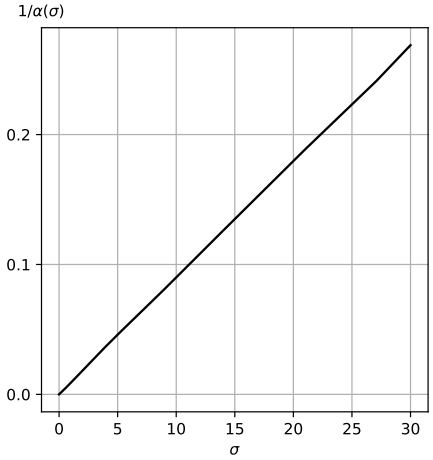

Figure 8.5: Plot of $1/\alpha(\sigma)$ vs σ, where $\alpha(\sigma) = \sum_{i=1}^{N_C} \alpha_i(\sigma)$ encodes the average behavior across the channels.

denoising [338–342]. Indeed, given that $\mathbf{W}^T\mathbf{W} \simeq \mathbf{I}$, the gradient-descent step can be approximated as $\mathbf{x} \mapsto \mathbf{x} - \mathbf{W}^T \boldsymbol{\varphi}(\mathbf{W}\mathbf{x})/\|\varphi'\|_\infty \simeq \mathbf{W}^T\phi(\mathbf{W}\mathbf{x})$ with $\phi = \mathrm{Id} - \varphi/\|\varphi'\|_\infty$. Since ϕ is zero around the origin and is the identity for sufficiently large inputs, it qualitatively stands between the soft- and hard-thresholding functions that have been key components for wavelet and framelet denoising for years. Finally, note that framelet-denoising models are themselves closely related to proximal operators [249].

8.5 Extension to Generic Inverse Problems

We now use the regularizer $R_{\boldsymbol{\theta}(\sigma)}$ trained in Section 8.4 to solve inverse problems based on the variational formulation (8.2). Here, the key challenge is the possible non-convexity of the objective, which prevents us from minimizing (8.2) globally. It is, however, possible to search for critical points. These are still of particular interest, especially because the regularizer has a simple structure with an *almost* convex energy landscape.

Proximal vs Gradient Methods The standard PnP frameworks rely on proximal-based methods with an explicit denoising step. The motivation there is that the denoising step is typically efficient to perform while neither the regularizer (if it exists) is explicitly known, nor its gradient. In our setting, on the contrary, it is very efficient to evaluate the regularizer and its gradient, and hence AGD methods [334], which are applicable to general non-convex problems [343], are better suited. In our setting, AGD is also known to attain optimal convergence rates among first-order methods. In the sequel, we recall the main features of AGD and show how to leverage the knowledge of the weak-convexity modulus of the objective.

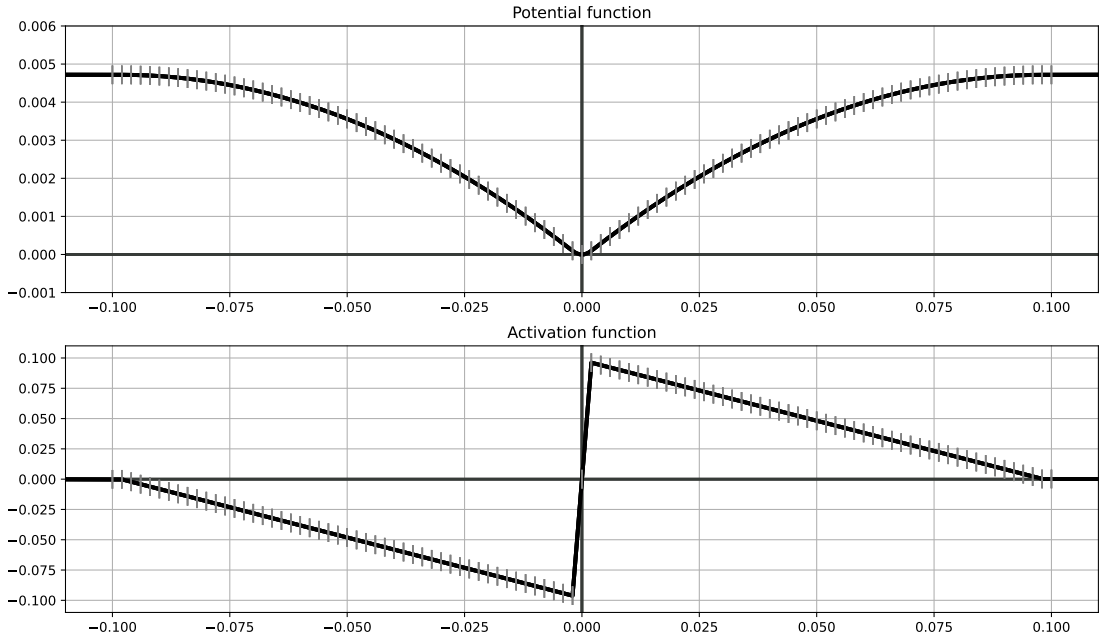

Figure 8.6: Potential function ψ and activation function $\varphi = \psi'$ of the learned WCRR-NN. These functions are splines of degrees 2 and 1, respectively. The vertical markers indicate the control points of the splines.

8.5.1 Accelerated Gradient Descent

To solve the inverse problem, we minimize the regularized objective

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda R_{\boldsymbol{\theta}(\sigma)}(\mathbf{x}), \tag{8.26}$$

where $R_{\boldsymbol{\theta}(\sigma)}$ is the 1-weakly convex regularizer from Section 8.4.4 and $\lambda > 0$ is a regularization parameter. Since the objective is differentiable, we can rely on gradient-based methods to find critical points of $\mathcal{J}$ as a convenient alternative to proximal algorithms. To reduce the reconstruction time, we propose an AGD variant in Algorithm 2, which is tailored to λ-weakly convex functionals $\mathcal{J}$ with L-Lipschitz-continuous gradient.

From (8.14), we infer that $\boldsymbol{\nabla}\mathcal{J}$ is L-Lipschitz-continuous with $L \leq \|\mathbf{H}\|^2 + \lambda \max(\mu, 1)$, which implies for $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ the standard upper estimate

$$\mathcal{J}(\mathbf{x}_1) \leq \mathcal{J}(\mathbf{x}_2) + \boldsymbol{\nabla}\mathcal{J}(\mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2) + \frac{L}{2}\|\mathbf{x}_1 - \mathbf{x}_2\|^2. \tag{8.27}$$

As $R_{\boldsymbol{\theta}(\sigma)}$ is 1-weakly convex, $\mathcal{J}$ is λ-weakly convex. Hence, the subgradient inequality for convex functions leads for $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ to the lower estimate

$$\mathcal{J}(\mathbf{x}_1) \geq \mathcal{J}(\mathbf{x}_2) + \boldsymbol{\nabla}\mathcal{J}(\mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2) - \frac{\lambda}{2}\|\mathbf{x}_1 - \mathbf{x}_2\|^2. \tag{8.28}$$

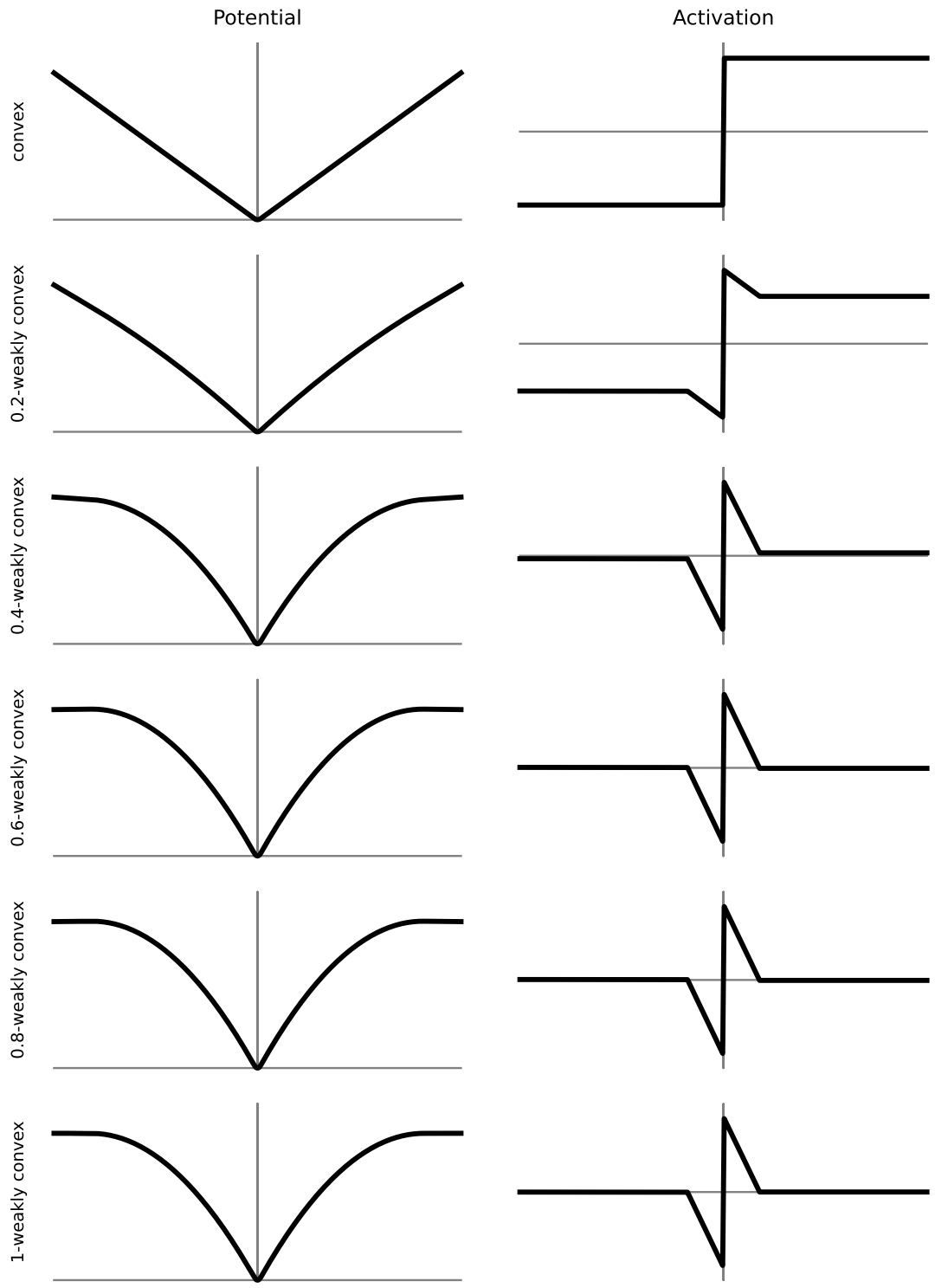

Figure 8.7: Potential functions ψ learned and the corresponding activation functions $\varphi = \psi'$ of the WCRR-NN for different weak-convexity modulus.

Algorithm 2 Safeguarded AGD for λ-weakly convex $\mathcal{J}$ with L-Lipschitz gradient

Input: initialization $x_0 \in \mathbb{R}^d$, tolerance $\epsilon > 0$, $a > 1$
Set $t_0 = t_1 = 1$, $k = 1$, $\mathbf{z}_0 = \mathbf{x}_0$, $\mathbf{x}_1 = \mathbf{x}_0$
while $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|/\|\mathbf{x}_{k-1}\| > \epsilon$ or $k = 1$ **do**
  $\mathbf{z}_k = \mathbf{x}_k + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1})$
  $\text{crit} = \boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k)^T(\mathbf{z}_k - \mathbf{z}_{k-1}) + \frac{a\lambda}{2}\|\mathbf{z}_k - \mathbf{z}_{k-1}\|^2$
  if crit > 0 **then**
    $\mathbf{z}_k = \mathbf{x}_k$
    $t_k = 1$
  $\mathbf{x}_{k+1} = \mathbf{z}_k - \frac{1}{L}\boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k)$
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
  $k \leftarrow k + 1$
Output: Approximate solution $\mathbf{x}_k$

Given some initilization $\mathbf{x}_0 = \mathbf{x}_{-1} = \mathbf{z}_0 \in \mathbb{R}^d$ and a sequence of Nesterov momentum parameters $\{\beta_k\}_{k\in\mathbb{N}} \subset [0, 1]$, the standard AGD [334] update steps read

$$\mathbf{z}_k = \mathbf{x}_k + \beta_k(\mathbf{x}_k - \mathbf{x}_{k-1}), \tag{8.29}$$

$$\mathbf{x}_{k+1} = \mathbf{z}_k - \frac{1}{L}\boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k). \tag{8.30}$$

The combination of (8.30) and (8.27) yields the decrease estimate

$$\mathcal{J}(\mathbf{x}_{k+1}) - \mathcal{J}(\mathbf{z}_k) \leq -\frac{L}{2}\|\mathbf{x}_{k+1} - \mathbf{z}_k\|^2. \tag{8.31}$$

However, the update (8.29) does not necessarily guarantee the decrease of $\{\mathcal{J}(\mathbf{z}_k)\}_{k\in\mathbb{N}}$. Hence, for a predefined $a > 1$, we propose to check the condition

$$\boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k)^T(\mathbf{z}_k - \mathbf{z}_{k-1}) + \frac{a\lambda}{2}\|\mathbf{z}_k - \mathbf{z}_{k-1}\|^2 \leq 0 \tag{8.32}$$

after having tentatively performed (8.29). If (8.32) is violated, we perform the plain gradient update $\mathbf{z}_k = \mathbf{x}_k$ (instead of (8.29)) and apply a restart technique, as proposed in [335]. In this case, we get from (8.31) at the previous step that

$$\mathcal{J}(\mathbf{z}_k) - \mathcal{J}(\mathbf{z}_{k-1}) \leq -\frac{L}{2}\|\mathbf{z}_k - \mathbf{z}_{k-1}\|^2. \tag{8.33}$$

Otherwise, the incorporation of (8.28) implies that

$$\mathcal{J}(\mathbf{z}_k) - \mathcal{J}(\mathbf{z}_{k-1}) \leq \boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k)^T(\mathbf{z}_k - \mathbf{z}_{k-1}) + \frac{\lambda}{2}\|\mathbf{z}_k - \mathbf{z}_{k-1}\|^2 \leq -\frac{(a-1)\lambda}{2}\|\mathbf{z}_k - \mathbf{z}_{k-1}\|^2. \tag{8.34}$$

To sum up, the acceleration steps are kept only if they lead to a sufficient decrease of the objective. Otherwise, the plain gradient-descent step guarantees this decrease.

Remark 8.4. *The gradient-based condition* (8.32) *is more restrictive than the objective-based condition* (8.33)*. However,* (8.32) *is computationally cheaper to verify as it only involves inner products of already computed quantities. In practice, we observed that* (8.32) *is rarely violated.*

Remark 8.5. *The parameter a in* (8.32) *must be greater than the weak-convexity modulus of $R_{\boldsymbol{\theta}}$ to ensure convergence. At this point, a precise estimate of this modulus—which we know to be bounded by one in our setting—weakens the condition* (8.32) *and typically yields faster convergence. On the contrary, the reliance on a loose bound leads to frequent restarts, at the detriment of acceleration.*

Regarding Algorithm 2, we now derive a convergence result in Theorem 8.1 using [344, Theorem 3.7], which itself extends the seminal work [96] to the inertia setting. Note that the objective (8.26) is semi-algebraic since the profile function ψ is piecewise-polynomial. Hence, it satisfies the required (quite technical) KL property, see also [96].

Theorem 8.1. *Assume that $\mathcal{J}$ satisfies the KL property and is bounded from below. If the sequence $(\mathbf{z}_k)_{k\in\mathbb{N}}$ generated by Algorithm 2 (without the stopping criterion) is bounded, then it converges to a critical point $\hat{\mathbf{z}}$ of $\mathcal{J}$. Moreover, the sequence $(\mathbf{z}_k)_{k\in\mathbb{N}}$ has finite length, in the sense that*

$$\sum_k \|\mathbf{z}_{k+1} - \mathbf{z}_k\| < \infty. \tag{8.35}$$

Proof. According to [344, Theorem 3.7], we need to check that

(H1) there exists $a > 0$ such that $\mathcal{J}(\mathbf{z}_k) + a\|\mathbf{z}_k - \mathbf{z}_{k-1}\|^2 \leq \mathcal{J}(\mathbf{z}_{k-1})$ for all $k \in \mathbb{N}$;

(H2) there exists $b > 0$ such that $\|\boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k)\| \leq 2b(\|\mathbf{z}_k - \mathbf{z}_{k-1}\| + \|\mathbf{z}_{k+1} - \mathbf{z}_k\|)$ for all $k \in \mathbb{N}$;

(H3) there exists a subsequence $(\mathbf{z}_{k_j})_{j\in\mathbb{N}}$ such that $\mathbf{z}_{k_j} \to \mathbf{z}$ and $\mathcal{J}(\mathbf{z}_{k_j}) \to \mathcal{J}(\mathbf{z})$.

These three conditions are needed in order to conclude that the iterates $(\mathbf{z}_k)_{k\in\mathbb{N}}$ satisfy (8.35) and converge to a critical point $\hat{\mathbf{z}}$ of $\mathcal{J}$. We have already verified (H1) in (8.33) and (8.34). For (H2), we first note that, if (8.32) is violated, then it directly holds that

$$\|\boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k)\| = L\|\mathbf{z}_{k+1} - \mathbf{z}_k\|. \tag{8.36}$$

Otherwise, $\|\boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k)\| = L\|\mathbf{x}_{k+1} - \mathbf{z}_k\|$, and it follows that

$$\begin{aligned}\|\boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k)\| &\leq L\|\mathbf{z}_{k+1} - \mathbf{z}_k\| + \beta_k L\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \\ &\leq L\|\mathbf{z}_{k+1} - \mathbf{z}_k\| + L\left\|\mathbf{z}_k - \frac{1}{L}\boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_k) - \mathbf{z}_{k-1} + \frac{1}{L}\boldsymbol{\nabla}\mathcal{J}(\mathbf{z}_{k-1})\right\| \\ &\leq L\|\mathbf{z}_{k+1} - \mathbf{z}_k\| + 2L\|\mathbf{z}_k - \mathbf{z}_{k-1}\|. \end{aligned} \tag{8.37}$$

Since we assume that the sequence $(\mathbf{z}_k)_{k\in\mathbb{N}}$ is bounded and since $\mathcal{J}$ is continuous, also (H3) holds and the result follows from [344, Theorem 3.7]. □

Remark 8.6. *To ensure that $(\mathbf{z}_k)_{k\in\mathbb{N}}$ remains bounded, one can simply add a regularization term $\kappa\|\mathbf{x}\|^2$ to $\mathcal{J}$, where $\kappa > 0$ can be arbitrarily small. Then, $\mathcal{J}$ becomes coercive because the profile ψ has linear extensions (see Lemma 8.2). Therefore, $(\mathbf{z}_k)_{k\in\mathbb{N}}$ must remain bounded, otherwise $(\mathcal{J}(\mathbf{z}_k))_{k\in\mathbb{N}}$ could not be decreasing (see* (8.33)*). Empirically, however, this "trick" was found to be unnecessary as the iterates would remain bounded in all settings explored.*

Lemma 8.2. *Let R be a ridge regularizer of the form* (7.4)*, where the profiles ψ_j are continuous, even, and have affine extensions*[8]*. Then*

$$\mathbf{x} \mapsto \|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + R(\mathbf{x}) + \kappa\|\mathbf{x}\|_2^2 \tag{8.38}$$

is coercive for any $\kappa > 0$.

Proof. By assumption all ψ_j are affine on $[t_0, +\infty)$ with slope $u_j \in \mathbb{R}$. Hence, it holds for $|t| > t_0$ that $\psi_j(t) = \psi_j(|t|) = u_j(|t| - t_0) + \psi_j(t_0)$. Next, we define $v_j = \min_{|t|\leq t_0}(\psi_j(t) - u_j(|t| - t_0)) \leq \psi_j(t_0)$, which is well-defined since ψ_j are continuous. By definition of v_j, it holds for any $t \in \mathbb{R}$ that $\psi_j(t) \geq u_j(|t| - t_0) + v_j$, and the objective in (8.38) is lower bounded by

$$\mathbf{x} \mapsto \|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + \kappa\|\mathbf{x}\|_2^2 + \sum_{j=1}^{d\times N_C} u_j(|\mathbf{w}_j^T\mathbf{x}| - t_0) + v_j, \tag{8.39}$$

which is coercive for any $\kappa > 0$.

□

Remark 8.7. *For $\sqrt{\lambda_{\min}(\mathbf{H}^T\mathbf{H})} \geq \lambda$, the objective* (8.26) *is $(\sqrt{\lambda_{\min}(\mathbf{H}^T\mathbf{H})} - \lambda)$-strongly convex and, hence, convex. Then, Algorithm 2 is guaranteed to converge to a global minimum of the objective. Otherwise, some results on convergence to local minima come into play, including with rates[96, 345].*

As the problem is potentially non-convex, the initialization of the algorithm may influence the final reconstruction. However, we did not observe such a dependence in our

[8] In the sense that there exists $t_0 \in \mathbb{R}$ such that ψ_j is affine on $(-\infty, -t_0]$ and on $[t_0, +\infty)$.

	Number of Parameters
TV	1
ACR [286]	$6 \cdot 10^5$
PnP-βCNN [314]	$1 \cdot 10^5$
CRR-NN [330]	$1 \cdot 10^4$
AR [295]	$2 \cdot 10^7$
WCRR-NN	$1 \cdot 10^4$
PnP-DnCNN [81]	$6 \cdot 10^5$
Prox-DRUNet [76]	$2 \cdot 10^7$

experimental settings. Therefore, we opted for a zero-initialization or when applicable used a fast-to-compute initial guess, e.g. FBP for CT. A more sophisticated strategy may take as initial configuration the reconstruction of a trustworthy convex variational model such as [330]. Then, Algorithm 2 would be used to refine the reconstruction.

8.5.2 Experiments

The WCRR-NN model trained in Section 8.4 to perform denoising on the BSD500 dataset is now deployed to solve two image-reconstruction problems using safeguarded AGD. For each setup, λ and σ are tuned over a validation set to maximize the peak signal-to-noise ratio (PSNR) with the coarse-to-fine routine from [330], and then used for evaluation.

MRI (I) The ground truth consists of fully sampled knee images with size (320×320) from the fastMRI dataset [282]. The corresponding MRI measurements are a subsampled version of the 2D Fourier transforms (k-space). This subsampling is performed with a Cartesian mask that has two parameters: the acceleration $M_{\text{acc}} = 4$ and the center fraction $M_{\text{cf}} = 0.08$. All the $\lfloor 320 M_{\text{cf}} \rfloor$ columns in the center of the k-space (low frequencies) are retained in full, while columns in the other region of the k-space are uniformly sampled. More precisely, we are left with $\lfloor 320/M_{\text{acc}} \rfloor$ selected columns. Lastly, both the real and imaginary parts of the measurements are corrupted by Gaussian noise with standard deviation $\sigma_{\mathbf{n}} = 10^{-4}$. For validation and testing, we picked 10 and 99 images, respectively, all normalized to have a maximum value of one.

MRI (II) The first MRI experiment just described was replicated from our preprint [346]. For completeness, we also provide in this thesis results for the other MRI experiments explored in Chapter 6 and 7. These experiments include single- and multi-coil MRI, and we refer the reader to Chapter 6 for the experimental details.

Table 8.4: PSNR and SSIM values for MRI (I) and CT reconstruction experiments.

(a) MRI (I)

Metric	PSNR	SSIM
Zero-fill	27.92	0.711
TV[28]	32.03	0.7922
CRR-NN [330]	33.14	0.842
WCRR-NN	<u>34.55</u>	<u>0.858</u>
Prox-DRUNet [76]	**35.09**	**0.864**

(b) CT

Metric	PSNR	SSIM
TV	31.57	0.852
PnP-βCNN (ReLU)	30.34	0.782
PnP-βCNN (LLS)	31.85	0.844
ACR [286]	32.17	0.868
CRR-NN	32.87	0.862
AR [295]	33.62	0.875
WCRR-NN	<u>34.06</u>	<u>0.895</u>
PnP-DnCNN [81]	33.83	0.881
Prox-DRUNet	**34.20**	**0.901**

Table 8.5: Single-coil MRI (II).

	2-fold				4-fold			
	PSNR		SSIM		PSNR		SSIM	
	PD	PDFS	PD	PDFS	PD	PDFS	PD	PDFS
Zero-fill	33.32	34.49	0.871	0.872	27.40	29.68	0.729	0.745
TV	39.22	37.73	0.947	0.917	32.44	32.67	0.833	0.781
PnP-βCNN (ReLU)	38.15	37.41	0.938	0.918	30.62	31.45	0.818	0.786
PnP-βCNN (LLS)	40.06	38.63	0.955	0.931	32.81	33.04	0.859	0.817
CRR-NN	40.95	38.91	0.961	0.934	33.99	33.75	0.880	0.831
PnP-DnCNN [81]	40.52	39.02	0.956	<u>0.935</u>	35.24	<u>34.63</u>	0.884	<u>0.840</u>
WCRR-NN	<u>41.71</u>	<u>39.10</u>	<u>0.966</u>	<u>0.935</u>	<u>35.76</u>	34.62	<u>0.898</u>	0.838
Prox-DRUNet [76]	**41.85**	**39.12**	**0.967**	**0.937**	**36.20**	**35.05**	**0.901**	**0.847**

Table 8.6: Multi-coil MRI (II).

	4-fold				8-fold			
	PSNR		SSIM		PSNR		SSIM	
	PD	PDFS	PD	PDFS	PD	PDFS	PD	PDFS
$\mathbf{H}^T\mathbf{y}$	27.71	29.94	0.751	0.759	23.80	27.19	0.648	0.681
TV	38.06	37.31	0.935	0.914	32.77	33.38	0.850	0.824
PnP-βCNN (ReLU)	37.21	37.06	0.929	0.915	31.37	32.57	0.837	0.822
PnP-βCNN (LLS)	38.68	37.96	0.943	0.924	32.75	33.61	0.859	0.835
CRR-NN	39.54	38.29	0.950	0.927	34.29	34.50	0.881	0.852
PnP-DnCNN [81]	39.55	**38.52**	0.947	**0.929**	35.11	<u>35.14</u>	0.881	**0.858**
WCRR-NN	<u>40.11</u>	38.43	<u>0.953</u>	0.926	<u>35.55</u>	**35.15**	<u>0.893</u>	0.856
Prox-DRUNet [76]	**40.26**	<u>38.47</u>	**0.955**	**0.929**	**35.78**	35.12	**0.894**	<u>0.857</u>

CT To provide a comparison with adversarial regularization (AR) [295] and its convex counterpart ACR [286] (see more details in Section 8.5.2), we include the sparse-view CT experiment proposed in [286]. Its data consist of human abdominal CT scans for 10 patients, publicly available as part of the low-dose CT Grand Challenge [285]. For validation, 6 images are taken uniformly from the first patient of the training set used by [286]. To benchmark all methods, we use the same set as [286], made of 128 slices with size (512×512) from a single patient, all normalized to have a maximum value of one. The CT measurements are simulated using a parallel-beam acquisition geometry with 200 angles and 400 detectors. These measurements are corrupted by Gaussian noise with standard deviation $\sigma_{\mathbf{n}} = 2.0$.

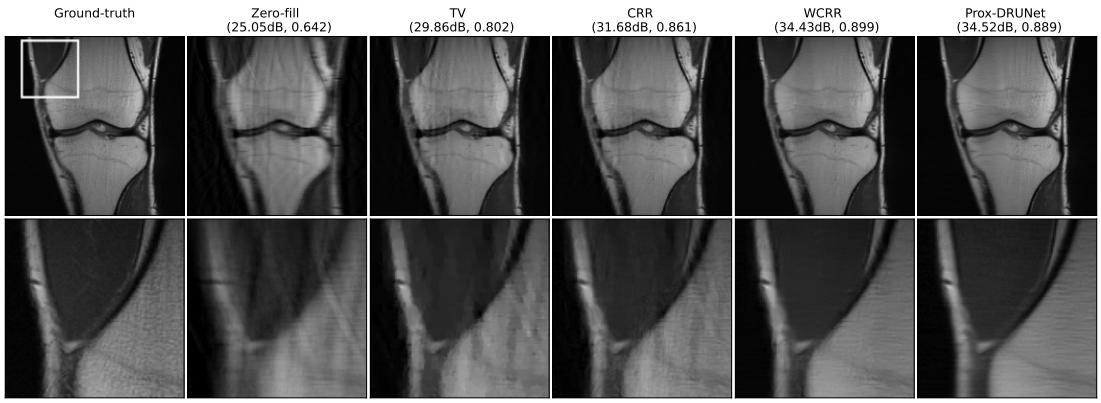

Figure 8.8: Reconstructed images for the MRI (I) experiment. The reported metrics are PSNR and SSIM.

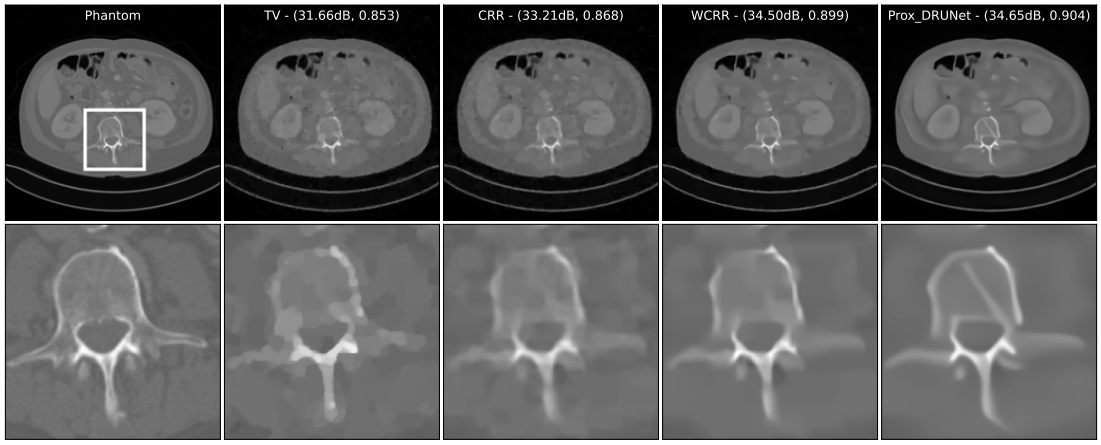

Figure 8.9: Reconstructions for the sparse-view CT experiment. The reported metrics are PSNR and SSIM.

Comparison and Discussion

The PSNR and structural similarity index measure (SSIM) values on the test sets are reported together with the parameter numbers in Table 8.4. The hyperparameters of each method are tuned to maximize the average PSNR over the validation sets with the coarse-to-fine method described in [330]. We observe that WCRR-NNs outperform the other energy-based methods and are close to the PnP approach. For both problems, reconstructions are provided in Figures 8.8 and 8.9, and examples of convergence curves for SAGD are given in Figures 8.10 and 8.11.

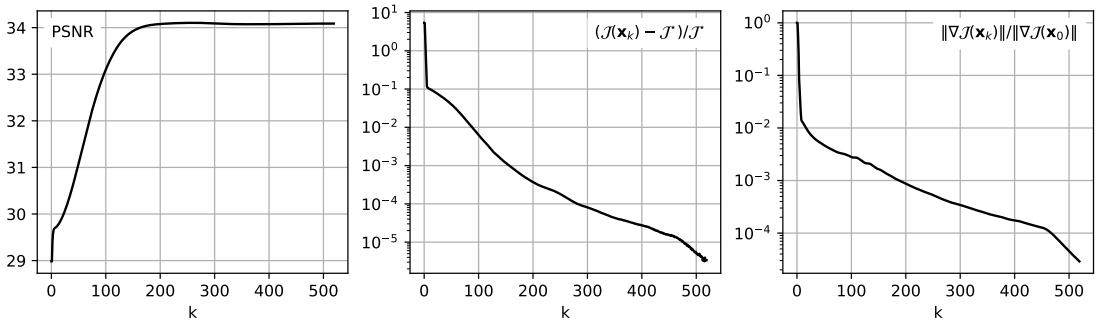

Figure 8.10: Example of convergence curves (MRI).

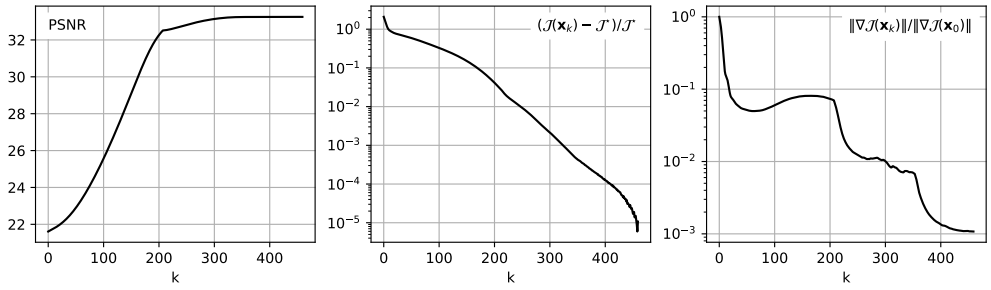

Figure 8.11: Example of convergence curves (CT).

Overall, the results illustrate the universality and efficiency of our method. In the following, we briefly comment on the competing methods used in our evaluation.

Convex Models The TV and CRR-NN reconstructions serve as references for convex methods. They are computed via the FISTA algorithm [28] with a nonnegativity constraint. Similar to denoising, we observe that the move from convex to weakly-convex regularization leads to significant improvements in quality. In the MRI experiment, the aliasing artifacts introduced by CRR-NN, and even more by TV, are suppressed by the weakly convex regularizer. In the CT experiment, the TV reconstruction includes staircasing artifacts, and CRR-NN has a slight tendency to blur the edges. On the contrary, our weakly convex regularizer is able to produce sharp edges without blur, but sometimes at the

cost of over-smoothing some background details. Note that convex models are still better understood from a theoretical perspective because convergence to global optima can be guaranteed. Hence, they might be favorable in certain settings.

Adversarial Regularization As references for explicit regularization approaches, we provide a comparison with the convex ACR [286, 302] framework and its non-convex counterpart AR [295]. Bypassing a gradient-based parameterization, these models parameterize the regularizer R directly and train it in an adversarial manner. As the regularizers of [286, 295, 302] are tailored to a specific inverse problem, we can only provide a comparison for their CT experiment. Even though ACR and AR have significantly more parameters than (W)CRR-NN, they perform less well. The numerical results present favorable evidence regarding the effectiveness of the parameterization used for WCRR-NNs. Note, however, that drawing a definitive conclusion on the parameterization only is delicate since AR and ACR rely on a different training procedure.

Plug-and-Play Our approach bears some resemblance with PnP methods since R is learned on a generic denoising task. Hence, it is natural to compare WCRR-NNs with a deep CNN version of this approach. Among countless variations, the recently proposed framework [76], which we refer to as Prox-DRUNet, is the closest to ours in terms of theoretical guarantees and existence of an underlying regularizer (see Section 8.4.4 for a discussion). We use the pre-trained DRUNet-based proximal denoiser from [76] within the PnP-PGD (proximal gradient descent) for CT and the PnP-DRS (Douglas-Rachford splitting) for MRI[9]. This approach, which is one of the state-of-the-art in energy-related PnP, yields slightly better PSNR and SSIM than our method. Note that Prox-DRUNet involves 3 orders of magnitude more parameters and days of training.

In the MRI experiment, both Prox-DRUNet and WCRR-NN can avoid the aliasing artifacts typically generated by methods that rely on a convex regularizer. In the CT experiment, the visual inspection of the reconstructions reveals that quality metrics are only part of the story. While the output of Prox-DRUNet always looked remarkably realistic, it was more prone to hallucination/artifact exaggeration, especially for hard problems such as the CT experiment. In that respect, the Prox-DRUNet reconstruction in Figure 8.9 is particularly telling: It includes an elongated structure that is not present in the ground truth, nor in any other reconstruction. While such *enhanced* images are desirable in many settings and lead to state-of-the-art denoising performance, they raise major concerns for sensitive applications, including medical imaging. Regarding the theoretical convergence guarantees of PnP-PGD, the necessary Lipschitz constraint is only enforced by regularization during training. Unfortunately, it is infeasible to verify if it is met after training. In practice indeed, it seems to be not fully met [76].

[9] PnP-DRS is well-suited to settings where the proximal operator of the data term can be efficiently computed, which includes MRI but not CT.

8.6 Conclusion

In this chapter, we proposed a method for the learning of a 1-weakly convex regularizer that leads to a convex denoising functional. To the best of our knowledge, this is the first instance of convex non-convex schemes that surpasses BM3D for the denoising of natural images. A key feature of our method is that the architecture deployed to parameterize the regularizer is shallow. Thereby, the role of each parameter is transparent: Parameters are adjusted to produce a sparsity-promoting prior. Although the regularization of inverse problems with the learned regularizer does not necessarily lead to a convex objective, gradient-based optimization methods are empirically effective and produce high-quality reconstructions. In the future, a better understanding of WCRR-NNs might help to boost the performance of lightweight and robust data-driven image-reconstruction models even further. This includes the dependence of the learned regularizer on the modality and/or on the image domain used during training. It is indeed expected, for instance, that a fine-tuning of the regularizer with modality-specific prior knowledge will improve the quality of the reconstruction.

8.7 Appendix - Additional Reconstructed Images

The following reconstructions correspond to the second MRI setup (MRI II) and the reported metrics are PSNR and SSIM.

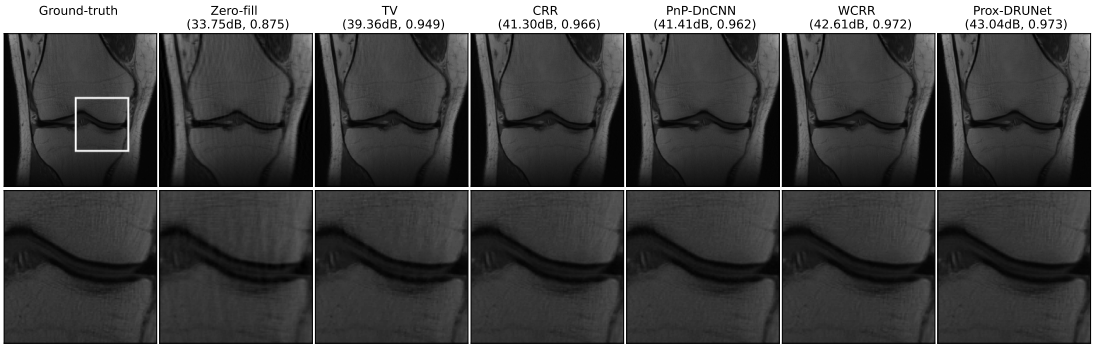

Figure 8.12: Acceleration 2, single coil, PD image.

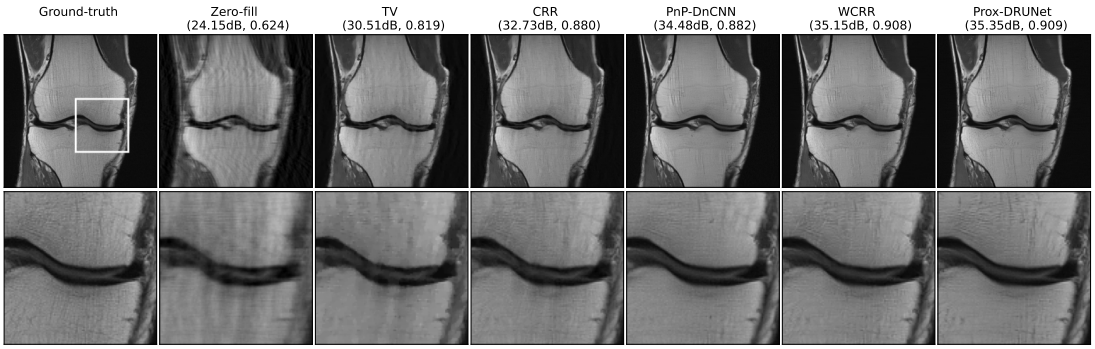

Figure 8.13: Acceleration 4, single coil, PD image.

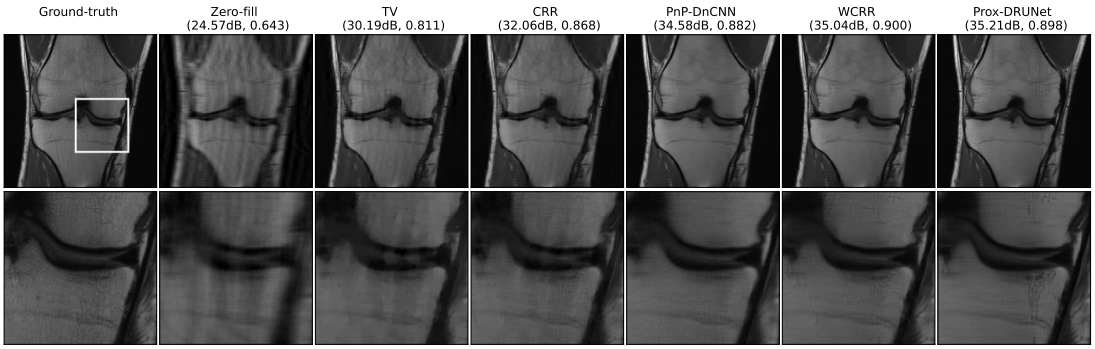

Figure 8.14: Acceleration 4, single coil, PD image.

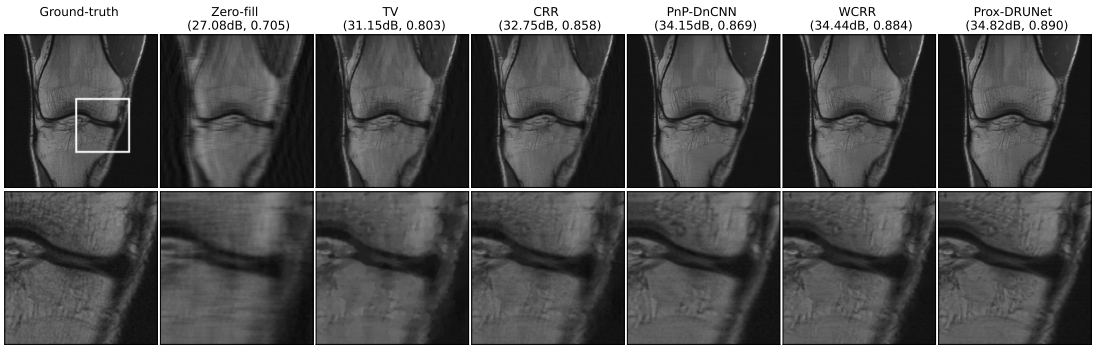

Figure 8.15: Acceleration 4, single coil, PD image.

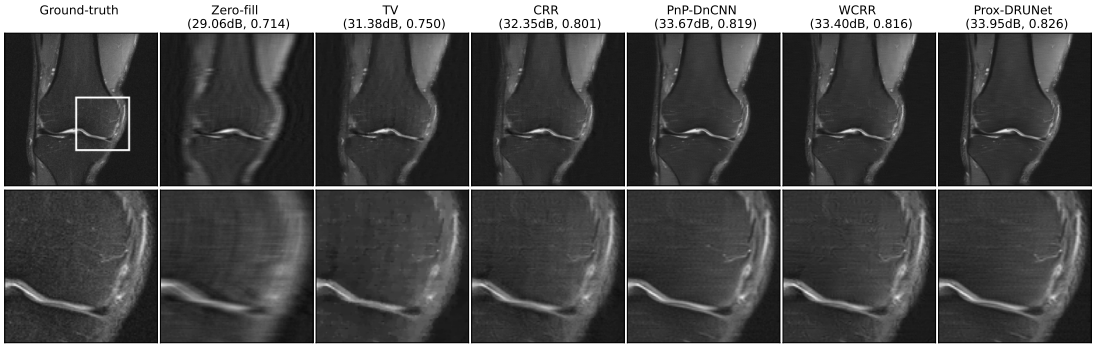

Figure 8.16: Acceleration 4, single coil, PDFS image.

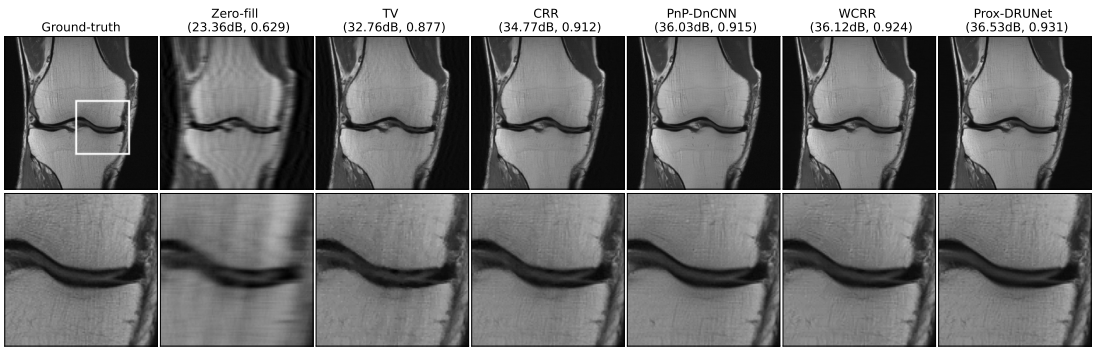

Figure 8.17: Acceleration 4, multiple coils, PD image.

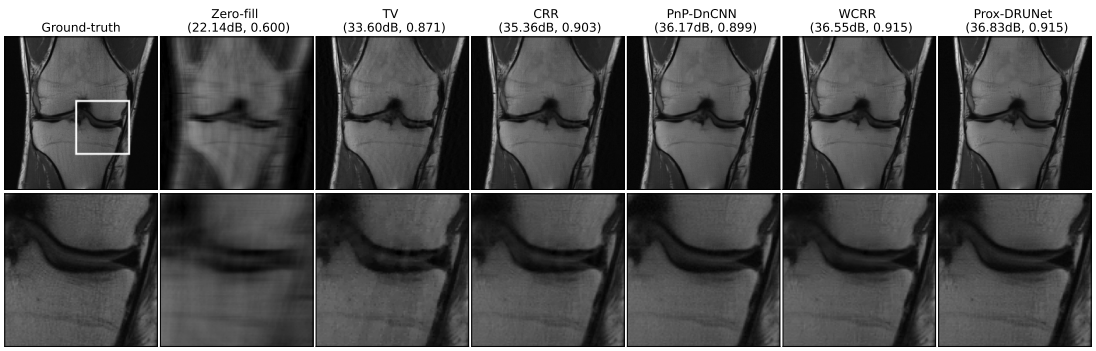

Figure 8.18: Acceleration 8, multiple coils, PD image.

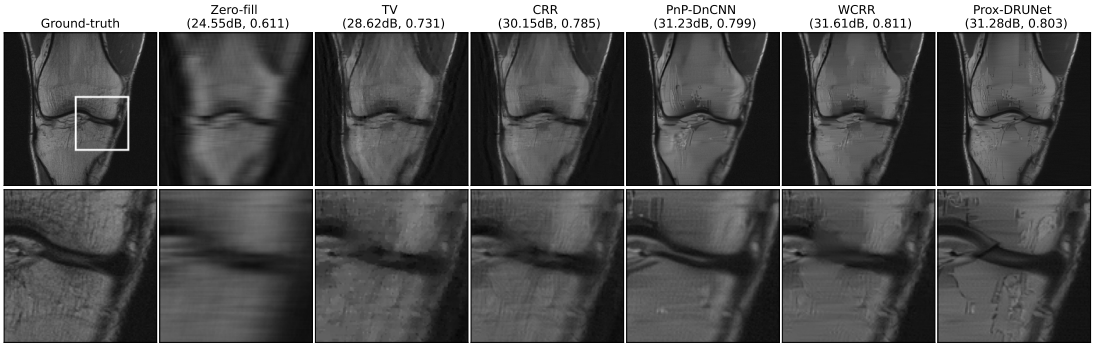

Figure 8.19: Acceleration 8, multiple coils, PD image.

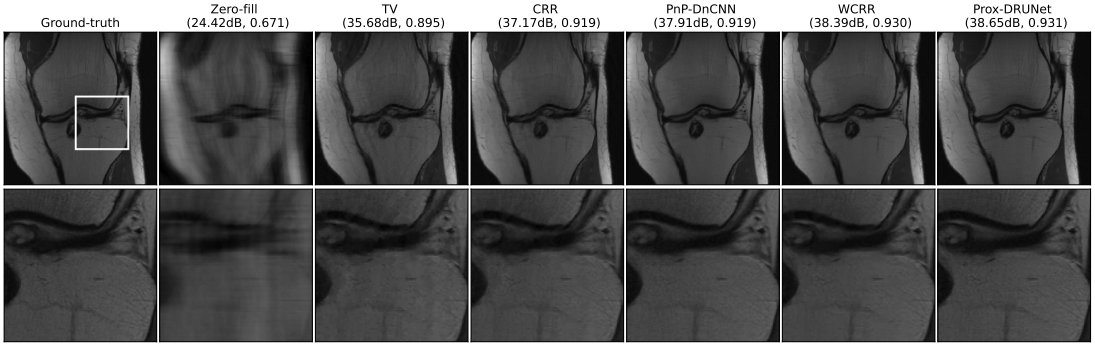

Figure 8.20: Acceleration 8, multiple coils, PD image.

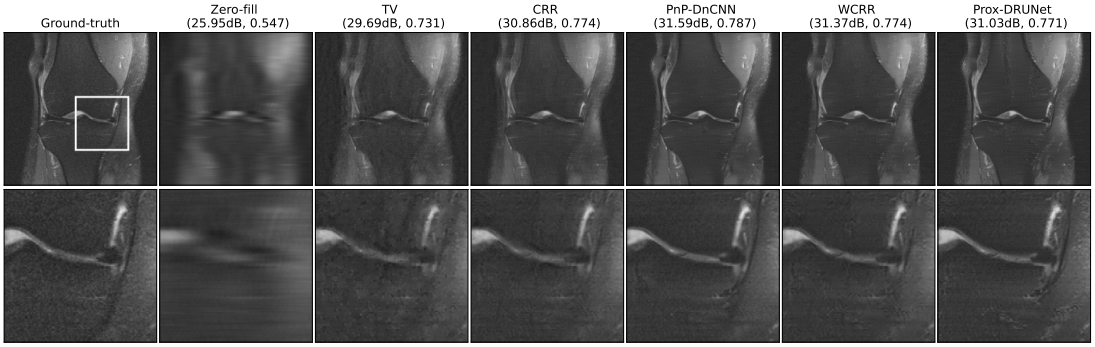

Figure 8.21: Acceleration 8, multiple coils, PDFS image.

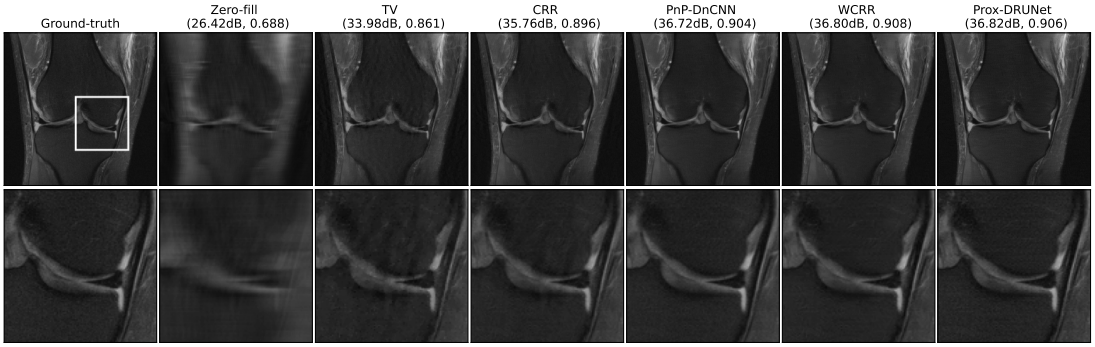

Figure 8.22: Acceleration 8, multiple coils, PDFS image.

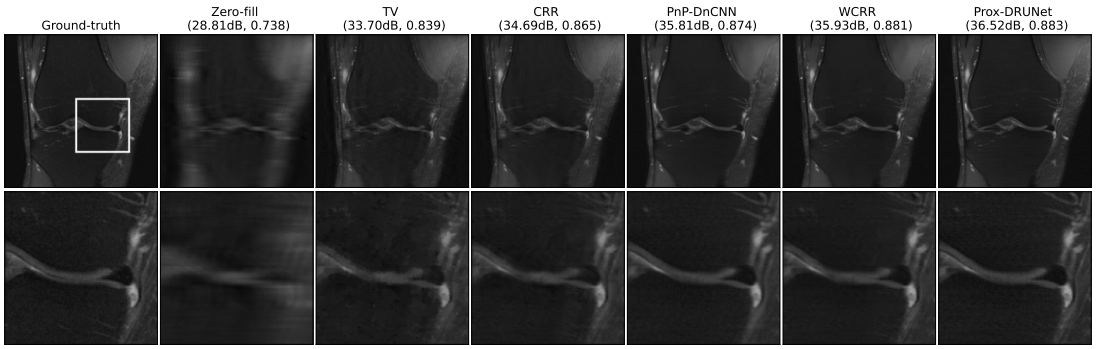

Figure 8.23: Acceleration 8, multiple coils, PDFS.

9 Conclusion

In this thesis, we pursued the goal of improving the trustworthiness of deep-learning-based image reconstruction techniques. Our efforts have been concentrated on devising reconstruction methods with a good tradeoff between interpretability, theoretical guarantees, and empirical performance in diverse settings. To achieve this, we have carefully designed constrained parameterizations and used learnable linear splines to boost expressivity under the constraints.

In Section 9.1, we first summarize the key contributions of the thesis. We then discuss in Section 9.2 possible future directions and we state several important open problems that we have encountered throughout the thesis.

9.1 Summary of the Main Contributions

9.1.1 The World of Splines

In part I, we focused on parameterizations for low-dimensional regression problems and, thereby, illustrated the power of splines as parameterization tools.

Shortest Multi-spline Bases

We first considered general multi-spline spaces. We introduced the notion of mB-splines (Definition 2.4) as a generalization of the well-known B-splines. We proved that mB-splines have the shortest possible support given their approximation abilities (Theorem 2.1) and that they form Riesz bases (Theorem 2.2). Finally, we proposed a recursive procedure to construct mB-splines for any multi-spline space (Theorem 2.3) and gave examples of such functions.

Stable Parameterization of CPWL Functions

We studied the parameterization of continuous and piecewise-linear (CPWL) functions with local hat basis functions. To begin with, we showed that nonlocal parameterization of CPWL functions, such as ReLU neural networks (NNs), are typically ill-conditioned (Section 3.8). We then analyzed the stability of the local parameterization which relies on a simplicial partition of the input domain (Section 3.2.1). We proved that the set of hat functions form a Riesz-basis under mild assumptions (Theorem 3.1), and provided in any number of dimensions the exact Riesz bounds in the case where the hat functions are linear box splines (Theorem 3.3). Finally, we showed how to compute the Lipschitz constant of CPWL functions given their local parameterization and gave fast-to-evaluate lower and upper bounds (Proposition 3.5).

9.1.2 Going Deeper with Stability Guarantees

In part II, we designed Lipschitz-constrained deep NNs (DNNs) with spline activation functions to build convergent and stable plug-and-play (PnP) algorithms.

On the Number of Regions of Continuous and Piecewise-Linear Neural Networks

To better understand deep parameterizations, we studied the expressivity of CPWL DNNs via the counting of their linear regions. To begin with, we introduced the notion of arrangement of convex partitions (Definition 4.5), and gave a sharp upper bound on the number of regions of arrangement of convex partitions (Theorem 4.1). We then provided upper and lower bounds on the maximum number of regions of CPWL DNNs based on their depth, width and the number of regions of their nonlinear modules (Theorem 4.2 and Corollary 4.1). Finally, we derived an upper bound on the expected density of linear regions of a CPWL DNN along 1D paths (Theorem 4.3).

Stable and Convergent Plug-and-Play Methods

To provide stable and convergent PnP algorithms, we studied the design of provably 1-Lipschitz DNNs. We followed the layer-wise approach to constrain each layer to be nonexpansive in terms of the 2-norm. In this context, we revealed some theoretical limitations of popular activation functions, including ReLU and leaky ReLU (Proposition 5.2, Proposition 5.3) as well as PReLU, absolute value, GroupSort and householder (Proposition 5.5). To bypass these shortcomings, we proposed to use Lipschitz-constrained learnable linear splines (LLS) and showed their optimality in this context (Theorem 5.2), even when they have no more than 3 linear regions (Theorem 5.1).

In Chapter 6, we then put the theoretical findings into practice. First, we characterized the stability of the measurement-to-reconstruction map for several PnP algorithms when deployed with Lipschitz-constrained LLS-NN denoisers (Proposition 6.1 and 6.2). We then proposed a reparameterization technique to train efficiently Lipchitz-constrained LLS and gave some underlying theoretical motivations (Proposition 6.3). Finally, we showed the empirical improvements conferred to nonexpansive convolutional NNs (CNNs) when activated by LLS in denoising, MRI and CT image reconstruction compared to nonexpansive CNNs activated by other activation functions.

9.1.3 From Convex to Weakly-convex Data-driven Regularization

In part III, we designed learnable and constrained regularizers to build convergent and sparsity-promoting reconstruction algorithms.

A Sparsity Promoting Convex Regularizer

In Chapter 7 we proposed to learn an interpretable convex regularizer to solve image reconstruction tasks. The gradient of the regularizer was parameterized with a one-hidden-layer CNN with increasing activation functions. In this constrained setting, we proved that LLS activations yield an expressivity that is optimal and superior to the one conferred by ReLU and PReLU activations (Proposition 7.3 and 7.4). We showed that the proposed regularizer yields a stable reconstruction map in the measurement domain for linear inverse problems (Proposition 7.2), and leads to optimization problems for which a minimizer always exists and can be computed (Proposition 7.1). We extended the toolbox introduced in Part II to learn efficiently nonexpansive and nondecreasing LLS activations and introduced a training procedure to learn the one-hidden-layer CNN within a few minutes as a multistep Gaussian denoiser. We analyzed the learned filters and activation functions and qualitatively showed that the regularizer promotes sparsity in a transformed domain. Finally, we deployed the learned CNN to reconstruct CT and MRI images and showed improved performance compared to other convex regularizers, including recent ones parameterized with deep CNNs.

From Convexity to Weak Convexity

In Chapter 8 we extended the framework of Chapter 7 to learn non-convex regularizers with a prescribed upper bound on their weak-convexity modulus. We showed that the parameterization of the gradient of the regularizer with 1-weakly-increasing activation functions and a nonexpansive convolutional layer gives rise to a variational denoiser that minimizes a convex energy (Proposition 8.2). We showed how to efficiently train our 1-weakly convex regularizer on a multi-noise-level denoising task with a bilevel-optimization routine. We analyzed the learned filters and activation functions and

qualitatively showed that the regularizer is promoting images with a sparse representation in a transformed domain. We showed that the proposed regularizer can be used to solve generic image reconstruction tasks via an optimization problem for which a minimizer always exists (Proposition 8.3). We gave an algorithm to compute critical points of the objective (Algorithm 1 and Theorem 8.1). Finally, we tested the performance of the regularizer on CT and MRI reconstruction resulting in an excellent tradeoff between performance, number of parameters, guarantees, and interpretability when compared to other data-driven approaches.

9.2 Future Directions, Open Problems and Perspectives

In the following, we first list some possible extensions of the image reconstruction frameworks developed in this thesis, then discuss some important remaining open problems.

Future Directions

The future directions are sorted from direct developments to more exploratory extensions.

3D Regularization The convex and weakly convex regularizers proposed in this thesis could easily be modified to process 3D volumes. This is expected to be useful for spatiotemporal data and also for CT and MRI images, in which stacks of slices could be reconstructed jointly. In both cases, high-memory requirements can limit the use of the classical deep-learning models, and thus the lightweight models proposed in this thesis could be better suited.

Multi-noise-level Lipschitz Constrained Denoisers We propose to train a Lipschitz-constrained Gaussian denoiser $\mathbf{x} \mapsto \boldsymbol{D}(\mathbf{x}, \sigma)$ at multiple noise levels $\sigma \in [0, \sigma_{\max}]$[1]. The motivation is to boost the generalization performance of the denoiser when used to solve image reconstruction problems and is inspired by the noise-level-dependent weakly convex regularizer introduced in Chapter 8 as well as the DRUNet-based denoiser [337].

A first simple way to achieve this is to use a LLS-NN denoiser (see Chapter 6), and plug 1-Lipschitz noise-level-dependent linear spline activation functions $t \mapsto \varphi(\cdot, \sigma)$. For example, one can use $t \mapsto \alpha(\sigma)\phi(t/\alpha(\sigma))$ as activation, where ϕ is a learnable linear spline and $\alpha(\cdot)$ another learnable spline, which remains 1-Lipschitz if ϕ is 1-Lipschitz. Another possible strategy, inspired by the DRUNet-based denoiser [337], could be to input as an additional channel a noise-level map to the denoiser, and then to analyze the constraints required to ensure nonexpansiveness.

[1] Recall that the denoisers designed in Chapter 6 were trained on a single noise level.

Multicomponent Regularization In some applications, images can be decomposed into multiple components with different characteristics, e.g. smooth and sparse components, low- and high-resolution components, etc. In such cases, it may be more effective to use a different regularizer for each component [153, 347–350]. We thus propose to extend the convex and weakly-convex regularizers introduced in Chapters 7 and 8 to build data-driven and multicomponent regularization schemes.

ICNNs with Learnable Activations In Chapter 7, we have seen that the one-hidden-layer convex regularizer CRR may outperform deep convex regularizers built with input convex NNs (ICNNs) [303]. ICNNs are usually deployed with ReLU, PReLU, or ELU activation functions. The use of ICNNs with learnable linear spline activations with appropriate constraints should generalize CRRs and thus improve the performance of ICNNs while maintaining convexity.

Oracle-based Regularization The reconstruction methods built in Chapter 6 and Chapter 7 rely on nonexpansive denoisers and convex regularizers respectively. Both setups require strong constraints on the models, which in turn provide a trusted reconstruction $\mathbf{x}_{\text{trusted}}$. To boost the performance, we believe that one can incorporate the knowledge of the oracle $\mathbf{x}_{\text{trusted}}$ in the structure of the denoiser or of the regularizer, with possibly no loss of theoretical guarantees, i.e. a convergent reconstruction algorithm and a stable reconstruction map.

- **PnP.** Instead of using a denoiser $\mathbf{x} \mapsto \boldsymbol{D}(\mathbf{x})$, one can plug a denoiser of the form $\mathbf{x} \mapsto \boldsymbol{D}(\mathbf{x}, \mathbf{x}_{\text{trusted}})$, that is constrained to be nonexpansive. This idea is exploited in [83] to build convergent PnP algorithms. This setup might benefit from our LLS toolbox and the stability of the reconstruction map could be investigated and compared to the one obtained in Proposition 6.1 and 6.2.
- **Convex regularization.** The oracle can serve to build a convex regularizer, for instance $\tilde{R}\colon \mathbf{x} \mapsto R_{\text{wcvx}}(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}_{\text{trusted}}\|_2^2$ with a 1-weakly convex regularizer R_{wcvx}. Such a regularizer is expected to improve over convex regularizers, and stability of the reconstruction map should hold provided that $\mathbf{y} \mapsto \mathbf{x}_{\text{trusted}}$ is also stable.

Nonlinear Inverse Problems In this thesis, only linear inverse problems were considered, but many reconstruction techniques explored can be used to solve nonlinear inverse problems. The optimization task would typically be non-convex, but convergent algorithms should still exist, for instance, through an appropriate modification of the safeguarded gradient descent algorithm used to deploy our weakly convex regularizer to solve ill-posed inverse problems.

Nonlocal Regularization The image reconstruction methods proposed in this thesis rely on convolutions. Even with large receptive fields, CNNs are known to be spatially local. In image denoising, using nonlocal information can boost performance, as illustrated for example with non-local mean denoisers [351–353]. The advent of vision transformers [354] calls for the use of parameterizations that facilitate long-range spatial interactions. We have already proposed a first improvement of the weakly convex regularizer in this direction [355]. Possible extensions include the incorporation of an attention module—the core component of transformers—which could lead to theoretical guarantees on the reconstruction if carefully constrained.

Open Problems

Layer-wise Lipschitz Constraints The canonical approach to impose that the Lipschitz constant of a DNN is less than L is to constrain the Lipschitz constant of each of its layers so that the product of their Lipschitz constant is less than L [79]. This layer-wise approach relies on the sub-multiplicativity of the Lipschitz constant with respect to the composition operation. While this strategy seems overly constraining for deep NNs, to the best of our knowledge it is still an open question whether, in the 2-norm setup and given a multidimensional input space[2], the set of 1-Lipschitz layer-wise NNs with free-form and pointwise 1-Lipschitz activation functions is dense in the set of 1-Lipschitz functions or not. If not, then it calls for the design of multivariate Lipschitz-constraint activation functions.

Approximation Properties of Sum of Convex Ridges The approximation capabilities of the sum of convex ridge functions $\mathbf{x} \mapsto \sum \psi_j(\mathbf{w}_j^T \mathbf{x})$ with ψ_j convex, as used in Chapter 7, have not been fully characterized yet to the best of our knowledge for multidimensional input spaces. In particular, can such functions approximate any convex function, alike the convex functions parameterized by ICNNs [356]?

Noise Dependence of the Weakly Convex Regularizer The weakly convex regularizer used for denoising with Gaussian noise seems to scale very simply with the noise level σ, namely $R(\mathbf{x}, \sigma) \simeq \sigma^2 R(\mathbf{x}/\sigma)$. This observation requires first to be further verified experimentally. If such a simple relation indeed holds, it calls for further analysis to discover possible theoretical justifications.

Implicit vs Explicit Regularization, the More Generic, the Less Well-performing? Replacing the proximal operator of a convex regularizer with a nonexpansive

[2]For $\mathbb{R} \to \mathbb{R}$ DNNs with linear-spline activations with 3 regions, we have proved that the layer-wise approach is not overly constraining, see Proposition 5.6

denoiser constitutes a generalization since not all nonexpansive operators are proximal ones[3]. Hence, PnP methods with nonexpansive denoisers should perform at least as well as convex regularization methods. In practice, we have observed the reverse. This paradox suggests that the parameterization and training of nonexpansive denoisers is still not optimal, and calls for further research on the topic.

Perspectives

The deep-learning revolution has pushed the performance of image reconstruction methods one step further, paving the way for major improvements in many applications, including medical imaging, with the possibility of reducing patient exposure and acquisition time. While these advances made classical reconstruction techniques seemingly outdated, the erratic behavior of deep neural networks has highlighted the trustworthiness of the classical techniques. Nowadays, the common approach to maintaining performance while enhancing trustworthiness consists of better balancing the old and the new, for instance with the incorporation of some deep learning into classical reconstruction methods. While it is known that there exists a fundamental tradeoff between performance and stability [289], it is very challenging in practice to assess how close to the limit the current state-of-the-art methods are—to this, the upcoming decade might hold the answer.

[3]The proximal operator of a convex function is firmly nonexpansive, i.e. half averaged and hence nonexpansive.

Bibliography

[1] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli, "Applications of deep learning and reinforcement learning to biological data", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2063–2079, 2018.

[2] M. Reichstein *et al.*, "Deep learning and process understanding for data-driven earth system science", *Nature*, vol. 566, no. 7743, pp. 195–204, Feb. 2019.

[3] D. George and E. Huerta, "Deep learning for real-time gravitational wave detection and parameter estimation: Results with advanced LIGO data", *Physics Letters B*, vol. 778, pp. 64–70, 2018.

[4] A. C. Mater and M. L. Coote, "Deep learning in chemistry", *Journal of Chemical Information and Modeling*, vol. 59, no. 6, pp. 2545–2559, 2019.

[5] K. Choudhary *et al.*, "Recent advances and applications of deep learning methods in materials science", *npj Computational Materials*, vol. 8, no. 1, p. 59, 2022.

[6] Y. Zheng, Z. Xu, and A. Xiao, "Deep learning in economics: A systematic and critical review", *Artificial Intelligence Review*, vol. 56, no. 9, pp. 9497–9539, 2023.

[7] M. Veres and M. Moussa, "Deep learning for intelligent transportation systems: A survey of emerging trends", *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3152–3168, 2020.

[8] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: A survey", *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 315–329, 2020.

[9] T. Lee, S. Mckeever, and J. Courtney, "Flying free: A research overview of deep learning in drone navigation autonomy", *Drones*, vol. 5, no. 2, 2021.

[10] A. Esteva *et al.*, "A guide to deep learning in healthcare", *Nature Medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[11] S. K. Zhou *et al.*, "A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises", *Proceedings of the IEEE*, vol. 109, no. 5, pp. 820–838, 2021.

[12] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 604–624, 2021.

[13] G. Demoment, "Image reconstruction and restoration: Overview of common estimation structures and problems", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, pp. 2024–2036, 1989.

[14] J.-B. Sibarita, "Deconvolution microscopy", *Advances in Biochemical Engineering/Biotechnology*, pp. 201–243, 2005.

[15] P. A. Penczek, "Chapter two - Image restoration in cryo-electron microscopy", in *Cryo-EM, Part B: 3-D Reconstruction*, ser. Methods in Enzymology, G. J. Jensen, Ed., vol. 482, Academic Press, 2010, pp. 35–72.

[16] M. T. McCann and M. Unser, "Biomedical image reconstruction: From the foundations to deep neural networks", *Foundations and Trends® in Signal Processing*, vol. 13, no. 3, pp. 283–359, 2019.

[17] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. Society for Industrial and Applied Mathematics, 2001.

[18] A. Ribes and F. Schmitt, "Linear inverse problems in imaging", *IEEE Signal Processing Magazine*, vol. 25, no. 4, pp. 84–99, 2008.

[19] A. N. Tikhonov, "Solution of incorrectly formulated problems and the regularization method", *Soviet Mathematics*, vol. 4, pp. 1035–1038, 1963.

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso", *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[21] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit", *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[22] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms", *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[23] M. Figueiredo and R. Nowak, "An EM algorithm for wavelet-based image restoration", *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 906–916, 2003.

[24] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint", *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[25] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration", *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 460–489, 2005.

[26] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation", *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[27] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems", *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2419–2434, 2009.

[28] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems", *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[29] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements", *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[30] D. L. Donoho, "Compressed sensing", *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[31] S. Vaiter, C. Deledalle, J. Fadili, G. Peyré, and C. Dossal, "The degrees of freedom of partly smooth regularizers", *Annals of the Institute of Statistical Mathematics*, vol. 69, no. 4, pp. 791–832, 2017.

[32] H. Chen *et al.*, "Low-dose CT with a residual encoder-decoder convolutional neural network.", eng, *IEEE transactions on medical imaging*, vol. 36, pp. 2524–2535, 12 Dec. 2017.

[33] H. Chen *et al.*, "Low-dose CT via convolutional neural network", *Biomedical Optics Express*, vol. 8, no. 2, pp. 679–694, 2017.

[34] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging", *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017.

[35] G. Yang *et al.*, "DAGAN: Deep de-aliasing generative adversarial networks for fast compressed sensing MRI reconstruction", *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1310–1321, 2018.

[36] 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, pp. 18–44, 2021.

[37] S. Linnainmaa, "Taylor expansion of the accumulated rounding error", *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, 1976.

[38] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition", *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[39] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis, "Dying ReLU and initialization: Theory and numerical examples", *arXiv:1903.06733*, 2019.

[40] A. L. Blum and R. L. Rivest, "Training a 3-node neural network is NP-complete", *Neural Networks*, vol. 5, no. 1, pp. 117–127, 1992.

[41] L. Jones, "The computational intractability of training sigmoidal neural networks", *IEEE Transactions on Information Theory*, vol. 43, no. 1, pp. 167–173, 1997.

[42] J. Síma, "Training a single sigmoidal neuron is hard.", eng, *Neural computation*, vol. 14, pp. 2709–28, 11 Nov. 2002.

[43] D. Kingma and J. Ba, "Adam: A method for stochastic optimization", in *International Conference on Learning Representations*, 2015.

[44] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization", in *International Conference on Learning Representations*, 2017.

[45] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design", *SIAM Journal on Scientific Computing*, vol. 43, no. 6, B1105–B1132, 2021.

[46] J. Chen and Y. Liu, "Probabilistic physics-guided machine learning for fatigue data analysis", *Expert Systems with Applications*, vol. 168, p. 114 316, 2021.

[47] F. Djeumou, C. Neary, E. Goubault, S. Putot, and U. Topcu, "Neural networks with physics-informed architectures and constraints for dynamical systems modeling", in *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, ser. Proceedings of Machine Learning Research, vol. 168, PMLR, Jun. 2022, pp. 263–277.

[48] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation", in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018, pp. 3839–3848.

[49] C. Szegedy *et al.*, "Intriguing properties of neural networks", *arXiv:1312.6199*, 2014.

[50] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples", *arXiv:1412.6572*, 2014.

[51] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[52] H. Xu *et al.*, "Adversarial attacks and defenses in images, graphs and text: A review", *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 151–178, Apr. 2020.

[53] N. Narodytska and S. Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks", in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1310–1318.

[54] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks", in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jun. 2019, pp. 2484–2493.

[55] D. Hendrycks *et al.*, "The many faces of robustness: A critical analysis of out-of-distribution generalization", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 8340–8349.

[56] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks", in *International Conference on Machine Learning*, 2019.

[57] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen, "On instabilities of deep learning in image reconstruction and the potential costs of AI", *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 088–30 095, 2020.

[58] M. Z. Darestani, A. S. Chaudhari, and R. Heckel, "Measuring robustness in deep learning based compressive sensing", in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, PMLR, Jul. 2021, pp. 2433–2444.

[59] 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, pp. 1119–1134, 2023.

[60] M. J. Muckley *et al.*, "Results of the 2020 fastMRI challenge for machine learning MR image reconstruction", *IEEE Transactions on Medical Imaging*, vol. 40, no. 9, pp. 2306–2317, 2021.

[61] G. Nataraj and R. Otazo, "Model-free deep MRI reconstruction: A robustness study", in *ISMRM Workshop on Data Sampling and Image*, 2020.

[62] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning", in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, IEEE, 2018, pp. 80–89.

[63] X. Huang *et al.*, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability", *Computer Science Review*, vol. 37, p. 100 270, 2020.

[64] S. Christin, É. Hervet, and N. Lecomte, "Going further with model verification and deep learning", *Methods in Ecology and Evolution*, vol. 12, no. 1, pp. 130–134, 2021.

[65] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems", in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17, Shanghai, China: Association for Computing Machinery, 2017, pp. 1–18.

[66] S. Chakraborty *et al.*, "Interpretability of deep learning models: A survey of results", in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2017, pp. 1–6.

[67] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning", *Journal of Artificial Intelligence Research*, vol. 70, pp. 245–317, 2021.

[68] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks", in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[69] G. Mårtensson *et al.*, "The reliability of a deep learning model in clinical out-of-distribution MRI data: A multicohort study", *Medical Image Analysis*, vol. 66, p. 101 714, 2020.

[70] G. F. Montùfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks", in *Proceedings of the 27th Conference on Advances in Neural Information Processing Systems*, vol. 27, Montréal, Canada, Dec. 2014.

[71] R. Balestriero and R. Baraniuk, "A spline theory of deep learning", in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, Jul. 2018, pp. 374–383.

[72] A. I. Humayun, R. Balestriero, G. Balakrishnan, and R. G. Baraniuk, "SplineCam: Exact visualization and characterization of deep network geometry and decision boundaries", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 3789–3798.

[73] T. Serra, C. Tjandraatmadja, and S. Ramalingam, "Bounding and counting linear regions of deep neural networks", *35th International Conference on Machine Learning, ICML 2018*, vol. 10, pp. 7243–7261, 2018.

[74] R. Parhi and R. D. Nowak, "Deep learning meets sparse regularization: a signal processing perspective", *IEEE Signal Processing Magazine*, vol. 40, no. 6, pp. 63–74, 2023.

[75] J.-C. Pesquet, A. Repetti, M. Terris, and Y. Wiaux, "Learning maximally monotone operators for image recovery", *SIAM Journal on Imaging Sciences*, vol. 14, no. 3, pp. 1206–1237, 2021.

[76] S. Hurault, A. Leclaire, and N. Papadakis, "Proximal denoiser for convergent Plug-and-Play optimization with nonconvex regularization", in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162, PMLR, Jul. 2022, pp. 9483–9505.

[77] A. Pramanik and M. Jacob, "Improved model based deep learning using monotone operator learning (MOL)", in *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, 2022, pp. 1–4.

[78] A. Pramanik, M. B. Zimmerman, and M. Jacob, "Memory-efficient model-based deep learning with convergence and robustness guarantees", *IEEE Transactions on Computational Imaging*, vol. 9, pp. 260–275, 2023.

[79] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks", in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 214–223.

[80] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks", in *International Conference on Learning Representations*, 2018, pp. 1–26.

[81] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-Play methods provably converge with properly trained denoisers", in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, PMLR, vol. 97, PMLR, Jun. 2019, pp. 5546–5557.

[82] C. Anil, J. Lucas, and R. Grosse, "Sorting out Lipschitz function approximation", in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97, Long Beach: PMLR, 2019, pp. 291–301.

[83] J. Hertrich, S. Neumayer, and G. Steidl, "Convolutional proximal neural networks and Plug-and-Play algorithms", *Linear Algebra and Applications*, vol. 631, pp. 203–234, 2021.

[84] S. Singla, S. Singla, and S. Feizi, "Improved deterministic l2 robustness on CIFAR-10 and CIFAR-100", *International Conference on Learning Representations*, 2022.

[85] B. Zhang, D. Jiang, D. He, and L. Wang, "Rethinking Lipschitz neural networks and certified robustness: A Boolean function perspective", *Advances in Neural Information Processing Systems*, vol. 35, pp. 19 398–19 413, 2022.

[86] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation", *arXiv:1410.8516*, 2014.

[87] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen, "Invertible residual networks", in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jun. 2019, pp. 573–582.

[88] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs", *Advances in Neural Information Processing Systems*, vol. 30, pp. 2644–2655, 2017.

[89] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction", in *IEEE Global Conference on Signal and Information Processing*, IEEE, 2013, pp. 945–948.

[90] 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, pp. 1804–1844, 2017.

[91] R. Nowak and M. Figueiredo, "Fast wavelet-based image deconvolution using the EM algorithm", in *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat.No.01CH37256)*, vol. 1, 2001, 371–375 vol.1.

[92] M. Elad, B. Kawar, and G. Vaksman, "Image denoising: The deep learning revolution and beyond—a survey paper", *SIAM Journal on Imaging Sciences*, vol. 16, no. 3, pp. 1594–1654, 2023.

[93] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-Play ADMM for image restoration: Fixed-point convergence and applications", *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 84–98, 2016.

[94] H. Gupta, K. H. Jin, H. Q. Nguyen, M. T. McCann, and M. Unser, "CNN-based projected gradient descent for consistent CT image reconstruction", *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1440–1453, 2018.

[95] Y. Sun, B. Wohlberg, and U. S. Kamilov, "An online plug-and-play algorithm for regularized image reconstruction", *IEEE Transactions on Computational Imaging*, vol. 5, no. 3, pp. 395–408, 2019.

[96] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods", *Mathematical Programming*, vol. 137, pp. 91–129, 2013.

[97] R. Gribonval and M. Nikolova, "A characterization of proximity operators", *Journal of Mathematical Imaging and Vision*, vol. 62, no. 6-7, pp. 773–789, 2020.

[98] S. Hurault, A. Chambolle, A. Leclaire, and N. Papadakis, "A relaxed proximal gradient descent algorithm for convergent plug-and-play with proximal denoiser", in *Scale Space and Variational Methods in Computer Vision*, Springer, 2023, pp. 379–392.

[99] Z. Zou, J. Liu, B. Wohlberg, and U. S. Kamilov, "Deep equilibrium learning of explicit regularization functionals for imaging inverse problems", *IEEE Open Journal of Signal Processing*, vol. 4, pp. 390–398, 2023.

[100] Y. Sun, J. Liu, and U. Kamilov, "Block coordinate regularization by denoising", in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[101] J. Liu, S. Asif, B. Wohlberg, and U. Kamilov, "Recovery analysis for Plug-and-Play priors using the restricted eigenvalue condition", in *Advances in Neural Information Processing Systems*, 2021.

[102] S. Hurault, A. Leclaire, and N. Papadakis, "Gradient step denoiser for convergent Plug-and-Play", in *International Conference on Learning Representations*, 2022.

[103] K. Hammernik *et al.*, "Learning a variational network for reconstruction of accelerated MRI data", *Magnetic Resonance in Medicine*, vol. 79, no. 6, pp. 3055–3071, 2018.

[104] H. K. Aggarwal, M. P. Mani, and M. Jacob, "MoDL: Model-based deep learning architecture for inverse problems", *IEEE Transactions on Medical Imaging*, vol. 38, no. 2, pp. 394–405, 2019.

[105] A. Goujon, S. Aziznejad, A. Naderi, and M. Unser, "Shortest-support multispline bases for generalized sampling", *Journal of Computational and Applied Mathematics*, vol. 395, p. 113 610, 2021.

[106] C. E. Shannon, "Communication in the presence of noise", *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[107] A. J. Jerri, "The Shannon sampling theorem–its various extensions and applications: A tutorial review", *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1565–1596, 1977.

[108] M. Unser, "Sampling—50 years after Shannon", *Proceedings of the IEEE*, vol. 88, no. 4, pp. 569–587, Apr. 2000.

[109] A. Papoulis, "Generalized sampling expansion", *IEEE Transactions on Circuits and Systems*, vol. 24, no. 11, pp. 652–654, 1977.

[110] A. Aldroubi and M. Unser, "Sampling procedures in function spaces and asymptotic equivalence with Shannon's sampling theory", *Numerical Functional Analysis and Optimization*, vol. 15, no. 1-2, pp. 1–21, 1994.

[111] M. Unser and A. Aldroubi, "A general sampling theory for nonideal acquisition devices", *IEEE Transactions on Signal Processing*, vol. 42, no. 11, pp. 2915–2925, 1994.

[112] R. Hummel, "Sampling for spline reconstruction", vol. 43, no. 2, pp. 278–288, 1983.

[113] M. Unser, A. Aldroubi, and M. Eden, "Polynomial spline signal approximations: Filter design and asymptotic equivalence with Shannon's sampling theorem", *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 95–103, 1992.

[114] A. Aldroubi, M. Unser, and M. Eden, "Cardinal spline filters: Stability and convergence to the ideal sinc interpolator", *Signal Processing*, vol. 28, no. 2, pp. 127–138, 1992.

[115] M. Unser and J. Zerubia, "A generalized sampling theory without band-limiting constraints", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 8, pp. 959–969, 1998.

[116] M. Unser and J. Zerubia, "Generalized sampling: Stability and performance analysis", vol. 45, no. 12, pp. 2941–2950, 1997.

[117] A. G. García, M. A. Hernández-Medina, and G. Pérez-Villalón, "Generalized sampling in shift-invariant spaces with multiple stable generators", *Journal of Mathematical Analysis and Applications*, vol. 337, no. 1, pp. 69–84, 2008.

[118] V. Pohl and H. Boche, "U-invariant sampling and reconstruction in atomic spaces with multiple generators", *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3506–3519, 2012.

[119] R. Radha, K. Sarvesh, and S. Sivananthan, "Sampling and reconstruction in a shift invariant space with multiple generators", *Numerical Functional Analysis and Optimization*, vol. 40, no. 4, pp. 365–385, 2019.

[120] C. de Boor, R. A. DeVore, and A. Ron, "The structure of finitely generated shift-invariant spaces in L2(Rd)", *Journal of Functional Analysis*, vol. 119, no. 1, pp. 37–78, 1994.

[121] A. Aldroubi, "Oblique projections in atomic spaces", *Proceedings of the American Mathematical Society*, vol. 124, no. 7, pp. 2051–2060, 1996.

[122] K. Gröchenig, J. L. Romero, and J. Stöckler, "Sampling theorems for shift-invariant spaces, Gabor frames, and totally positive functions", *Inventiones Mathematicae*, vol. 211, no. 3, pp. 1119–1148, 2018.

[123] C. de Boor, R. A. DeVore, and A. Ron, "Approximation from shift-invariant subspaces of L2(Rd)", *Transactions of the American Mathematical Society*, vol. 341, no. 2, pp. 787–806, 1994.

[124] C. de Boor and R. A. DeVore, "Partitions of unity and approximation", *Proceedings of the American Mathematical Society*, vol. 93, no. 4, pp. 705–709, 1985.

[125] T. Blu, P. Thévenaz, and M. Unser, "MOMS: Maximal-order interpolation of minimal support", *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1069–1080, 2001.

[126] M. Unser, "Splines: A perfect fit for signal and image processing", *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, 1999.

[127] I. J. Schoenberg, *Cardinal Spline Interpolation*. SIAM, 1973.

[128] I. J. Schoenberg, "On spline interpolation at all integer points of the real axis", *Séminaire Delange-Pisot-Poitou. Théorie des nombres*, vol. 9, no. 1, pp. 1–18, 1967.

[129] C. de Boor, "Splines as linear combinations of B-splines. a survey", *Approximation Theory*, 1976.

[130] C. de Boor, "On calculating with B-splines", *Journal of Approximation Theory*, vol. 6, no. 1, pp. 50–62, 1972.

[131] C. de Boor, *A Practical Guide to Splines*. Springer-Verlag New York, 1978.

[132] M. Unser and A. Aldroubi, "B-spline signal processing: Part I–theory", *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 821–833, 1993.

[133] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing: Part II–efficient design and applications", *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 834–848, 1993.

[134] P. R. Lipow and I. J. Schoenberg, "Cardinal interpolation and spline functions. III. cardinal Hermite interpolation", *Linear Algebra and Its Applications*, vol. 6, pp. 273–304, 1973.

[135] J. Fageot, S. Aziznejad, M. Unser, and V. Uhlmann, "Support and approximation properties of Hermite splines", *Journal of Computational and Applied Mathematics*, vol. 368, no. 112503, pp. 1–15, 2020.

[136] R. T. Farouki, "The Bernstein polynomial basis: A centennial retrospective", *Computer Aided Geometric Design*, vol. 26, no. 6, pp. 379–419, 2012.

[137] V. Uhlmann, J. Fageot, and M. Unser, "Hermite snakes with control of tangents", *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2803–2816, 2016.

[138] C. Conti, L. Romani, and M. Unser, "Ellipse-preserving Hermite interpolation and subdivision", *Journal of Mathematical Analysis and Applications*, vol. 426, no. 1, pp. 221–227, 2015.

[139] C. Conti, M. Cotronei, and T. Sauer, "Factorization of Hermite subdivision operators preserving exponentials and polynomials", *Advances in Computational Mathematics*, vol. 42, no. 5, pp. 1055–1079, Oct. 2016.

[140] L. Romani and A. Viscardi, "On the refinement matrix mask of interpolating Hermite splines", *Applied Mathematics Letters*, vol. 109, p. 106 524, 2020.

[141] O. Christensen, *An Introduction to Frames and Riesz Bases*. Springer, 2016.

[142] M. Unser and P. D. Tafti, *An Introduction to Sparse Stochastic Processes*. Cambridge University Press, 2014, p. 367.

[143] G. Strang and G. Fix, "A Fourier analysis of the finite element variational method", in *Constructive Aspects of Functional Analysis*, Springer, 2011, pp. 793–840.

[144] C. de Boor, R. A. DeVore, and A. Ron, "Approximation orders of FSI spaces in L2(Rd)", *Constructive Approximation*, vol. 14, no. 4, pp. 631–652, 1998.

[145] M. Unser and I. Daubechies, "On the approximation power of convolution-based least squares versus interpolation", *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1697–1711, 1997.

[146] S. Aziznejad, A. Naderi, and M. Unser, "Optimal spline generators for derivative sampling", in *2019 13th International conference on Sampling Theory and Applications (SampTA)*, IEEE, 2019, pp. 1–4.

[147] M. Antonelli, C. V. Beccari, and G. Casciola, "A general framework for the construction of piecewise-polynomial local interpolants of minimum degree", *Advances in Computational Mathematics*, vol. 40, no. 4, pp. 945–976, 2014.

[148] D. Ranirina and J. de Villiers, "On Hermite vector splines and multi-wavelets", *Journal of Computational and Applied Mathematics*, vol. 349, pp. 366–378, 2019.

[149] M. Lachance, "An introduction to splines for use in computer graphics and geometric modeling", *Computer Vision, Graphics, and Image Processing*, 1990.

[150] P. Alfeld, "On the dimension of multivariate piecewise polynomials", *Numerical analysis*, pp. 1–23, 1986.

[151] D. Petrinović, "Causal cubic splines: Formulations, interpolation properties and implementations", *IEEE Transactions on Signal Processing*, vol. 56, no. 11, pp. 5442–5453, 2008.

[152] H. P. Langtangen and K.-A. Mardal, "Function approximation by finite elements", in *Introduction to Numerical Methods for Variational Problems*. Springer International Publishing, 2019, pp. 69–129.

[153] T. Debarre, S. Aziznejad, and M. Unser, "Hybrid-spline dictionaries for continuous-domain inverse problems", *IEEE Transactions on Signal Processing*, vol. 67, no. 22, pp. 5824–5836, Nov. 2019.

[154] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[155] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[156] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks", in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, vol. 15, Apr. 2011, pp. 315–323.

[157] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units", in *International Conference on Learning Representations*, 2018.

[158] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks", in *29th Annual Conference on Learning Theory*, vol. 49, Columbia University, New York, New York, USA: PMLR, 2016, pp. 907–940.

[159] H. Mhaskar and T. Poggio, "Deep vs. shallow networks: An approximation theory perspective", *Analysis and Applications*, vol. 14, no. 06, pp. 829–848, Aug. 2016.

[160] 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, pp. 503–519, Feb. 2017.

[161] H. Gouk, E. Frank, B. Pfahringer, and M. Cree, "Regularisation of neural networks by enforcing Lipschitz continuity", *Machine Learning*, vol. 110, no. 2, pp. 393–416, 2021.

[162] J. He, L. Li, J. Xu, and C. Zheng, "ReLU deep neural networks and linear finite elements", *Journal of Computational Mathematics*, vol. 38, no. 3, pp. 502–527, 2020.

[163] C. De Boor, K. Höllig, and S. D. Riemenschneider, *Box Splines* (Applied Mathematical Sciences). New York, NY: Springer, 2011, vol. 98.

[164] M. Kim and J. Peters, "Symmetric box-splines on the A_{n*} lattice", *Journal of Approximation Theory*, vol. 162, no. 9, pp. 1607–1630, Sep. 2010.

[165] Y. Liu and G. Yin, "Nonparametric functional approximation with Delaunay triangulation learner", in *2019 IEEE International Conference on Big Knowledge (ICBK)*, 2019, pp. 167–174.

[166] Y. Liu and G. Yin, "The Delaunay triangulation learner and its ensembles", *Computational Statistics & Data Analysis*, vol. 152, p. 107 030, 2020.

[167] J. Gu and G. Yin, "Crystallization learning with the Delaunay triangulation", in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 3854–3863.

[168] J. Campos, S. Aziznejad, and M. Unser, "Learning of continuous and piecewise-linear functions with Hessian total-variation regularization", *IEEE Open Journal of Signal Processing*, vol. 3, pp. 36–48, 2022.

[169] M. Pourya, A. Goujon, and M. Unser, "Delaunay-triangulation-based learning with Hessian total-variation regularization", *IEEE Open Journal of Signal Processing*, vol. 4, pp. 167–178, 2023.

[170] C. D. Aliprantis, D. Harris, and R. Tourky, "Continuous piecewise linear functions", *Macroeconomic Dynamics*, vol. 10, no. 1, p. 77, 2006.

[171] S. Wang and X. Sun, "Generalization of hinging hyperplanes", *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4425–4431, 2005.

[172] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models", in *Proceedings of the 30th International Conference on Machine Learning*, Citeseer, vol. 30, 2013, p. 3.

[173] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification", in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[174] W. Shang, K. Sohn, D. Almeida, and H. Lee, "Understanding and improving convolutional neural networks via concatenated rectified linear units", in *33rd International Conference on Machine Learning*, 2016, pp. 2217–2225.

[175] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks", in *30th International Conference on Machine Learning*, International Machine Learning Society (IMLS), 2013, pp. 2356–2364.

[176] W. Dahmen and C. A. Micchelli, "Translates of multivarlate splines", *Linear Algebra and its Applications*, vol. 52, pp. 217–234, 1983.

[177] Y. Guan, S. Lu, and Y. Tang, "Characterization of compactly supported refinable splines whose shifts form a Riesz basis", *Journal of Approximation Theory*, vol. 133, no. 2, pp. 245–250, 2005.

[178] R. Jia and W. Zhao, "Riesz bases of wavelets and applications to numerical solutions of elliptic equations", *Mathematics of Computation*, vol. 80, no. 275, pp. 1525–1556, 2011.

[179] N. Fukuda, T. Kinoshita, and T. Kubo, "On the finite element method with Riesz bases and its applications to some partial differential equations", in *2013 10th International Conference on Information Technology: New Generations*, 2013, pp. 761–766.

[180] P. L. Bartlett, D. J. Foster, and M. Telgarsky, "Spectrally-normalized margin bounds for neural networks", in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6241–6250.

[181] U. von Luxburg and O. Bousquet, "Distance–based classification with Lipschitz functions", *J. Mach. Learn. Res.*, vol. 5, pp. 669–695, Dec. 2004.

[182] J. Sokolić, R. Giryes, G. Sapiro, and M. R. D. Rodrigues, "Robust large margin deep neural networks", *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4265–4280, 2017.

[183] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples", in *International Conference on Machine Learning*, PMLR, 2017, pp. 854–863.

[184] P. Hagemann and S. Neumayer, "Stabilizing invertible neural networks using mixture models", *Inverse Problems*, vol. 37(8), p. 85 002, 2021.

[185] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy", in *International Conference on Learning Representations, ICLR*, OpenReview.net, 2019.

[186] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks", in *Advances in Neural Information Processing Systems 31*, vol. 31, Curran Associates, Inc., 2018, pp. 6542–6551.

[187] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients", arXiv, Tech. Rep. arXiv:1711.09404, Nov. 2017.

[188] D. Zou, R. Balan, and M. Singh, "On Lipschitz bounds of general convolutional neural networks", *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1738–1759, 2020.

[189] P. Bohra, D. Perdios, A. Goujon, S. Emery, and M. Unser, "Learning Lipschitz-controlled activation functions in neural networks for Plug-and-Play image reconstruction methods", in *NeurIPS 2021 Workshop on Deep Learning and Inverse Problems*, 2021.

[190] S. Adeeb and V. G. Troitsky, "Locally piecewise affine functions and their order structure", *Positivity*, vol. 21, no. 1, pp. 213–221, 2017.

[191] J. De Loera, J. Rambau, and F. Santos, *Triangulations: Structures for Algorithms and Applications*. Springer Science & Business Media, 2010, vol. 25.

[192] H. W. Kuhn, "Some combinatorial lemmas in topology", *IBM Journal of Research and Development*, vol. 4, no. 5, pp. 518–524, 1960.

[193] E. Allgower and K. Georg, "Triangulations by reflections with applications to approximation", in *Numerische Methoden der Approximationstheorie: Vortragsauszüge der Tagung über numerische Methoden der Approximationstheorie vom 13.–19. November 1977 im Mathematischen Forschungsinstitut Oberwolfach (Schwarzwald)*. Birkhäuser Basel, 1978, pp. 9–32.

[194] D. F. Watson, "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes", *The Computer Journal*, vol. 24, no. 2, pp. 167–172, 1981.

[195] V. T. Rajan, "Optimality of the Delaunay triangulation in Rd", *Discrete & Computational Geometry*, vol. 12, no. 2, pp. 189–202, 1994.

[196] A. L. Edmonds, "Simplicial decompositions of convex polytopes", *Pi Mu Epsilon Journal*, vol. 5, no. 3, pp. 124–128, 1970.

[197] D. Van De Ville, T. Blu, M. Unser, W. Philips, I. Lemahieu, and R. Van de Walle, "Hex-splines: A novel spline family for hexagonal lattices", *IEEE Transactions on Image Processing*, vol. 13, no. 6, pp. 758–772, Jun. 2004.

[198] J. B. Lasserre and K. E. Avrachenkov, "The multi-dimensional version of $\int_b^a x^p dx$", *The American Mathematical Monthly*, vol. 108, no. 2, pp. 151–154, 2001.

[199] V. Baldoni, N. Berline, J. A. De Loera, M. Köppe, and M. Vergne, "How to integrate a polynomial over a simplex", *Mathematics of Computation*, vol. 80, no. 273, pp. 297–325, 2010.

[200] C. Heumann, M. Schomaker, and Shalabh, "Combinatorics", in *Introduction to Statistics and Data Analysis : With Exercises, Solutions and Applications in R*. Cham: Springer International Publishing, 2016, pp. 97–107.

[201] I. Kra and S. R. Simanca, "On circulant matrices", *Notices of the American Mathematical Society*, vol. 59, no. 03, p. 368, 2012.

[202] L. Kamenski, W. Huang, and H. Xu, "Conditioning of finite element equations with arbitrary anisotropic meshes", *Mathematics of computation*, vol. 83, no. 289, pp. 2187–2211, 2014.

[203] M. Kim, A. Entezari, and J. Peters, "Box spline reconstruction on the face-centered cubic lattice", *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1523–1530, Nov. 2008.

[204] L. Condat and D. Van De Ville, "Three-directional box-splines: Characterization and efficient evaluation", *IEEE Signal Processing Letters*, vol. 13, no. 7, pp. 417–420, Jul. 2006.

[205] J. Horacsek and U. Alim, "Evaluating box splines with reduced complexity", 2018.

[206] C. de Boor and K. Höllig, "B-splines from parallelepipeds.", Wisconsin Univ-Madison Mathematics Research Center, Tech. Rep., 1982.

[207] W. A. Dahmen and C. A. Micchelli, "On the linear independence of multivariate B-splines, I. triangulations of simploids", Tech. Rep. 5, 1982, pp. 993–1012.

[208] T. Zaslavsky, *Facing up to Arrangements: Face-Count Formulas for Partitions of Space by Hyperplanes*. American Mathematical Society, 1975, vol. 154.

[209] M. Jordan and A. G. Dimakis, "Exactly computing the local Lipschitz constant of ReLU networks", in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 7344–7353.

[210] H. Sedghi, V. Gupta, and P. M. Long, "The singular values of convolutional layers", in *International Conference on Learning Representations*, 2019, pp. 1–12.

[211] S. Neumayer, A. Goujon, P. Bohra, and M. Unser, "Approximation of Lipschitz functions using deep spline neural networks", *SIAM Journal on Mathematics of Data Science*, vol. 5, no. 2, pp. 306–322, 2023.

[212] Y. Bengio, *Learning Deep Architectures for AI*. Now Publishers Inc, 2009.

[213] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[214] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks", *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 2924–2932, 2014.

[215] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks", *International Conference on Learning Representations*, 2015.

[216] M. Unser, "A representer theorem for deep neural networks", *Journal of Machine Learning Research*, vol. 20, no. 110, pp. 1–30, Feb. 2019.

[217] S. Singla, S. Singla, and S. Feizi, "Improved deterministic l2 robustness on CIFAR-10 and CIFAR-100", *arXiv:2108.04062*, 2021.

[218] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778.

[219] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *International Conference on Machine Learning*, PMLR, 2015, pp. 448–456.

[220] A. Goujon, J. Campos, and M. Unser, "Stable parameterization of continuous and piecewise-linear functions", *Applied and Computational Harmonic Analysis*, vol. 67, p. 101 581, 2023.

[221] R. Balestriero and R. Baraniuk, "Mad Max: Affine spline insights into deep learning", *Proceedings of the IEEE*, vol. 109, no. 5, pp. 704–727, May 2021.

[222] J. M. Tarela, E. Alonso, and M. V. Martínez, "A representation method for PWL functions oriented to parallel processing", *Mathematical and Computer Modelling*, vol. 13, no. 10, pp. 75–83, 1990.

[223] J. M. Tarela and M. V. Martínez, "Region configurations for realizability of lattice piecewise-linear models", *Mathematical and Computer Modelling*, 1999.

[224] B. Hanin and D. Rolnick, "Deep ReLU networks have surprisingly few activation patterns", *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019.

[225] R. Balestriero, R. Cosentino, B. Aazhang, and R. Baraniuk, "The geometry of deep networks: Power diagram subdivision", in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[226] P. Hinz and S. de Geer, "A framework for the construction of upper bounds on the number of affine linear regions of ReLU reed-forward neural networks", *IEEE Transactions on Information Theory*, vol. 65, no. 11, pp. 7304–7324, 2019.

[227] H. Xiong, L. Huang, M. Yu, L. Liu, F. Zhu, and L. Shao, "On the number of linear regions of convolutional neural networks", in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119, PMLR, Jul. 2020, pp. 10 514–10 523.

[228] G. Montúfar, Y. Ren, and L. Zhang, "Sharp bounds for the number of regions of maxout networks and vertices of Minkowski sums", *SIAM Journal on Applied Algebra and Geometry*, vol. 6, no. 4, pp. 618–649, 2022.

[229] B. Hanin and D. Rolnick, "Complexity of linear regions in deep networks", *arXiv:1901.09021*, pp. 4585–4600, Jun. 2019.

[230] H. Tseran and G. F. Montúfar, "On the expected complexity of maxout networks", *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 995–29 008, 2021.

[231] U. Tanielian and G. Biau, "Approximating Lipschitz continuous functions with GroupSort neural networks", in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 130, PMLR, 2021, pp. 442–450.

[232] Z. Zhu, Y. Zhou, Y. Dong, and Z. Zhong, "PWLU: Learning specialized activation functions with the piecewise linear unit", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–19, 2023.

[233] E. León and G. M. Ziegler, "Spaces of convex n-partitions", *Bolyai Society of Mathematical Studies*, vol. 27, pp. 279–306, 2018.

[234] D. Bulavka, A. Goodarzi, and M. Tancer, "Optimal bounds for the colorful fractional Helly theorem", *arXiv:2010.15765*, 2020.

[235] G. Montúfar, "Notes on the number of linear regions of deep neural networks", in *2017, 12th International Conference on Sampling Theory and Applications (SampTA)*, Mar. 2017.

[236] P. Bohra, J. Campos, H. Gupta, S. Aziznejad, and M. Unser, "Learning activation functions in deep (spline) neural networks", *IEEE Open Journal of Signal Processing*, vol. 1, pp. 295–309, 2020.

[237] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, "On the expressive power of deep neural networks", in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Aug. 2017, pp. 2847–2854.

[238] B. Hanin, R. Jeong, and D. Rolnick, "Deep ReLU networks preserve expected length", *arXiv:2102.10492*, 2021.

[239] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using Lipschitz bounds", *IEEE Control Systems Letters*, vol. 6, pp. 121–126, 2022.

[240] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning", *Advances in Neural Information Processing Systems*, vol. 31, pp. 5949–5958, 2017.

[241] T. Meinhardt, M. Moeller, C. Hazirbas, and D. Cremers, "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems", in *IEEE International Conference on Computer Vision*, IEEE, 2017, pp. 1799–1808.

[242] S. Sreehariand, S. V. Venkatakrishnan, and B. Wohlberg, "Plug-and-play priors for bright field electron tomography and sparse interpolation", *IEEE Transactions on Computational Imaging*, vol. 2, no. 4, pp. 408–423, 2016.

[243] M. Terris, A. Repetti, J. Pesquet, and Y. Wiaux, "Building firmly nonexpansive convolutional neural networks", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2020, pp. 8658–8662.

[244] L. Bungert, R. Raab, T. Roith, L. Schwinn, and D. Tenbrinck, "CLIP: Cheap Lipschitz training of neural networks", in *Scale Space and Variational Methods in Computer Vision*, Springer, Cham, 2021, pp. 307–319.

[245] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of Lipschitz constants for deep neural networks", in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019, pp. 11 427–11 438.

[246] F. Latorre, P. Rolland, and V. Cevher, "Lipschitz constant estimation of neural networks via sparse polynomial optimization", in *International Conference on Learning Representations*, 2020, pp. 1–16.

[247] K. Roth, Y. Kilcher, and T. Hofmann, "Adversarial training is a form of data-dependent operator norm regularization", in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 14 973–14 985.

[248] O. Calin, *Deep Learning Architectures: A Mathematical Approach*. Springer, Cham, 2020.

[249] M. Hasannasab, J. Hertrich, S. Neumayer, G. Plonka, S. Setzer, and G. Steidl, "Parseval proximal neural networks", *The Journal of Fourier Analysis*, vol. 26, p. 59, 2020.

[250] L. Huang, X. Liu, B. Lang, A. W. Yu, Y. Wang, and B. Li, "Orthogonal weight normalization: Solution to optimization over multiple dependent Stiefel manifolds in deep neural networks", in *32nd AAAI Conference on Artificial Intelligence*, AAAI Press, 2018, pp. 3271–3278.

[251] Q. Li, S. Haque, C. Anil, J. Lucas, R. Grosse, and J.-H. Jacobsen, "Preventing gradient attenuation in Lipschitz constrained convolutional networks", in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019, pp. 15 364–15 376.

[252] T. Huster, C.-Y. J. Chiang, and R. Chadha, "Limitations of the Lipschitz constant as a defense against adversarial examples", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, Cham, 2018, pp. 16–29.

[253] J. E. Cohen, T. P. Huster, and R. Cohen, "Universal Lipschitz approximation in bounded depth neural networks", *arXiv:1904.04861*, 2019.

[254] U. Tanielian, M. Sangnier, and G. Biau, "Approximating Lipschitz continuous functions with GroupSort neural networks", in *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, PMLR, 2021, pp. 442–450.

[255] S. Aziznejad, H. Gupta, J. Campos, and M. Unser, "Deep neural networks with trainable activations and controlled Lipschitz constant", *IEEE Transactions on Signal Processing*, vol. 68, pp. 4688–4699, Aug. 2020.

[256] J. Su, W. Byeon, and F. Huang, "Scaling-up diverse orthogonal convolutional networks by a paraunitary framework", in *Proceedings of the 39th International Conference on Machine Learning*, Jun. 2022.

[257] A. Chernodub and D. Nowicki, "Norm-preserving orthogonal permutation linear unit activation functions (OPLU)", *arXiv:1604.02313*, 2016.

[258] S. Aziznejad and M. Unser, "Deep spline networks with control of Lipschitz regularity", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2019, pp. 3242–3246.

[259] K. Bredies and M. Holler, "Higher-order total variation approaches and generalisations", *Inverse Problems*, vol. 36, no. 12, p. 123 001, 2020.

[260] T. Debarre, Q. Denoyelle, M. Unser, and J. Fageot, "Sparsest piecewise-linear regression of one-dimensional data", *Journal of Computational and Applied Mathematics*, vol. 406, no. C, p. 114 044, 2022.

[261] S. Aziznejad, T. Debarre, and M. Unser, "Sparsest univariate learning models under Lipschitz constraint", *IEEE Open Journal of Signal Processing*, vol. 3, pp. 140–154, 2022.

[262] H. Lin and S. Jegelka, "ResNet with one-neuron hidden layers is a universal approximator", in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018, pp. 6172–6181.

[263] L. Schwartz, *Théorie des Distributions* (Publications de l'Institut de Mathématique de l'Université de Strasbourg, IX-X). Hermann, Paris, 1966, pp. xiii+420.

[264] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling", *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[265] P. Combettes and V. Wajs, "Signal recovery by proximal forward-backward splitting", *Multiscale Modeling & Simulation*, vol. 4, pp. 1168–1200, 2005.

[266] J. Douglas and H. H. Rachford, "On the numerical solution of heat conduction problems in two and three space variables", *Transactions of the American Mathematical Society*, vol. 82, no. 2, pp. 421–439, 1956.

[267] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers", *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[268] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations", *IEEE Transactions on Computational Imaging*, vol. 5, no. 1, pp. 52–67, 2019.

[269] R. Cohen, M. Elad, and P. Milanfar, "Regularization by denoising via fixed-point projection (RED-PRO)", *SIAM Journal on Imaging Sciences*, vol. 14, no. 3, pp. 1374–1406, 2021.

[270] X. Xu, J. Liu, Y. Sun, B. Wohlberg, and U. S. Kamilov, "Boosting the performance of Plug-and-Play priors via denoiser scaling", in *54th Asilomar Conference on Signals, Systems, and Computers*, 2020, pp. 1305–1312.

[271] Y. Sun, Z. Wu, X. Xu, B. Wohlberg, and U. S. Kamilov, "Scalable plug-and-play ADMM with convergence guarantees", *IEEE Transactions on Computational Imaging*, vol. 7, pp. 849–863, 2021.

[272] 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, pp. 3142–3155, 2017.

[273] R. Cohen, Y. Blau, D. Freedman, and E. Rivlin, "It has potential: Gradient-driven denoisers for convergent solutions to inverse problems", *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[274] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer New York, 2011.

[275] I. Shvartsman, "On stability of minimizers in convex programming", *Nonlinear Analysis: Theory, Methods & Applications*, vol. 75, no. 3, pp. 1563–1571, 2012, Variational Analysis and Its Applications.

[276] J. F. Bonnans and A. Shapiro, *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

[277] P. del Aguila Pla, S. Neumayer, and M. Unser, "Stability of image-reconstruction algorithms", *IEEE Transactions on Computational Imaging*, vol. 9, pp. 1–12, 2023.

[278] C. Li *et al.*, "ALICE: Towards understanding adversarial learning for joint distribution matching", in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[279] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-CNN for image restoration", in *Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, 2018, pp. 886–88 609.

[280] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "SwinIR: Image restoration using Swin transformer", in *International Conference on Computer Vision Workshops*, IEEE, 2021, pp. 1833–1844.

[281] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation", en, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[282] F. Knoll *et al.*, "fastMRI: A publicly available raw k-space and DICOM dataset of knee images for accelerated MR image reconstruction using machine learning", *Radiology: Artificial Intelligence*, vol. 2, no. 1, 2020.

[283] M. Uecker *et al.*, "ESPIRiT-an eigenvalue approach to autocalibrating parallel MRI: where SENSE meets GRAPPA", en, *Magn. Reson. Med.*, vol. 71, no. 3, pp. 990–1001, Mar. 2014.

[284] M. Uecker *et al.*, "Software toolbox and programming library for compressed sensing and parallel imaging", in *ISMRM Workshop on Data Sampling and Image Reconstruction*, 2013, p. 41.

[285] C. McCollough, "TU-FG-207A-04: Overview of the low dose CT Grand Challenge", *Medical Physics*, vol. 43, no. 6Part35, pp. 3759–3760, 2016.

[286] S. Mukherjee, S. Dittmer, Z. Shumaylov, S. Lunz, O. Öktem, and C.-B. Schönlieb, "Learned convex regularizers for inverse problems", *arXiv:2008.02839*, 2021.

[287] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, "Solving inverse problems using data-driven models", *Acta Numerica*, vol. 28, pp. 1–174, 2019.

[288] G. Ongi, A. Jalal, C. A. Metzle, R. G. Baraniuk, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging", *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 39–56, 2020.

[289] N. M. Gottschling, V. Antun, B. Adcock, and A. C. Hansen, "The troublesome kernel: Why deep learning for inverse problems is typically unstable", *arXiv:2001.01258*, 2020.

[290] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen, "Image reconstruction by domain-transform manifold learning", *Nature*, vol. 555, no. 7697, pp. 487–492, 2018.

[291] C. M. Hyun, H. P. Kim, S. M. Lee, S. Lee, and J. K. Seo, "Deep learning for undersampled MRI reconstruction", *Physics in Medicine & Biology*, vol. 63, no. 13, p. 135 007, 2018.

[292] S. Roth and M. J. Black, "Fields of experts", *International Journal of Computer Vision*, vol. 82, no. 2, pp. 205–229, 2009.

[293] Y. Chen, R. Ranftl, and T. Pock, "Insights into analysis operator learning: From patch-based sparse models to higher order MRFs", *IEEE Transactions on Image Processing*, vol. 23, pp. 1060–72, 2014.

[294] A. Effland, E. Kobler, K. Kunisch, and T. Pock, "Variational networks: An optimal control approach to early stopping variational methods for image restoration", *Journal of Mathematical Imaging and Vision*, vol. 62, no. 3, pp. 396–416, 2020.

[295] S. Lunz, O. Öktem, and C.-B. Schönlieb, "Adversarial regularizers in inverse problems", *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[296] M. Duff, N. D. F. Campbell, and M. J. Ehrhardt, "Regularising inverse problems with generative machine learning models", *arXiv:2107.11191*, 2021.

[297] H. Li, J. Schwab, S. Antholzer, and M. Haltmeier, "NETT: Solving inverse problems with deep neural networks", *Inverse Problems*, vol. 36, no. 6, p. 065 005, 2020.

[298] E. Kobler, A. Effland, K. Kunisch, and T. Pock, "Total deep variation for linear inverse problems", in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[299] R. Fermanian, M. Le Pendu, and C. Guillemot, "PnP-ReG: Learned regularizing gradient for plug-and-play gradient descent", *SIAM Journal on Imaging Sciences*, vol. 16, no. 2, pp. 585–613, 2023.

[300] E. Kobler, T. Klatzer, K. Hammernik, and T. Pock, "Variational networks: Connecting variational methods and deep learning", in *Pattern Recognition*, 2017, pp. 281–293.

[301] P. Nair and K. N. Chaudhury, "On the construction of averaged deep denoisers for image regularization", *arXiv:2207.07321*, 2022.

[302] S. Mukherjee, C.-B. Schönlieb, and M. Burger, "Learning convex regularizers satisfying the variational source condition for inverse problems", in *NeurIPS Workshop on Deep Learning and Inverse Problems*, 2021.

[303] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks", in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Aug. 2017, pp. 146–155.

[304] H. Q. Nguyen, E. Bostan, and M. Unser, "Learning convex regularizers for optimal Bayesian denoising", *IEEE Transactions on Signal Processing*, vol. 66, no. 4, pp. 1093–1105, 2017.

[305] G. Peyré and J. M. Fadili, "Learning analysis sparsity priors", in *SampTA'11*, 2011, p. 4.

[306] Y. Chen, T. Pock, and H. Bischof, "Learning ℓ_1-based analysis and synthesis sparsity priors using bi-level optimization", in *26th Neural Information Processing Systems Confercence*, 2012.

[307] L. B. Willner, "On the distance between polytopes", *Quarterly of Applied Mathematics*, vol. 26, no. 2, pp. 207–212, 1968.

[308] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models", in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[309] D. Gilton, G. Ongie, and R. Willett, "Deep equilibrium architectures for inverse problems in imaging", *IEEE Transactions on Computational Imaging*, vol. 7, pp. 1123–1133, 2021.

[310] A. Pramanik, H. K. Aggarwal, and M. Jacob, "Deep generalization of structured low-rank algorithms (deep-slr)", *IEEE Transactions on Medical Imaging*, vol. 39, no. 12, pp. 4186–4197, 2020.

[311] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin, "JFB: Jacobian-free backpropagation for implicit networks", in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[312] Y. Malitsky and K. Mishchenko, "Adaptive gradient descent without descent", in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119, PMLR, Jul. 2020, pp. 6702–6712.

[313] P. Latafat, A. Themelis, L. Stella, and P. Patrinos, "Adaptive proximal algorithms for convex optimization under local Lipschitz continuity of the gradient", *arXiv:2301.04431*, 2023.

[314] S. Ducotterd, A. Goujon, P. Bohra, D. Perdios, S. Neumayer, and M. Unser, "Improving Lipschitz-constrained neural networks by learning activation functions", *arXiv:2210.16222*, 2022.

[315] J. R. Chand and M. Jacob, "Multi-scale energy (MuSE) plug and play framework for inverse problems", *arXiv:2305.04775*, 2023.

[316] A. Chambolle, "An algorithm for total variation minimization and applications", *Journal of Mathematical imaging and vision*, vol. 20, no. 1, pp. 89–97, 2004.

[317] 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, pp. 891–906, 1991.

[318] M. Unser and N. Chenouard, "A unifying parametric framework for 2D steerable wavelet transforms", *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 102–135, 2013.

[319] Y. Nesterov, "Smooth convex optimization", in *Introductory Lectures on Convex Optimization:A Basic Course*. Boston, MA: Springer US, 2004, pp. 51–110.

[320] M. Nikolova, "Energy minimization methods", in *Handbook of Mathematical Methods in Imaging*, Springer, New York, 2015, pp. 157–204.

[321] A. Lanza, S. Morigi, I. W. Selesnick, and F. Sgallari, "Convex non-convex variational models", in *Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging: Mathematical Imaging and Vision*. Cham: Springer International Publishing, 2021, pp. 1–57.

[322] B. Tan, Y. Li, H. Zhao, X. Li, and S. Ding, "A novel dictionary learning method for sparse representation with nonconvex regularizations", *Neurocomputing*, vol. 417, pp. 128–141, 2020.

[323] J. Li, J. Li, Z. Xie, and J. Zou, "Plug-and-Play ADMM for MRI reconstruction with convex nonconvex sparse regularization", *IEEE Access*, vol. 9, pp. 148 315–148 324, 2021.

[324] J. Abe, M. Yamagishi, and I. Yamada, "Linearly involved generalized Moreau enhanced models and their proximal splitting algorithm under overall convexity condition", *Inverse Problems*, vol. 36, no. 3, p. 035 012, 2020.

[325] A. Lanza, S. Morigi, I. W. Selesnick, and F. Sgallari, "Sparsity-inducing nonconvex nonseparable regularization for convex image processing", *SIAM Journal on Imaging Sciences*, vol. 12, no. 2, pp. 1099–1134, 2019.

[326] J. Zou, M. Shen, Y. Zhang, H. Li, G. Liu, and S. Ding, "Total variation denoising with non-convex regularizers", *IEEE Access*, vol. 7, pp. 4422–4431, 2019.

[327] G. Scrivanti, É. Chouzenoux, and J.-C. Pesquet, "A CNC approach for directional total variation", in *30th European Signal Processing Conference*, IEEE, 2022, pp. 488–492.

[328] J. D. Lee, I. Panageas, G. Piliouras, M. Simchowitz, M. I. Jordan, and B. Recht, "First-order methods almost always avoid strict saddle points", *Mathematical Programming*, vol. 176, pp. 311–337, 2019.

[329] S. Mukherjee, A. Hauptmann, O. Öktem, M. Pereyra, and C.-B. Schönlieb, "Learned reconstruction methods with convergence guarantees: A survey of concepts and applications", *IEEE Signal Processing Magazine*, vol. 40, no. 1, pp. 164–182, 2023.

[330] A. Goujon, S. Neumayer, P. Bohra, S. Ducotterd, and M. Unser, "A neural-network-based convex regularizer for image reconstruction", *IEEE Transactions on Computational Imaging*, vol. 9, pp. 781–795, 2023.

[331] 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, pp. 1256–1272, 2016.

[332] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering", *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[333] A. Goujon, A. Etemadi, and M. Unser, "On the number of regions of piecewise linear neural networks", *Journal of Computational and Applied Mathematics*, vol. 441, p. 115 667, 2024.

[334] Y. E. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k^2)$", *Doklady Akademii Nauk*, vol. 269, no. 3, pp. 543–547, 1983.

[335] B. O'Donoghue and E. Candès, "Adaptive restart for accelerated gradient schemes", *Foundations of Computational Mathematics*, vol. 15, no. 3, pp. 715–732, 2015.

[336] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks", *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2017.

[337] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, "Plug-and-play image restoration with deep denoiser prior", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 6360–6376, 2022.

[338] D. Donoho, "De-noising by soft-thresholding", *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, 1995.

[339] M. Lang, H. Guo, J. Odegard, C. Burrus, and R. Wells, "Noise reduction using an undecimated discrete wavelet transform", *IEEE Signal Processing Letters*, vol. 3, no. 1, pp. 10–12, 1996.

[340] S. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression", *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1532–1546, 2000.

[341] T. Blu and F. Luisier, "The sure-let approach to image denoising", *IEEE Transactions on Image Processing*, vol. 16, no. 11, pp. 2778–2786, 2007.

[342] A. Parekh and I. W. Selesnick, "Convex denoising using non-convex tight frame regularization", *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1786–1790, 2015.

[343] S. Ghadimi and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming", *Mathematical Programming*, vol. 156, no. 1-2, Ser. A, pp. 59–99, 2016.

[344] P. Ochs, Y. Chen, T. Brox, and T. Pock, "IPiano: Inertial proximal algorithm for nonconvex optimization", *SIAM Journal on Imaging Sciences*, vol. 7, no. 2, pp. 1388–1419, 2014.

[345] P. Ochs, "Local convergence of the heavy-ball method and iPiano for non-convex optimization", *Journal of Optimization Theory and Applications*, vol. 177, no. 1, pp. 153–180, 2018.

[346] A. Goujon, S. Neumayer, and M. Unser, "Learning weakly convex regularizers for convergent image-reconstruction algorithms", *SIAM Journal on Imaging Sciences*, 2023, (to appear).

[347] C. De Mol and M. Defrise, "Inverse imaging with mixed penalties", in *Proceedings URSI EMTS 2004*, 2004, pp. 798–800.

[348] J.-L. Starck, M. Elad, and D. Donoho, "Redundant multiscale transforms and their application for morphological component separation", *Advances in Imaging and Electron Physics*, vol. 132, pp. 287–348, Dec. 2004.

[349] M. Elad, J.-L. Starck, P. Querre, and D. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)", *Applied and Computational Harmonic Analysis*, vol. 19, no. 3, pp. 340–358, 2005.

[350] M. Huska, A. Lanza, S. Morigi, and I. Selesnick, "A convex-nonconvex variational method for the additive decomposition of functions on surfaces", *Inverse Problems*, vol. 35, no. 12, p. 124 008, Nov. 2019.

[351] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one", *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490–530, 2005.

[352] D. Van De Ville and M. Kocher, "SURE-based non-local means", *IEEE Signal Processing Letters*, vol. 16, no. 11, pp. 973–976, 2009.

[353] A. Buades, B. Coll, and J.-M. Morel, "Non-local means denoising", *Image Processing On Line*, vol. 1, pp. 208–212, 2011, https://doi.org/10.5201/ipol.2011.bcm_nlm.

[354] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale", in *International Conference on Learning Representations*, 2021.

[355] S. Neumayer, M. Pourya, A. Goujon, and M. Unser, "Boosting weakly convex ridge regularizers with spatial adaptivity", in *NeurIPS 2023 Workshop on Deep Learning and Inverse Problems*, 2023.

[356] Y. Chen, Y. Shi, and B. Zhang, "Optimal control via neural networks: A convex approach", in *International Conference on Learning Representations*, 2019.

ALEXIS GOUJON

alexis.goujon@epfl.ch

EDUCATION

Doctoral studies, Biomedical imaging group, EPFL (*Switzerland*) 2020 – 2023
Thesis: "Towards trustworthy deep learning for image reconstruction"
Supervisor: Prof. Michael Unser

Master of science, Mechanical engineering, University of California, Berkeley (*USA*) 2017 – 2018
Major: Fluid mechanics

Engineering studies, École polytechnique (*France*) 2014 – 2018
Fields of study: Mechanical engineering and applied mathematics

WORK EXPERIENCE

Full-stack software developer, Groupe Ampere (*France*) 2018-2019
Conceived and coded a single-platform software from graphical design to manufacturing of electric wires
Deployed the software and trained users in 4 countries (> 40 daily users)
Software maintenance and continuous upgrade 2020-2023

INTERNSHIPS AND RESEARCH PROJECTS

University of California, Berkeley (*USA*) 2018
Master project
Project: "Bubble tracking under a plunging jet"
Supervisor: Prof. Simo Makiharju

Massachusetts Institute of Technology, Cambridge (*USA*) 2017
Research internship
Project: "Emergent order in hydrodynamic spin lattices"
Supervisor: Prof. John W. Bush

École polytechnique, Palaiseau (*France*) 2016
Master project
Project: "Dynamics of rowing"
Supervisor: Prof. Christophe Clanet

INVAP, San Carlos de Bariloche (*Patagonia, Argentina*) 2016
Intern as engineer in the industry
Topic: Design of a knowledge management platform

TEACHING

Teaching assistant, EPFL (*Switzerland*) 2020-2023
Teaching assistant for Signal and Systems I and II (around 200 students)
Head teaching assistant (4 semesters)

Student project supervision, EPFL (*Switzerland*) 2020-2023

Semester project	"Investigating orthogonal convolutions for denoising" Valérie Costa
Semester project	"Glacial motion" Lucas Mutschler
Master thesis	"Lipschitz-constrained deep spline neural networks" Stanislas Ducotterd
Summer internship	"Number of regions of continuous and piecewise-linear neural networks" Arian Etemadi
Semester project	"Linear regions of continuous and piecewise linear functions" Haojun Zhu

Oral examiner in Mathematics, Lycée Stanislas (*France*) 2016

PUBLICATIONS

Preprints

1. **A. Goujon**, S. Neumayer and M. Unser "Learning weakly convex regularizers for convergent image-reconstruction algorithms", accepted in *SIAM Journal on Imaging Sciences*, 2023.
2. S. Ducotterd, **A. Goujon**, P. Bohra, D. Perdios, S. Neumayer and M. Unser "Improving Lipschitz-constrained neural networks by learning activation functions", submitted to *Journal of Machine Learning Research*.

Journal papers:

1. **A. Goujon**, A. Etemadi and M. Unser "On the number of regions of piecewise linear neural-networks", *Journal of Computational and Applied Mathematics*, volume 441, paper 115667, 2024.
2. **A. Goujon**, S. Neumayer, P. Bohra, S. Ducotterd and M. Unser "A neural-network-based convex regularizer for inverse problems", *IEEE Transactions on Computational Imaging*, volume 9, page 781-795, August 2023.
3. {**A. Goujon**[1], S. Neumayer[1]}, P. Bohra and M. Unser "Approximation of Lipschitz functions using deep spline neural networks", *SIAM Journal on Mathematics of Data Science*, volume 5, page 306-322, issue 2, June 2023.
4. M. Pourya, **A. Goujon** and M. Unser "Delaunay-Triangulation-Based Learning with Hessian Total-Variation Regularization", *IEEE Open Journal of Signal Processing*, volume 4, page 167-178, February 2023.
5. **A. Goujon**, J. Campos and M. Unser "Stable parameterization of continuous and piecewise-linear functions", *Applied and Computational Harmonic Analysis*, volume 67, paper 101581, November 2023.
6. **A. Goujon**, S. Aziznejad, A. Naderi and M. Unser "Shortest-support multi-spline bases for generalized sampling", *Journal of Computational and Applied Mathematics*, volume 395, paper 113610, October 2021.

[1] equal contribution

7. P. Sáenz, G. Pucci, S. Turton, **A. Goujon**, R. Rosales, J. Dunkel and J. Bush "Emergent order in hydrodynamic spin lattices", *Nature*, volume 596, page 58-62, August 2021.
8. P. Sáenz, G. Pucci, **A. Goujon**, T. Cristea-Platon, R. Rosales, J. Dunkel and J. Bush "Spin lattices of walking droplets", *Physical Review Fluids*, volume 3, page 100508, October 2018.

Conferences and Workshops

1. S. Neumayer, M. Pourya, **A. Goujon** and M. Unser, "Boosting Weakly Convex Ridge Regularizers with Spatial Adaptivity", *NeurIPS 2023 Workshop on Deep Learning and Inverse Problems.*
2. P. Bohra, D. Perdios, **A. Goujon**, S. Emery and M. Unser, "Learning Lipschitz-controlled activation functions in neural networks for plug-and-play image reconstruction methods", *NeurIPS 2021 Workshop on Deep Learning and Inverse Problems.*
3. **A. Goujon**, "Shortest Multi-Spline Bases for Generalized Sampling", *Online Seminars on Numerical Approximation and Applications (OSNA2'20)*, Passau (virtual) Germany, 2020.

AWARDS

- Best teaching assistant award, EPFL 2022
- Ocean technology graduate fellowship for outstanding application, 50,000 $, University of California Berkeley 2017
- Gallery of Fluid Motion Award Winner, Denver 2017
- Best research project award in Mechanical Engineering, École polytechnique 2016