Thèse n° 9824

Statistical Inference for Inverse Problems: From Sparsity-Based Methods to Neural Networks

Présentée le 26 avril 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

Pakshal Narendra BOHRA

Acceptée sur proposition du jury

Prof. J.-Ph. Thiran, président du jury
Prof. M. Unser, directeur de thèse
Prof. T. Pock, rapporteur
Prof. M. Pereyra, rapporteur
Prof. F. Krzakala, rapporteur

École
polytechnique
fédérale
de Lausanne

2024

To Dadi, Mummy, Papa, and Ankit.

Abstract

In inverse problems, the task is to reconstruct an unknown signal from its possibly noise-corrupted measurements. Penalized-likelihood-based estimation and Bayesian estimation are two powerful statistical paradigms for the resolution of such problems. They allow one to exploit prior information about the signal as well as handle the noisy measurements in a principled manner. This thesis is dedicated to the development of novel signal-reconstruction methods within these paradigms, ranging from those that involve classical sparsity-based signal models to those that leverage neural networks.

In the first part of the thesis, we focus on sparse signal models in the context of linear inverse problems for one-dimensional (1D) signals. As our first contribution, we devise an algorithm for solving generalized-interpolation problems with L_p-norm regularization. Through a series of experiments, we examine features induced by this regularization, namely, sparsity, regularity, and oscillatory behaviour, which gives us new insight about it. As our second contribution, we present a framework based on 1D sparse stochastic processes to objectively evaluate and compare the performance of signal-reconstruction algorithms. Specifically, we derive efficient Gibbs sampling schemes to compute the minimum mean-square-error estimators for these processes. This allows us to specify a quantitative measure of the degree of optimality for any given method. Our framework also provides access to arbitrarily many training data, thus enabling the benchmarking of neural-network-based approaches.

The second part of the thesis is devoted to neural networks which have become the focus of much of the current research in inverse problems as they typically outperform the classical sparsity-based methods. First, we develop an efficient module for the learning of component-wise continuous piecewise-linear activation functions in neural networks. We deploy this module to train 1-Lipschitz denoising convolutional neural networks and learnable convex regularizers, both of which can be used to design provably convergent iterative reconstruction methods. Next, we design a complete Bayesian inference pipeline for nonlinear inverse problems that leverages the power of deep generative signal models to produce high-quality reconstructions together with uncertainty maps. Finally, we propose a neural-network-based spatiotemporal regularization scheme for dynamic Fourier ptychography (FP), where the goal is to recover a sequence of high-resolution images from several low-resolution intensity measurements. Our approach does not require training data and yields state-of-the-art reconstructions.

Keywords: Inverse problems, statistical inference, Bayesian inference, sparsity, neural networks, activation functions, 1-Lipschitz, learnable regularizers, deep generative models, deep image prior.

Zusammenfassung

Bei inversen Problemen besteht die Aufgabe darin, ein unbekanntes Signal aus seinen möglicherweise durch Rauschen verfälschten Messungen zu rekonstruieren. Die Schätzung auf der Grundlage der bestraften Wahrscheinlichkeit und die Bayes'sche Schätzung sind zwei leistungsstarke statistische Paradigmen für die Lösung solcher Probleme. Sie ermöglichen es, vorherige Informationen über das Signal auszunutzen und die verrauschten Messungen auf prinzipielle Weise zu handhaben. Diese Arbeit widmet sich der Entwicklung neuartiger Signalrekonstruktionsmethoden innerhalb dieser Paradigmen, die von solchen reichen, die klassische, auf Sparsity basierende Signalmodelle beinhalten, bis hin zu solchen, die neuronale Netze nutzen.

Im ersten Teil der Arbeit konzentrieren wir uns auf dünn besetzte Signalmodelle im Kontext linearer inverser Probleme für eindimensionale (1D) Signale. Als unseren ersten Beitrag entwickeln wir einen Algorithmus zur Lösung verallgemeinerter Interpolationsprobleme mit L_p-Norm-Regularisierung. Durch eine Reihe von Experimenten untersuchen wir die Eigenschaften, die durch diese Regularisierung hervorgerufen werden, nämlich Sparsamkeit, Regelmäßigkeit und oszillatorisches Verhalten, was uns neue Einblicke in die Regularisierung ermöglicht. Als unseren zweiten Beitrag stellen wir einen Rahmen vor, der auf spärlichen stochastischen 1D-Prozessen basiert, um die Leistung von Signalrekonstruktionsalgorithmen objektiv zu bewerten und zu vergleichen. Insbesondere leiten wir effiziente Gibbs-Sampling-Schemata ab, um die Schätzer des minimalen mittleren quadratischen Fehlers für diese Prozesse zu berechnen. Dadurch können wir ein quantitatives Maß für den Grad der Optimalität einer jede beliebige Methode angeben. Unser Rahmenwerk bietet auch Zugang zu beliebig vielen Trainingsdaten und ermöglicht so das Benchmarking von auf neuronalen Netzen basierenden Ansätzen.

Der zweite Teil der Arbeit ist neuronalen Netzen gewidmet, die im Mittelpunkt der aktuellen Forschung zu inversen Problemen stehen, da sie in der Regel die klassischen, auf Sparsity basierenden Methoden übertreffen. Zunächst entwickeln wir ein effizientes Modul für das Lernen von komponentenweise kontinuierlichen, stückweise linearen Aktivierungsfunktionen in neuronalen Netzen. Wir setzen dieses Modul ein, um 1-Lipschitz-entrauschende Faltungs-Neuronale Netze und lernbare konvexe Regularisierer zu trainieren, die beide zum Entwerfen nachweislich konvergenter iterativer Rekonstruktionsmethoden verwendet werden können. Als Nächstes entwerfen wir eine vollständige Bayes'sche Inferenzpipeline für nichtlineare inverse Probleme, die die Leistungsfähigkeit tiefer generativer Signalmodelle nutzt, um hochwertige Rekonstruktionen zusammen mit Unsicherheitskarten zu erstellen. Schließlich schlagen wir ein auf neuronalen Netzwerken basierendes räumlich-zeitliches Regularisierungsschema für die dynamische Fourier-Ptychographie (FP) vor, bei der das Ziel darin besteht, eine Sequenz von hochauflösenden Bildern aus mehreren niedrigauflösenden Intensitätsmessungen wiederherzustellen. Unser Ansatz erfordert keine Trainingsdaten und liefert hochmoderne Rekonstruktionen.

Schlüsselwörter: Inverse Probleme, statistische Inferenz, Bayes'sche Inferenz, Sparsity, neuronale Netze, Aktivierungsfunktionen, 1-Lipschitz, lernbare Regularisierer, tiefe generative Modelle, tiefe Modelle für Bildverteilungen.

Contents

List of Figures

List of Tables

1 Introduction

The topic of this thesis is the development of novel statistical methods for solving ill-posed inverse problems. In this introductory chapter, we provide some context for the thesis, followed by a summary of our contributions.

1.1 Background

What are inverse problems?

Put simply, the objective of an *inverse problem* is to determine from observed data, its underlying cause. Such problems are encountered in many fields of science, such as astrophysics, biomedical imaging, geophysics, and optics, to name a few [1–4]. There, a physical quantity of interest, which we also refer to as a signal, is observed only indirectly by performing a series of measurements. The measurement-acquisition process is typically assumed to be known, and the task at hand is then to "invert" this process and recover the signal from the measured data.

To make the above notion concrete, let us consider the example of computed tomography (CT) [5] for medical imaging. During a CT scan, X-rays are directed at the patient from multiple angles. As they pass through the patient's body, they interact with tissues and are absorbed to varying degrees depending on the density of the tissue. The intensities of the attenuated X-rays exiting the patient thus contain some information about the tissues within the body and are recorded by suitably-placed detectors. This acquired data (measurements) then needs to be processed appropriately to reconstruct a three-dimensional map of the internal structures (signal) of the patient, which can be used by healthcare personnel for diagnostic purposes.

As evident in the example of CT, the ability to solve inverse problems is remarkably useful as it provides one with access to physical quantities that cannot be observed directly. Since the mid-twentieth century, aided by the rise of computers, there has been steady progress in the development of efficient numerical methods for solving inverse problems. This has greatly contributed to a better understanding of the physical world by enabling us, for instance, to visualize the structures of biomolecules [6, 7], body tissues [5, 8], as well as celestial objects [9]. Thus, over time, due to its wide-ranging practical implications, the resolution of inverse problems has become a crucial area of scientific research.

Why is it challenging to solve inverse problems?

In several applications, the inverse problem one faces is *ill-posed* in the sense that there exist a multitude of plausible signals that can explain the measured data. For example, in sparse-view CT [10], X-ray measurements are acquired only from a few angles to reduce the patient's radiation exposure. Consequently, they do not contain enough information to uniquely determine the underlying signal (the anatomy of the patient). Thus, for such ill-posed problems, one cannot rely on the direct inversion of the measurement-acquisition process to obtain relevant solutions. Further, in practice, the collected measurements are generally noisy, which adds to the difficulty of the reconstruction task.

How can we solve inverse problems?

The resolution of an ill-posed inverse problem hinges on the use of prior knowledge about the signal of interest. In this thesis, we focus on two well-known statistical paradigms for solving such problems, which allow one to exploit additional information about the signal as well as handle the noise in the measurements in a principled manner.

1. *Penalized-Likelihood-Based Estimation:* In penalized-likelihood-based estimation, the signal is treated as a fixed or deterministic quantity. The cornerstone of this paradigm is the maximum penalized-likelihood (MPL) estimator[1], where the estimate of the signal is (equivalently) specified as the minimizer of a cost functional that consists of a data-fidelity term and a penalty (or regularization) term. The data-fidelity term is derived from a suitably chosen statistical model for the noise in the measurements. It promotes solutions that yield a high likelihood (probability) of observing the measured data, and thus ensures consistency with the measurements. On the other hand, the regularization term imposes some constraints on the solution by penalizing undesirable properties. This cost functional is typically minimized with the help of iterative optimization algorithms.

2. *Bayesian Estimation:* In Bayesian estimation, the signal is assumed to be a realization of a random quantity (for example, a random vector or process) with an appropriate probability distribution that reflects our prior knowledge about

[1]In literature, this is also known as the penalized maximum-likelihood estimator or the regularized maximum-likelihood estimator.

it. The idea there is to characterize the posterior distribution of the signal using statistical models for the measurement noise and signal, and to make inferences based on it. The posterior distribution can be used for the derivation of several point estimators. One such example is the maximum a posteriori (MAP) estimator, which is the mode of the posterior distribution and leads to an optimization problem that resembles the one seen in MPL estimation. Another example is the minimum mean-square-error (MMSE) estimator which turns out to be the posterior mean. Besides the derivation of such point estimators, the Bayesian framework allows one to quantify the uncertainty about the signal. In general, inference tasks entail the computation of expected values with respect to the posterior distribution. Typically, these are high-dimensional integrals that cannot be evaluated analytically. Thus, one relies on sampling algorithms to draw samples from the posterior and then use them to approximate the integrals.

There is a fundamental difference between these paradigms in terms of what the signal model—the regularization term in penalized-likelihood-based estimation and the prior probability distribution in Bayesian estimation—represents. So, although a given MPL estimator can also be interpreted as a MAP estimator for a specific choice of the prior distribution, the two signal models do not necessarily reflect the same information about the underlying signal. Thus, one must be careful while making such interpretations as they can often be misleading [11, 12]. In Chapter 2, we provide a detailed description of the two paradigms, including a discussion about the above-mentioned important distinction.

Practically speaking, the choice of the signal model is mainly driven by the consideration that it should capture the characteristics of the signal of interest while allowing for the deployment of an efficient reconstruction algorithm. For both paradigms, the process of designing such models has undergone a similar transition over time.

Classical Signal Models

Early approaches for solving ill-posed inverse problems were based on quadratic (Tikhonov) regularization terms [13, 14] and Gaussian random processes [15, 16]. Such models impose some smoothness on the estimate of the signal. Their main advantage is that they yield methods that are generally fast, well-understood, and come with performance and stability guarantees. However, during the 1990s, these methods were found to be outperformed by those that take into account sparsity—the property that a signal admits a concise representation in some transform domain (e.g., wavelets) [17].

In MPL estimation, one typically uses an ℓ_1-norm penalty to obtain sparse reconstructions [18–20]. The corresponding optimization problem is non-smooth and is thus solved with the help of sophisticated iterative algorithms [21–24]. A popular example of such models that is widely used in practice is the total-variation regularizer [25, 26], which promotes solutions with sparse gradients. Besides the convex ℓ_1-norm penalty, there has also been some interest in investigating non-convex sparsity-promoting penalties such as the ones based on ℓ_p-quasinorms ($p < 1$) [27] and relaxations of the ℓ_0-pseudonorm [28, 29].

Within the Bayesian paradigm, a standard way of enforcing sparsity is to model the elements of the signal (e.g., pixels, voxels) or its transform-domain coefficients as independent and identically distributed (i.i.d.) random variables with a suitable probability distribution, such as one that exhibits a mass at zero (e.g., Bernoulli-Gaussian-Mixtures [30–33]) or one that is heavy-tailed (e.g., Student's t [34, 35], horseshoe [36]). The resulting posterior distribution is then sampled using an efficient sampling algorithm tailored to the chosen prior distribution.

Neural-Network-Based Signal Models

Over the past few years, researchers have started to deploy neural-network-based methods to solve inverse problems [37, 38]. Such methods have been shown to yield significantly better reconstructions than sparsity-promoting techniques. Broadly speaking, their underlying principle is to utilize large amounts of training data to improve the reconstruction quality, as opposed to the specification of prior information about the signal in the form of "hand-crafted" mathematical models, as in the classical approaches discussed above.

Neural networks (NNs) are powerful learning architectures that are typically constructed via the composition of simple basic modules—linear (or affine) mappings and nonlinear transformations (also called activation functions) [39, 40]. The first successful applications of NNs in signal recovery involve training the network as a nonlinear mapping that relates a low-quality estimate of the signal to the desired high-quality estimate [41–43]. The reconstruction pipeline then consists of using a fast classical algorithm to obtain an initial solution and then correcting for its artifacts using the trained network. This category of methods includes unrolling [44–49], where the architecture of the network is designed by studying iterations of algorithms used for computing the MPL estimator. There also exist analogues of such approaches that involve training the network to approximate a Bayesian estimator or even directly generate samples from the posterior distribution [50]. While these end-to-end learning methods have achieved state-of-the-art performances in several inverse problems, they suffer from the limitation of not being "flexible". The networks in such methods are trained on large datasets consisting of signals and their measurements and are thus highly sensitive to the corresponding measurement-acquisition setup. In this thesis, we will instead mainly focus on more versatile or "universal" NN-based methods, where a network that has been pretrained to model only the prior knowledge about the signal (in a generic way that does not depend on the inverse problem at hand) is applied within the penalized-likelihood-based estimation or Bayesian estimation paradigm.

The plug-and-play priors (PnP) [51] and regularization-by-denoising (RED) [52, 53] frameworks are two successful examples of integrating NNs into the penalized-likelihood-based estimation paradigm. In PnP (RED) methods, the idea is to replace the proximal (gradient) operator of the regularization term that appears in the iterations of the proximal (gradient) algorithms used for MPL estimation by an off-the-shelf denoiser (the residual of an off-the-shelf denoiser). Generally, this denoiser only plays the role of an implicit regularizer, that is, there is no explicit penalty term associated with it. Nonetheless, the fixed-point convergence of such iterative schemes can be ensured if the denoiser belongs to a suitable class of 1-Lipschitz operators [54, 55]. Learning-based variants of these frameworks involve the use of a denoiser that is constructed from an appropriately trained NN [56–62]. However, in order to ensure convergence, the network must be constrained such that the denoiser belongs to the desired class of 1-Lipschitz operators. This is a challenging task and remains an active area of research [55, 63]. More recently, gradient-step NN denoisers have been used to devise PnP and RED methods that actually minimize an explicit global cost functional [64–66]. Outside of these frameworks, NNs have also been deployed for designing an explicit learnable general-purpose regularization term [67]. There also exists another class of methods that involves deep generative models such as variational autoencoders (VAEs) [68] and generative adversarial networks (GANs) [69]. These models include a generator network that maps a low-dimensional latent space to the high-dimensional signal space. They are trained on a dataset of signals such that they capture its statistics and generate sample signals similar to those in the dataset. Once such a deep generative model has been successfully trained, its application to an inverse problem consists of finding a signal in the range of the generator that is consistent with the given measurements. One way of performing this task is to formulate a suitable estimator in the latent space [70–72]. Most of the NN-based signal models described above can also be utilized in the context of Bayesian estimation. Specifically, efficient customized posterior sampling schemes have been developed for prior probability distributions encoded by denoising NNs (such as the ones used in the PnP or RED frameworks) [73–75], GANs [76], VAEs [77, 78], score-based generative models [79, 80], and energy-based generative models [81].

So far, we have only discussed NN-based methods that require training data. Remarkably, it is also possible to define a signal model using an untrained NN. In such methods [82–84], the signal of interest is represented as the output of a network corresponding to some fixed input. The parameters of the network are then estimated such that the generated signal is in agreement with the acquired measurements. This is typically done by minimizing an appropriate data-fidelity term. In some scenarios, such schemes are deployed with early stopping as deep networks have the capacity to fit noise. Alternately, one can consider MPL estimation or Bayesian estimation for the network parameters with simple models such as ℓ_2-norm regularization or Gaussian priors [84]. The success of these methods involving untrained NNs is attributed to the implicit signal model imposed by the architecture of the network, which favours natural-looking signals (“good” solutions) over noisy ones (“bad” solutions).

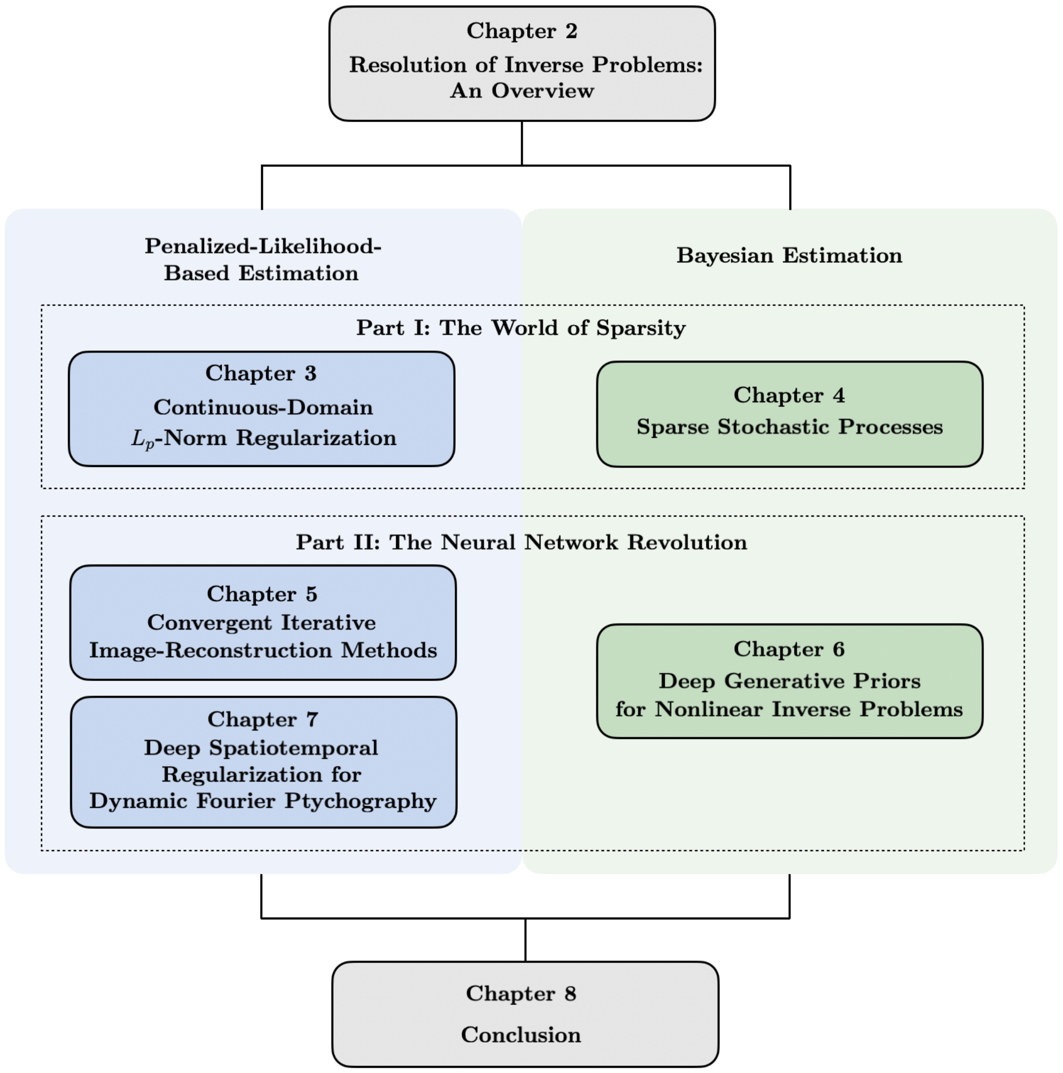

Chapter 2
Resolution of Inverse Problems: An Overview
Penalized-Likelihood-Based Estimation
Bayesian Estimation
Part I: The World of Sparsity
Chapter 3
Continuous-Domain L_p-Norm Regularization
Chapter 4
Sparse Stochastic Processes
Part II: The Neural Network Revolution
Chapter 5
Convergent Iterative Image-Reconstruction Methods
Chapter 7
Deep Spatiotemporal Regularization for Dynamic Fourier Ptychography
Chapter 6
Deep Generative Priors for Nonlinear Inverse Problems
Chapter 8
Conclusion

Figure 1.1: Roadmap of the thesis.

1.2 Contributions

This thesis is dedicated to the development of new signal-reconstruction methods within the penalized-likelihood-based estimation and Bayesian estimation paradigms, ranging from those that involve sparsity-based models to those that leverage neural networks. The roadmap of the thesis is shown in Figure 1.1. Next, we present a summary of our contributions along with a list of the relevant publications.

Part I: The World of Sparsity

In the first part of the thesis, we visit *the world of sparsity* in the context of linear inverse problems for one-dimensional (1D) signals.

Continuous-Domain L_p-Norm Regularization (Chapter 3)

Most real-world inverse problems are concerned with the recovery of a continuous-domain signal. The typical pipeline for tackling such problems first involves formulating them in terms of a discrete representation of the underlying signal. Prior information about the signal is then introduced through a model for its discrete representation. Alternately, one can also directly specify the estimation task and signal model in the continuum (provided that a solution can be computed analytically or numerically). In this chapter, we seek to understand the effect of one such model for 1D signals—continuous-domain L_p-norm regularization for $p \geq 1$ and with a multi-order derivative regularization operator D^{N_0}. To that end, we develop a numerical method to solve the L_p-regularized generalized-interpolation problem. Through a series of experiments, we then identify properties of this regularization.

Specifically, we formulate our reconstruction problem as the task of finding a 1D continuous-domain signal that minimizes the L_p-norm regularization term subject to some strict data constraints (generalized-interpolation problem). We cast this problem exactly as a finite-dimensional one by restricting the search space to a suitable space of polynomial splines with knots on a uniform grid. Our splines are represented in a B-spline basis, which results in a well-conditioned discretization. For a sufficiently fine grid, our search space contains functions that are arbitrarily close to the solution of the underlying problem where our constraint that the solution must live in a spline space would have been lifted. This remarkable property is due to the approximation power of splines. We use the alternating-direction method of multipliers along with a multiresolution strategy to compute our solution. Through our numerical experiments for spatial and Fourier interpolation, we examine features induced by L_p-norm regularization, namely, sparsity, regularity (smoothness) and, oscillatory behaviour and overshoot.

Related publication
P. Bohra and M. Unser, "Continuous-Domain Signal Reconstruction Using L_p-Norm Regularization", *IEEE Transactions on Signal Processing*, vol. 68, pp. 4543-4554, 2020.

Sparse Stochastic Processes (Chapter 4)

We present a benchmarking environment based on sparse stochastic processes to objectively evaluate and compare the performance of reconstruction algorithms for linear inverse problems involving 1D signals. Our framework offers quantitative measures of the degree of optimality (in the mean-square-error sense) for any given reconstruction method. Since it is based on stochastic modelling, it provides access to unlimited amounts of data, which enables the proper benchmarking of NN-based approaches without having to worry about the representativity of the training data.

In our framework, we generate synthetic signals as realizations of 1D sparse stochastic processes. We derive Gibbs sampling schemes to compute the minimum mean-square error estimators for processes with Laplace, Student's t, and Bernoulli-Laplace innovations. These allow us to provide statistical guarantees of optimality by specifying an upper limit on the reconstruction performance. We showcase our framework by benchmarking the performance of some well-known classical MPL estimators (such as the total-variation-regularized method) and convolutional neural network architectures that perform direct nonlinear reconstructions in the context of deconvolution and Fourier sampling. Our experimental results support the understanding that, while these neural networks outperform the sparsity-based MPL estimators and achieve near-optimal results in many settings, their performance deteriorates severely for signals associated with heavy-tailed distributions.

Related publication
P. Bohra, P. del Aguila Pla, J. -F. Giovannelli, and M. Unser, "A Statistical Framework To Investigate the Optimality of Signal-Reconstruction Methods", *IEEE Transactions on Signal Processing*, vol. 71, pp. 2043-2055, 2023.

Part II: The Neural Network Revolution

The second part of the thesis is driven by *the neural network revolution* in the field of inverse problems. In particular, we investigate the integration of neural networks into the penalized-likelihood-based estimation and Bayesian estimation paradigms for image reconstruction.

Convergent Iterative Image-Reconstruction Methods (Chapter 5)

In this chapter, we first present an efficient module for learning continuous piecewise-linear activation functions in neural networks. We then deploy this module to train 1-Lipschitz denoising convolutional neural networks and learnable convex regularizers, both of which can be used to design provably convergent iterative image-reconstruction methods. The details of these contributions are provided in what follows.

1. Learning Activation Functions in Neural Networks
We develop an efficient computational solution to train neural networks with free-form component-wise activation functions. To make the problem well-posed, we augment the cost functional of the neural network by adding an appropriate shape regularization: the sum of the second-order total-variations of the trainable nonlinearities. The representer theorem for neural networks tells us that the optimal activation functions are adaptive piecewise-linear splines, which allows us to recast the problem as a parametric optimization. The challenging point is that the corresponding basis functions (ReLUs) are poorly conditioned and that the determination of their number and positioning is also part of the problem. We circumvent the difficulty by using an equivalent B-spline basis to encode the

activation functions and by expressing the regularization as an ℓ_1-penalty. This results in the specification of parametric activation function modules that can be implemented and optimized efficiently on standard development platforms. We present experimental results that demonstrate the benefit of our approach.

2. Lipschitz-Constrained Neural Networks for Plug-and-Play Reconstruction
Within the PnP framework, one can use denoisers based on 1-Lipschitz NNs to design convergent iterative reconstruction schemes. Since Lipschitz-constrained ReLU networks have provable disadvantages, we instead consider the use of learnable 1-Lipschitz linear spline activation functions. We propose an efficient method that utilizes our B-spline module to train these neural networks. Our numerical experiments, which include denoising and CT and MRI reconstruction, show that our trained networks compare favorably with existing 1-Lipschitz neural architectures.

3. A Neural-Network-Based Convex Regularizer
Finally, we present a framework to learn a regularization term that is the sum of convex-ridge functions. We use a one-hidden-layer neural network with learnable increasing linear spline activation functions, which are again implemented using our B-spline module, to parametrize the gradient of the regularizer. This network is trained within a few minutes as a multistep Gaussian denoiser. Through numerical experiments for denoising and CT and MRI reconstruction, we show that our method outperforms others that offer similar reliability guarantees.

Related publications

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.

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.*

S. Ducotterd, A. Goujon, P. Bohra, D. Perdios, S. Neumayer, and M. Unser, "Improving Lipschitz-Constrained Neural Networks by Learning Activation Functions", *arXiv preprint arXiv:2210.16222*, 2022.

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*, vol. 9, pp. 781-795, 2023.

Deep Generative Priors for Nonlinear Inverse Problems (Chapter 6)

In this chapter, we develop a Bayesian inference pipeline that leverages the power of deep generative models as image priors to produce high quality reconstructions together with uncertainty maps. To the best of our knowledge, this is one of the first deployments of such techniques for the resolution of nonlinear inverse problems.

Specifically, we present a Bayesian reconstruction framework for nonlinear inverse problems where we specify the prior information about the image through a deep latent variable generative model such as a GAN or a VAE. We develop a tractable posterior-sampling scheme based on the Metropolis-adjusted Langevin algorithm for the class of nonlinear inverse problems where the forward model has a neural-network-like structure. This class includes most practical imaging modalities. We also introduce the notion of augmented deep generative priors in order to suitably handle the recovery of quantitative images. We illustrate the advantages of our framework by applying it to two nonlinear imaging modalities—phase retrieval and optical diffraction tomography.

Related publication

P. Bohra, T. -a. Pham, J. Dong, and M. Unser, "Bayesian Inversion for Nonlinear Imaging Models Using Deep Generative Priors", *IEEE Transactions on Computational Imaging*, vol. 8, pp. 1237-1249, 2022.

Deep Spatiotemporal Regularization for Dynamic Fourier Ptychography (Chapter 7)

In our last contribution, we explore the use of an untrained neural network as an implicit regularizer in the context of Fourier ptychography (FP). This modality involves the acquisition of several low-resolution intensity images of a sample under varying illumination angles. They are then combined into a high-resolution complex-valued image by solving a phase-retrieval problem. The objective in dynamic FP is to obtain a sequence of high-resolution images of a moving sample. There, the application of standard frame-by-frame reconstruction methods limits the temporal resolution due to the large number of measurements that must be acquired for each frame. We instead propose a neural-network-based reconstruction framework for dynamic FP, which achieves high temporal resolution without compromising the spatial resolution. It does not require training data and also recovers the pupil function of the microscope.

Specifically, in our framework, each image in the sequence is represented as the output of a shared deep convolutional network fed with an input vector that lies on a one-dimensional manifold that encodes time. The parameters of the network and the pupil function of the microscope, which is represented using Zernike polynomials, are then estimated by optimizing a likelihood-based criterion. Here, the architecture of the network and the constraints on the input vectors impose a spatiotemporal regularization on the sequence of images. Through numerical experiments, we show that our framework drastically improves the quality of reconstruction over standard frame-by-frame methods and thus paves the way for high-quality ultrafast FP.

Related publication

P. Bohra, T. -a. Pham, Y. Long, J. Yoo, and M. Unser, “Dynamic Fourier Ptychography With Deep Spatiotemporal Priors”, *Inverse Problems*, vol. 39, no. 6, paper no. 064005, 2023.

2 Resolution of Inverse Problems: An Overview

In this chapter, we set the scene for the thesis by presenting a mathematical formulation of inverse problems and by briefly describing two statistical reconstruction paradigms—penalized-likelihood-based estimation and Bayesian estimation—for their resolution.

2.1 Inverse Problems

2.1.1 Continuous-Domain Formulation

The goal in an inverse problem is to recover an unknown signal s_0 from a collection of its possibly noisy measurements $\mathbf{y} \in \mathbb{R}^M$. Since most real-world signals are analog in nature, we consider the signal of interest $s_0 : \mathbb{R}^d \to \mathbb{R}$ to be a d-dimensional continuous-domain function. We model the measurements as

$$\mathbf{y} = \mathbf{N}\Big(\boldsymbol{\nu}(s_0)\Big), \tag{2.1}$$

where $\boldsymbol{\nu} : s \mapsto \boldsymbol{\nu}(s) \in \mathbb{R}^M$ is the (linear or nonlinear) forward operator that describes the physics of the acquisition process and $\mathbf{N} : \mathbb{R}^M \to \mathbb{R}^M$ is an operator that models the corruption of measurements by noise. Here, we have assumed that the signal and measurements are real-valued in order to simplify the exposition. However, the inverse problem formulation and reconstruction methods presented in this chapter can be easily extended to handle complex-valued signals and measurements.

2.1.2 Discretization

The first step towards solving an inverse problem is the discretization of the signal s_0 and the forward operator $\boldsymbol{\nu}$ as this allows us to perform computations digitally. We can then write the discrete measurement model as

$$\mathbf{y} = \mathbf{N}\Big(\mathbf{H}(\mathbf{s}_0)\Big), \tag{2.2}$$

where $\mathbf{s}_0 \in \mathbb{R}^K$ is the finite-dimensional discrete representation of s_0 (the typical choice is a vector containing samples of s_0 within some region of interest) and $\mathbf{H}$[1] $: \mathbb{R}^K \to \mathbb{R}^M$ is the discrete counterpart of $\boldsymbol{\nu}$. Ideally, the discretization of $\boldsymbol{\nu}$ is performed such that $\mathbf{H}$ only yields a small discretization error (if any), that is, $\boldsymbol{\nu}(s_0) \approx \mathbf{H}(\mathbf{s}_0)$, while being computation-friendly, that is, it can be evaluated efficiently in terms of computation time and memory. We will present some concrete examples of discretization schemes in Chapters 3, 4 and 7. For the remainder of this chapter, it is assumed that s_0 and $\boldsymbol{\nu}$ have been discretized appropriately.

2.1.3 Ill-Posedness

So, the task at hand now is to recover $\mathbf{s}_0$ from $\mathbf{y}$ by “inverting” the measurement model in (2.2). Besides the presence of noise in the measurements, which can already make the reconstruction task difficult, most practical inverse problems are ill-posed in the sense that the same set of measurements can be generated by multiple signals. Therefore, prior knowledge about the signal of interest is required for the resolution of such problems.

2.2 Statistical Reconstruction Paradigms

Next, we describe two well-known statistical reconstruction paradigms for solving ill-posed inverse problems. They enable one to incorporate prior information about the signal and handle the noise in the measurements in a systematic way.

2.2.1 Penalized-Likelihood-Based Estimation

In this paradigm, the signal of interest $\mathbf{s}_0$ is treated as a *fixed* or *deterministic* quantity. The main idea here is to specify the reconstructed signal as the solution of an optimization problem that balances a data-fidelity term, which is based on a statistical model for the noisy measurements, and a penalty (regularization) term, which imposes some favourable properties on it.

Likelihood Function

In order to account for the nonideality of the measurement-acquisition setup, we consider a statistical model that relates the noisy measurements $\mathbf{y}$ and the signal $\mathbf{s}_0$. Specifically,

[1]When $\mathbf{H}$ is a linear operator, by abuse of notation, we will also use the symbol $\mathbf{H}$ to denote its matrix representation. Thus, in such cases, the quantity $\mathbf{H}(\mathbf{s})$ will be written as $\mathbf{H}\mathbf{s}$.

we assume that the operator $\mathbf{N}$ in (2.2) generates $\mathbf{y}$ as a realization of a random vector $\mathbf{Y}$ that is distributed according to

$$\mathbf{Y} \sim p_{\text{noise}}\Big(\cdot\ ;\ \boldsymbol{\phi} = \mathbf{H}(\mathbf{s}_0)\Big), \tag{2.3}$$

where the probability density function (pdf) p_{noise} models the statistics of the noise in the acquisition system and $\boldsymbol{\phi}$ denotes (some of) its parameters.

Many practical setups involve multiple independent sources of noise. Thus, it is reasonable to assume an additive white-Gaussian-noise (AWGN) model, as dictated by the central limit theorem. For an AWGN model with variance σ^2, we can write (2.2) as

$$\mathbf{y} = \mathbf{H}(\mathbf{s}_0) + \mathbf{n}, \tag{2.4}$$

where $\mathbf{n} \in \mathbb{R}^M$ is a realization of a Gaussian random vector consisting of i.i.d.entries with zero mean and variance σ^2. In this case, the pdf p_{noise} is given by

$$p_{\text{noise}}(\cdot\ ;\ \boldsymbol{\phi}) = p_{\text{Gaussian}}(\cdot\ ;\ \boldsymbol{\phi}) = \frac{1}{(2\pi\sigma^2)^{M/2}} \exp\left(- \frac{\|\cdot - \boldsymbol{\phi}\|_2^2}{2\sigma^2}\right). \tag{2.5}$$

Another model that is commonly used in practice is the shot- or Poisson-noise model. In this case, we have that

$$p_{\text{noise}}(\cdot\ ;\ \boldsymbol{\phi}) = p_{\text{Poisson}}(\cdot\ ;\ \boldsymbol{\phi}) = \prod_{m=1}^{M} \frac{([\boldsymbol{\phi}]_m)^{[\cdot]_m}}{([\cdot]_m)!} \exp\Big(- [\boldsymbol{\phi}]_m\Big). \tag{2.6}$$

Based on the statistical model for the noisy measurements in (2.3), the likelihood function is defined as

$$\mathcal{L}(\cdot\ ;\ \mathbf{y}) \coloneqq p_{\text{noise}}\Big(\mathbf{y}\ ;\ \boldsymbol{\phi} = \mathbf{H}(\cdot)\Big). \tag{2.7}$$

Maximum-Penalized-Likelihood Estimator

The maximum-penalized-likelihood (MPL) estimator for the signal is an extension of the classical maximum-likelihood (ML) estimator. It is specified as

$$\begin{aligned} \mathbf{s}^*_{\text{MPL}}(\mathbf{y}) &\in \arg\max_{\mathbf{s}\in\mathbb{R}^K} \left(\log\Big(\mathcal{L}(\mathbf{s};\mathbf{y})\Big) - \tau\mathcal{R}(\mathbf{s})\right) \\ &\in \arg\min_{\mathbf{s}\in\mathbb{R}^K} \left(\underbrace{-\log\Big(p_{\text{noise}}\big(\mathbf{y};\boldsymbol{\phi} = \mathbf{H}(\mathbf{s})\big)\Big)}_{\mathcal{D}\left(\mathbf{y},\ \mathbf{H}(\mathbf{s})\right)} + \tau\mathcal{R}(\mathbf{s})\right), \end{aligned} \tag{2.8}$$

where the data-fidelity term $\mathcal{D} : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}_+$ ensures consistency with the measurements by promoting solutions that yield a high likelihood of observing the measured data, the penalty (or regularization) term $\mathcal{R} : \mathbb{R}^K \to \mathbb{R}_+$ reflects our prior knowledge on the signal by penalizing solutions with undesirable properties, and $\tau \in \mathbb{R}_+$ is a tunable hyperparameter that controls the strength of the regularization.

Remark: The family of MPL estimators constitutes a subclass of the so-called "variational" methods that are well-known in the inverse problems community. In a generic variational reconstruction method, the data-fidelity term is not necessarily derived from a statistical model for the noise.

The cost functional in (2.8) is typically minimized in an iterative manner using gradient or proximal methods [85, 86]. Here, we present two simple examples of these methods—the gradient-descent (GD) and forward-backward splitting (FBS) [87] algorithms—that are applicable when the data-fidelity term $\mathcal{D}$ is differentiable. The vanilla GD algorithm can be used to compute the solution when the regularization term $\mathcal{R}$ is also differentiable. The iterations for GD are given by

$$\mathbf{s}_{k+1} = \mathbf{s}_k - \gamma\Big(\boldsymbol{\nabla}\mathcal{D}\Big(\mathbf{y}, \mathbf{H}(\mathbf{s}_k)\Big) + \tau\boldsymbol{\nabla}\mathcal{R}(\mathbf{s}_k)\Big), \tag{2.9}$$

where $\gamma > 0$ is a suitably chosen step-size. On the other hand, when $\mathcal{R}$ is non-differentiable, one can solve the optimization task with the help of a proximal method such as FBS. At each iteration in FBS, the estimate is updated as

$$\mathbf{s}_{k+1} = \mathrm{prox}_{\gamma\tau\mathcal{R}}\Big(\mathbf{s}_k - \gamma\boldsymbol{\nabla}\mathcal{D}\Big(\mathbf{y}, \mathbf{H}(\mathbf{s}_k)\Big)\Big), \tag{2.10}$$

where $\gamma > 0$ is an appropriate step-size and the proximal operator of a function $g : \mathbb{R}^K \to \mathbb{R}$ is defined as

$$\mathrm{prox}_g(\cdot) = \underset{\mathbf{u}\in\mathbb{R}^K}{\arg\min}\left(\frac{1}{2}\|\cdot - \mathbf{u}\|_2^2 + g(\mathbf{u})\right). \tag{2.11}$$

Note that, in general, the convergence of these routines to a global minimum is guaranteed only when $\mathcal{D}$ and $\mathcal{R}$ are convex functions. When the cost functional in (2.8) is non-convex, we only expect the deployed optimization algorithm to find one of the stationary points.

2.2.2 Bayesian Estimation

In the Bayesian paradigm, the signal $\mathbf{s}_0$ is assumed to be a realization of a *random* vector S with a probability distribution p_{S} that captures our prior knowledge about it. The idea here is to make inferences about the signal based on its posterior distribution, which is characterized using a statistical model for the measurement noise and p_{S}.

Likelihood Function

We model the noisy data measured by the acquisition system as a random vector Y that is related to the random vector S via the conditional distribution

$$p_{\mathsf{Y}|\mathsf{S}}(\cdot \mid \mathbf{s}) = p_{\text{noise}}\Big(\cdot\ ;\ \boldsymbol{\phi} = \mathbf{H}(\mathbf{s})\Big), \tag{2.12}$$

where $\mathbf{s} \in \mathbb{R}^K$ and, similar to what we have in the penalized-likelihood-based estimation paradigm, p_{noise} is a pdf that accounts for noise in the acquisition system and $\boldsymbol{\phi}$ denotes some or all of its parameters (please see Equations (2.5) and (2.6) for examples of p_{noise}). Under this statistical model, the observed measurements $\mathbf{y}$ can be interpreted as a realization of the random vector $\mathsf{Y}|\mathsf{S} = \mathbf{s}_0$. Here, the likelihood function is given by

$$\mathcal{L}(\cdot \mid \mathbf{y}) = p_{\mathsf{Y}|\mathsf{S}}(\mathbf{y} \mid \cdot) = p_{\text{noise}}\Big(\mathbf{y}\ ;\ \boldsymbol{\phi} = \mathbf{H}(\cdot)\Big). \tag{2.13}$$

Note that this function is equal to the one shown in (2.7); it has just been specified under a different formalism.

Posterior Distribution

In Bayesian estimation, the quantity of interest is the posterior distribution of the random vector $\mathsf{S}|\mathsf{Y} = \mathbf{y}$ as it provides a complete statistical characterization of the inverse problem. Using Bayes' theorem, its pdf is written as

$$p_{\mathsf{S}|\mathsf{Y}}(\cdot|\mathbf{y}) = \frac{p_{\mathsf{Y}|\mathsf{S}}(\mathbf{y}|\cdot)p_{\mathsf{S}}(\cdot)}{\int_{\mathbb{R}^K} p_{\mathsf{Y}|\mathsf{S}}(\mathbf{y}|\tilde{\mathbf{s}})p_{\mathsf{S}}(\tilde{\mathbf{s}})\mathrm{d}\tilde{\mathbf{s}}}. \tag{2.14}$$

The posterior distribution $p_{\mathsf{S}|\mathsf{Y}}(\cdot|\mathbf{y})$ can be used for the derivation of various point estimators for the signal $\mathbf{s}_0$. Two examples of such estimators that are commonly used in practice are the maximum a posteriori (MAP) estimator and the minimum mean-square-error (MMSE) estimator. The MAP estimator calculates the mode of $p_{\mathsf{S}|\mathsf{Y}}(\cdot|\mathbf{y})$ and is given by

$$\begin{aligned}\mathbf{s}^*_{\text{MAP}}(\mathbf{y}) &= \underset{\mathbf{s}\in\mathbb{R}^K}{\arg\max}\ p_{\mathsf{S}|\mathsf{Y}}(\mathbf{s}|\mathbf{y})\\ &= \underset{\mathbf{s}\in\mathbb{R}^K}{\arg\max}\ \left(\log\Big(p_{\mathsf{Y}|\mathsf{S}}(\mathbf{y}|\mathbf{s})\Big) + \log\Big(p_{\mathsf{S}}(\mathbf{s})\Big)\right)\\ &= \underset{\mathbf{s}\in\mathbb{R}^K}{\arg\min}\ \left(-\log\Big(p_{\text{noise}}\big(\mathbf{y};\boldsymbol{\phi} = \mathbf{H}(\mathbf{s})\big)\Big) - \log\Big(p_{\mathsf{S}}(\mathbf{s})\Big)\right).\end{aligned} \tag{2.15}$$

The primary reason for the popularity of MAP estimators is that they can be computed efficiently using the iterative gradient or proximal algorithms mentioned earlier. It is noteworthy that the optimization problem in (2.15) closely resembles the one formulated in MPL estimation (see Equation (2.8)). We will discuss this link between MPL and

MAP estimation in detail later in Section 2.2.4. On the other hand, the MMSE estimator is given by

$$\begin{aligned}\mathbf{s}^*_{\text{MMSE}}(\mathbf{y}) &= \arg\min_{\mathbf{s}\in\mathbb{R}^K}\left(\int_{\mathbb{R}^K} \|\tilde{\mathbf{s}}-\mathbf{s}\|_2^2\; p_{\mathsf{S}|\mathsf{Y}}(\tilde{\mathbf{s}}|\mathbf{y})\mathrm{d}\tilde{\mathbf{s}}\right)\\ &= \int_{\mathbb{R}^K} \tilde{\mathbf{s}}\; p_{\mathsf{S}|\mathsf{Y}}(\tilde{\mathbf{s}}|\mathbf{y})\mathrm{d}\tilde{\mathbf{s}}, \end{aligned} \tag{2.16}$$

which is the mean of the posterior distribution $p_{\mathsf{S}|\mathsf{Y}}(\cdot|\mathbf{y})$. In general, computing the MMSE estimator is challenging as it requires one to numerically evaluate the high-dimensional integral in (2.16).

Besides the derivation of point estimators, the Bayesian paradigm allows one to perform advanced inferences such as uncertainty quantification (for example, computing higher-order moments of $p_{\mathsf{S}|\mathsf{Y}}(\cdot|\mathbf{y})$ or specifying credible regions that indicate where most of the mass of the posterior distribution is concentrated) and model selection. Typically, performing such inferences poses the same challenge as in MMSE estimation as it involves calculating integrals of the form

$$\mathrm{I}_f(\mathbf{y}) = \int_{\mathbb{R}^K} f(\tilde{\mathbf{s}})p_{\mathsf{S}|\mathsf{Y}}(\tilde{\mathbf{s}}|\mathbf{y})\mathrm{d}\tilde{\mathbf{s}}, \tag{2.17}$$

where $f:\mathbb{R}^K \to \mathbb{R}$ is a real-valued function.

The high dimensionality of the integral in (2.17) (of which (2.16) is a special case) makes its approximation by simple techniques such as uniform-grid-based Riemann sums infeasible. Instead, one can rely on stochastic simulation techniques such as Markov Chain Monte Carlo (MCMC) methods [88–91] for the numerical approximation of (2.17) in a tractable manner. MCMC methods are designed for generating random samples from nontrivial high-dimensional probability distributions. Broadly speaking, the idea in MCMC is to construct a Markov chain such that the distribution that one wishes to draw samples from is its stationary distribution. The desired samples can be obtained by simulating the Markov chain and recording its states after a sufficient period of time (assuming the chain converges theoretically [92]). Thus, in order to compute the integral in (2.17), one first generates samples $\{\mathbf{s}^{(q)}\}_{q=1}^Q$ from $p_{\mathsf{S}|\mathsf{Y}}(\cdot|\mathbf{y})$ using an MCMC method and then approximates $\mathrm{I}_f(\mathbf{y})$ by its empirical estimate $\frac{1}{Q}\sum_{q=1}^Q f(\mathbf{s}^{(q)})$. Later in the thesis, we will detail two MCMC algorithms—Gibbs sampling (Chapter 4) and the Metropolis Adjusted Langevin algorithm (Chapter 6).

2.2.3 Signal Models

We now discuss some signal models—the regularization term in penalized-likelihood-based estimation and the prior probability distribution in Bayesian estimation—that have been proposed in the literature, ranging from classical ones to recent neural-network-based ones.

Classical Models

Tikhonov regularization of the form $\mathcal{R}(\mathbf{s}) = \|\mathbf{L}\mathbf{s}\|_2^2$ [13, 14], where $\mathbf{L}$ is a linear transformation such as the discrete version of the gradient operator, is one of the classical choices for the penalty term in (2.8). It imposes a constraint on the energy of the transformed signal $\mathbf{L}\mathbf{s}$ and thus promotes solutions with some degree of smoothness. In particular, the use of such a quadratic penalty has been extensively studied for linear inverse problems with an AWGN model. There, it leads to solutions of the form

$$\mathbf{s}^*_{\mathrm{MPL},\ell_2}(\mathbf{y}) = \Big(\mathbf{H}^T\mathbf{H} + \sigma^2\tau\mathbf{L}^T\mathbf{L}\Big)^{-1}\mathbf{H}^T\mathbf{y}, \tag{2.18}$$

where σ is the standard deviation of the Gaussian noise, thereby yielding linear reconstruction methods that are fast, well-understood, and equipped with stability guarantees with respect to perturbations in the measurements.

The Gaussian prior distribution

$$p_{\mathbf{S}}(\mathbf{s}) = \frac{1}{\sqrt{(2\pi)^K|\det(\mathbf{C})|}}\exp\bigg(-\frac{1}{2}(\mathbf{s}-\mathbf{m})^T\mathbf{C}^{-1}(\mathbf{s}-\mathbf{m})\bigg), \tag{2.19}$$

where $\mathbf{s} \in \mathbb{R}^K$, $\mathbf{m} \in \mathbb{R}^K$ is the mean of the distribution and $\mathbf{C} \in \mathbb{R}^{K\times K}$ is the covariance matrix of the distribution, is the Bayesian counterpart of Tikhonov regularization. It also yields estimates of the signal that exhibit some form of smoothness. Interestingly, for linear inverse problems with an AWGN model, the MAP and MMSE estimates corresponding to this Gaussian prior turn out to be equivalent. They are given by the reconstruction scheme

$$\mathbf{s}^*_{\mathrm{Gaussian}}(\mathbf{y}) = \mathbf{m} + \Big(\mathbf{H}^T\mathbf{H} + \sigma^2\mathbf{C}^{-1}\Big)^{-1}\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{m}), \tag{2.20}$$

where σ^2 is the variance of the Gaussian noise.

Another popular category of classical regularization schemes involves the use of penalty terms based on sparsity—the property that a signal can be represented in some transform domain (e.g., wavelets) with only a few parameters [17]—which typically lead to better reconstructions than their quadratic counterparts. This powerful concept of sparsity is at the heart of the theory of compressed sensing, which gives conditions under which the recovery of a signal from a limited set of its linear measurements is feasible [93–96] and stable [97, 98]. One typically uses ℓ_1-norm regularization of the form $\mathcal{R}(\mathbf{s}) = \|\mathbf{L}\mathbf{s}\|_1$ [18–20] to enforce sparsity in the transform domain specified by $\mathbf{L}$ (e.g., wavelet transform or gradient operator). Since the ℓ_1-norm is non-differentiable, the corresponding optimization problem is often solved using proximal algorithms such as FBS and its variants [21, 23], or the alternating direction method of multipliers (ADMM) [24]. Besides the convex ℓ_1-norm penalty, one can also use non-convex penalties based on ℓ_p-quasinorms ($p < 1$) [27] and relaxations of the ℓ_0-pseudonorm [28, 29] to obtain sparse(r) reconstructions.

In Bayesian estimation, to enforce a sparse representation of the signal in a linear transform domain $\mathbf{L}$ (e.g., wavelets), one typically models the random vector $\mathsf{U} = \mathsf{LS}$ to have i.i.d. entries with a suitable "sparse" pdf p_{U}. Examples of such pdfs include those with a mass at the origin (e.g., Bernoull-Gaussian-Mixtures [30–33]) and those that exhibit heavy tails (e.g., Student's t [34, 35], horseshoe [36]). For these sparsity-based priors, the resulting posterior distribution is sampled using tailored MCMC methods. In Chapter 4, we will study in detail prior distributions corresponding to the family of infinitely divisible pdfs (for p_{U}).

Neural-Network-Based Models

Neural-network-based methods, having been found to outperform the sparsity-promoting methods discussed above, are now the focus of much of the research in signal reconstruction [37, 38]. Broadly speaking, their principle is to exploit prior information about the signal learned from a large collection of training data.

Neural networks (NNs) are powerful learning architectures that are typically constructed via the composition of basic modules such as linear (or affine) mappings and nonlinear transformations (also called activation functions) [39, 40]. For example, an archetypal feedforward NN $\boldsymbol{f_\theta} : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ with component-wise ReLU activation functions is of the form

$$\boldsymbol{f_\theta}(\mathbf{x}) = \boldsymbol{A}_L \circ \cdots \circ \boldsymbol{\sigma}_\ell \circ \boldsymbol{A}_\ell \circ \cdots \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{A}_1(\mathbf{x}), \tag{2.21}$$

where the affine layer $\boldsymbol{A}_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$ (for $\ell = 1, \dots, L$) is given by

$$\boldsymbol{A}_\ell(\mathbf{x}) = \mathbf{W}_\ell \mathbf{x} + \mathbf{b}_\ell, \tag{2.22}$$

with weight matrices $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and bias vectors $\mathbf{b}_\ell \in \mathbb{R}^{N_\ell}$, the component-wise activation function $\boldsymbol{\sigma}_\ell : \mathbb{R}^{N_\ell} \to \mathbb{R}^{N_\ell}$ (for $\ell = 1, \dots, L-1$) is given by

$$\boldsymbol{\sigma}_\ell(\mathbf{x}) = \Big(\mathrm{ReLU}(x_1), \dots, \mathrm{ReLU}(x_{N_\ell})\Big), \tag{2.23}$$

with $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$, and $\boldsymbol{\theta} \coloneqq (\mathbf{W}_\ell, \mathbf{b}_\ell)_{\ell=1}^{L}$ denotes the complete set of its adjustable parameters. The idea behind using NNs for a specific task is to tune their parameters with the help of a training dataset such that they exhibit the desired behaviour.

As mentioned in the introductory chapter, NNs (in particular, convolutional NNs or CNNs which involve linear layers parametrized via convolutional operators with learnable kernels) have been applied in several ways for solving ill-posed inverse problems. The first successful applications of NNs in signal recovery build upon (fast) classical reconstruction algorithms, training a network to correct for their artifacts and output the desired high-quality estimates [41–43]. Unrolling methods [44–49], where the architecure of the network is based on iterations of the algorithms used in MPL estimation, also fall into this class of direct nonlinear reconstruction schemes. The Bayesian analogues of such approaches involve training the network to approximate a chosen estimator or even directly generate samples from the posterior distribution [50]. While these *end-to-end learning* methods have achieved state-of-the-art performances in a variety of applications such as MRI, CT, optical imaging, and ultrasound, they are not "flexible". More specifically, in such methods, the networks are trained on datasets consisting of signals and their measurements, which makes them highly sensitive to the corresponding measurement-acquisition setup. In this thesis, we will mainly focus on *universal* NN-based reconstruction methods, which involve using a network that has been pretrained to capture only some prior information about the signal (in a generic way that is independent of the inverse problem at hand) for its recovery. Here, we discuss some schemes that belong to the above-described category.

The plug-and-play (PnP) priors [51] and regularization-by-denoising (RED) [52, 53] frameworks are two well-known frameworks where NNs are deployed within the penalized-likelihood-based estimation paradigm. In PnP (RED) methods, the proximal (gradient) operator of $\mathcal{R}$ that appears in the iterations of the proximal (gradient) algorithms used for MPL estimation is replaced by an off-the-shelf denoiser (the residual of an off-the-shelf denoiser). To give some examples, the iterations for the PnP-FBS and RED-GD methods are given by

$$\mathbf{s}_{k+1} = \mathrm{D}\Big(\mathbf{s}_k - \gamma \boldsymbol{\nabla}\mathcal{D}\big(\mathbf{y}, \mathbf{H}(\mathbf{s}_k)\big)\Big) \tag{2.24}$$

and

$$\mathbf{s}_{k+1} = \mathbf{s}_k - \gamma\Big(\boldsymbol{\nabla}\mathcal{D}\big(\mathbf{y}, \mathbf{H}(\mathbf{s}_k)\big) + \tau\big(\mathbf{s}_k - \mathrm{D}(\mathbf{s}_k)\big)\Big), \tag{2.25}$$

respectively, where $\mathrm{D} : \mathbb{R}^K \to \mathbb{R}^K$ is the chosen denoiser, $\gamma \in \mathbb{R}_+$ is the step-size and $\tau \in \mathbb{R}_+$ is the regularization parameter. Generally speaking, such iterative schemes do not minimize an explicit cost functional, that is, the denoiser D plays the role of an implicit regularizer. However, convergence of the iterates to a fixed point can still be ensured if D belongs to a suitable class of 1-Lipschitz[2] operators [54, 55] (we provide the details for PnP-FBS in Section 5.2.2 of Chapter 5). In the learning-based variants of these frameworks, one uses denoising routines constructed from appropriately trained NNs [56–62]. The delicate point there is that in order to ensure convergence, the network must be constrained such that the corresponding denoiser belongs to the desired class of 1-Lipschitz operators. This is a challenging task and remains an active area of research [55, 63]. In Chapter 5 (Section 5.2), we will present a novel approach for designing and training powerful 1-Lipschitz denoising NNs which can then be used to develop provably convergent

[2]An operator $\mathrm{T}\colon \mathbb{R}^K \to \mathbb{R}^K$ is L-Lipschitz (with respect to the norm $\|\cdot\|$) if $\|\mathrm{T}(\mathbf{x}_1) - \mathrm{T}(\mathbf{x}_2)\| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|$ for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^K$.

iterative reconstruction methods within the PnP and RED frameworks.

More recently, gradient-step denoisers of the form $\mathrm{D} = \mathrm{Id} - \boldsymbol{\nabla} g_{\boldsymbol{\theta}}$, where the function $g_{\boldsymbol{\theta}} : \mathbb{R}^K \to \mathbb{R}_+$ is parametrized with the help of a network, have been used to devise PnP and RED methods that actually minimize an explicit global cost functional [64–66] and are thus more interpretable. Outside of these frameworks, NNs have also been deployed for directly designing an explicit learnable general-purpose regularization term [67]. In line with this recent trend of developing interpretable convergent NN-based reconstruction algorithms, we will present an efficient approach for learning convex regularizers $\mathcal{R}$ in Chapter 5 (Section 5.3).

There also exist regularization schemes that utilize deep latent variable generative models such as variational autoencoders (VAEs) [68] and generative adversarial networks (GANs) [69]. These models consist of a generator network $\mathrm{G}_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^K$ $(d \ll K)$, where $\boldsymbol{\theta} \in \mathbb{R}^P$ denotes its parameters, that maps a low-dimensional latent space to the high-dimensional signal space. They are trained on a dataset of signals such that they capture its statistics and generate sample signals $\mathrm{G}_{\boldsymbol{\theta}^*}(\mathbf{z})$, where $\boldsymbol{\theta}^*$ are the parameters of the network after the training is complete and $\mathbf{z} \in \mathbb{R}^d$ is sampled from a fixed simple distribution such as the uniform or Gaussian distribution, similar to those in the dataset. The application of such a trained deep generative model (DGM) to an inverse problem [70–72] typically involves specifying the signal estimator as $\mathbf{s}^*_{\mathrm{DGM}}(\mathbf{y}) = \mathrm{G}_{\boldsymbol{\theta}^*}(\mathbf{z}^*_{\mathrm{DGM}}(\mathbf{y}))$ with

$$\mathbf{z}^*_{\mathrm{DGM}}(\mathbf{y}) \in \underset{\mathbf{z} \in \mathbb{R}^d}{\arg\min}\ \mathcal{D}\Big(\mathbf{y},\ \mathbf{H}(\mathrm{G}_{\boldsymbol{\theta}^*}(\mathbf{z}))\Big). \tag{2.26}$$

Here, $\mathbf{s}^*_{\mathrm{DGM}}(\mathbf{y})$ is an MPL estimator for the signal corresponding to the regularization term that is an indicator function that assigns an infinite cost to any signal not in the range of $\mathrm{G}_{\boldsymbol{\theta}^*}$. As proposed in [70], one can also include a suitable penalty term in (2.26) to introduce a bias towards certain regions in the latent space. In this case, although $\mathbf{z}^*_{\mathrm{DGM}}(\mathbf{y})$ can be viewed an MPL estimator (for the latent vector), $\mathbf{s}^*_{\mathrm{DGM}}(\mathbf{y})$ no longer has such an interpretation.

Most of the NN-based models mentioned above can also be utilized within the Bayesian estimation paradigm. In fact, efficient posterior sampling schemes have been developed for prior probability distributions encoded by denoising NNs [73–75], GANs [76], VAEs [77, 78], score-based generative models [79, 80], and energy-based generative models [81]. In Chapter 6, we will present a Bayesian framework for solving nonlinear inverse problems that leverages deep latent variable models (e.g., VAEs, GANs).

Finally, we also discuss a class of NN-based methods that remarkably do not require any training data [82, 83, 99]. There, the main idea is to use the structure of an untrained neural network (UNN) $\mathrm{G}_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^K$, where $\boldsymbol{\theta} \in \mathbb{R}^P$ denotes its parameters, to specify a signal model. In these methods, the reconstruction is given by $\mathbf{s}^*_{\mathrm{UNN}}(\mathbf{y}) = \mathrm{G}_{\boldsymbol{\theta}^*_{\mathrm{UNN}}(\mathbf{y})}(\mathbf{z}_{\mathrm{in}})$, where

$$\boldsymbol{\theta}^*_{\mathrm{UNN}}(\mathbf{y}) \in \underset{\boldsymbol{\theta}\in\mathbb{R}^P}{\arg\min}\ \mathcal{D}\Big(\mathbf{y},\ \mathbf{H}(\mathrm{G}_{\boldsymbol{\theta}}(\mathbf{z}_{\mathrm{in}}))\Big) \tag{2.27}$$

and $\mathbf{z}_{\mathrm{in}} \in \mathbb{R}^d$ is an input vector that is randomly initialized and then kept fixed during the optimization of the network parameters. Similar to the case of trained DGMs, $\mathbf{s}^*_{\mathrm{UNN}}(\mathbf{y})$ is an MPL estimator for the signal corresponding to the regularization term that constrains it to lie in the range of the network for the fixed input $\mathbf{z}_{\mathrm{in}}$. In practice, if $\mathrm{G}_{\boldsymbol{\theta}}$ is a deep network, one typically deploys early stopping while optimizing the criterion in (2.27) to prevent the network from fitting the noise in the measurements. Alternately, one can also perform MPL estimation or Bayesian estimation for the network parameters with simple models such as ℓ_2-norm regularization or Gaussian priors [84]. The success of these schemes involving untrained NNs is attributed to the implicit signal model imposed by the architecture of the network, which favours natural-looking signals ("good" solutions) over noisy ones ("bad" solutions). In Chapter 7, we will present an extended version of such a method in the context of dynamic Fourier ptychography.

2.2.4 A Philosophical Note

We can see from Equations (2.8) and (2.15) that for a fixed noise model (p_{noise}), an MPL estimator with the regularization term $\mathcal{R}(\cdot)$ can be interpreted as a MAP estimator corresponding to the prior distribution $p_{\mathsf{S}}(\cdot) \propto \exp(-\mathcal{R}(\cdot))$. However, as pointed out in some works such as [11, 12], one must exercise caution while making such interpretations. The key point that we want to emphasize here is that there is a fundamental difference between the penalized-likelihood-based estimation and Bayesian estimation paradigms in terms of what the respective signal models ($\mathcal{R}$ and p_{S}) represent. Specifically, the goal in MPL estimation is to choose $\mathcal{R}$ such that the solution to the optimization problem in (2.8) exhibits some desirable properties. On the other hand, in Bayesian estimation, the holy grail is to specify p_{S} such that samples generated from this distribution resemble the signal of interest. An important implication of this difference in philosophies is that the above-mentioned interpretation of a given MPL estimator as some MAP estimator can be misleading as the two signal models—$\mathcal{R}(\cdot)$ and $p_{\mathsf{S}}(\cdot) \propto \exp(-\mathcal{R}(\cdot))$—might not reflect the same information about the underlying signal. For example, the regularization term $\mathcal{R}(\mathbf{s}) = \|\mathbf{s}\|_1$ is known to promote sparse solutions [100]. On the contrary, samples from the multivariate Laplace distribution $p_{\mathsf{S}}(\mathbf{s}) \propto \exp(-\|\mathbf{s}\|_1)$ are not sparse vectors.

2.3 Summary

In this chapter, we have discussed two well-known statistical reconstruction paradigms—penalized-likelihood-based estimation and Bayesian estimation—for solving ill-posed inverse problems. In the following chapters, we will present our contributions to developing novel signal-reconstruction methods within these paradigms.

Part I The World of Sparsity

3 Continuous-Domain L_p-Norm Regularization

As mentioned in Chapter 2, most real-world inverse problems are concerned with the recovery of a continuous-domain signal. The typical pipeline for tackling such problems first involves formulating them in terms of a discrete representation of the underlying signal. Prior information about the signal is then introduced through a model for its discrete representation. Alternately, one can also directly specify the estimation task and signal model in the continuum (provided that a solution can be computed analytically or numerically) [13, 101]. [1]In this chapter, we seek to understand the effect of one such model for 1D signals—continuous-domain L_p-norm regularization for $p \geq 1$ and with a multi-order derivative regularization operator D^{N_0}. To that end, we develop a numerical method to solve the L_p-regularized generalized-interpolation problem. Through a series of experiments, we then identify properties of this regularization.

3.1 Introduction

For a 1D continuous-domain signal s, a natural candidate for the regularization term is $\|\mathrm{L}\{s\}\|$, where L is a linear operator. Continuous-domain regularization schemes such as Tikhonov [13, 103, 104], which uses the L_2-norm $\|\cdot\|_{L_2}$, and generalized total variation (gTV) [101, 105], which involves the use of the $\mathcal{M}$-norm $\|\cdot\|_{\mathcal{M}}$ (an extension of the L_1-norm), have been intensively studied and their behavior is well-documented. To see the effect of these schemes, we consider the interpolation problem shown in Figure 3.1. The objective there is to construct a continuously defined function that passes through the given data points exactly. However, as shown in the figure, it is possible to construct infinitely many valid solutions. In this problem, we regularize the solution by imposing a minimum-norm requirement of the form $\|\mathrm{L}\{s\}\|$. This enables us to obtain solutions with certain desired properties. It is well-known that Tikhonov (or L_2) regularization tends to produce smooth solutions while gTV regularization promotes sparsity. These characteristics can be seen in Figure 3.1. For example, when we impose gTV regularization

[1]This chapter is based on our work [102].

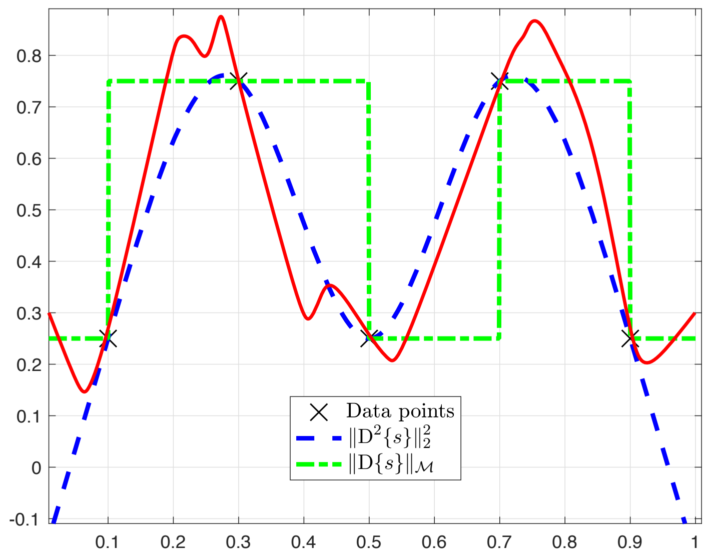


Figure 3.1: Interpolation of data points symbolized by crosses. The solid line represents an arbitrary solution. For the other two cases, it is regularization that dictates how the points are connected.

with $L = D$ (the derivative operator), we obtain a piecewise-constant solution whose derivative is sparse.

The purpose of this chapter is to study the effect of continuous-domain L_p-norm regularization for a general $p \geq 1$ and a multi-order derivative operator $L = D^{N_0}$. To that end, we consider the generalized interpolation problem with L_p-norm regularization. Generalized interpolation is an extension of interpolation. Specifically, given certain measurement functionals $(\nu_1, ..., \nu_M)$ and a value (or measurement) y_m corresponding to each functional, we aim at constructing a continuously defined function that explains the measurements exactly. We formulate this problem as

$$\min_s \|D^{N_0}\{s\}\|_{L_p} \quad \text{s.t.} \quad \langle \nu_m, s\rangle = y_m, \ m = 1, 2, ..., M, \tag{3.1}$$

where $\|\cdot\|_{L_p}$ denotes the L_p-norm.

3.1.1 Why Generalized Interpolation?

Consider the problem of reconstructing a signal s_0 from a finite number of its noisy linear measurements $\mathbf{y} \in \mathbb{R}^M$. The continuous-domain MPL estimate for s_0 can be written as

$$\mathcal{S} = \arg\min_{s\in\mathcal{X}} \left(\mathcal{D}\Big(\mathbf{y}, \boldsymbol{\nu}(s)\Big) + \tau\mathcal{R}(s)\right), \tag{3.2}$$

where $\mathcal{X}$ is a suitable function space, the operator $\boldsymbol{\nu} : s \mapsto \boldsymbol{\nu}(s) = (\langle \nu_1, s\rangle, \dots, \langle \nu_M, s\rangle)$ describes the measurement model, $\mathcal{D} : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$ is the data-fidelity term which depends on the statistics of the noise and $\mathcal{R}$ is the regularization. It can be shown (see Appendix 3.8) that, if $\mathcal{D}$ is strictly convex and $\mathcal{R}$ is convex, then all the solutions $s^* \in \mathcal{S}$ generate the same measurement vector $\mathbf{z}_0 = \boldsymbol{\nu}(s^*) \in \mathbb{R}^M$. This property allows us to characterize the solution set $\mathcal{S}$ as

$$\mathcal{S} = \arg\min_{s \in \mathcal{X}} \ \mathcal{R}(s) \ \text{ s.t. } \ \boldsymbol{\nu}(s) = \mathbf{z}_0. \tag{3.3}$$

By understanding the effect of the regularization term $\mathcal{R}(s)$ in (3.3), we can understand its effect for a much broader class of problems such as (3.2).

3.1.2 Related Work

The L_p-regularized interpolation problem and its variants, with $p \geq 1$ and $\mathrm{L} = \mathrm{D}^{N_0}$, have been studied in [106–111] in the context of approximation theory and splines. These works are theoretical, for the most part. They discuss the existence of a solution, conditions of optimality, and provide the functional form of the N_0th derivative of the solution. A specific instance of minimizing the L_p-norm of the second derivative of polynomial spline interpolants has been looked at in [112] and [113]. To the best of our knowledge, however, there exists no work that numerically solves the general continuous-domain problem (3.1) and demonstrates the effect of L_p-norm regularization.

3.1.3 Contributions

In this chapter, we propose an algorithm to compute the solution to (3.1). Through a series of experiments, we then identify some properties of L_p-norm regularization. Here is a list of our contributions.

- We discretize the continuous-domain problem (3.1) by using a basis that consists of shifted polynomial B-splines of degree N_0, with knots on a uniform grid. This basis leads to an exact discretization, thus transforming our continuous-domain problem into an equivalent finite-dimensional discrete one which can be solved by algorithms such as the alternating-direction method of multipliers (ADMM) [24].

- We implement a multiresolution algorithm that progressively decreases the grid size until a solution with the desired precision is obtained. This strategy relies on the theory of approximation. It dictates that, when the grid size is sufficiently small, the search space spanned by our B-spline basis contains functions that are arbitrarily close to the solution of the full continuous-domain problem where our constraint that the solution must live in a spline space would have been lifted.

- We present numerical results for measurement operators that correspond to interpolation in the spatial and Fourier domains. In these experiments, we show the existence of a continuum of solutions as p varies from ∞ to 1. We then examine properties of L_p-regularized solutions such as sparsity, regularity (smoothness) and, oscillatory behavior and overshoot, as well as the effect of N_0 on the solutions.

The chapter is organized as follows: In Section 3.2, we introduce the continuous-domain framework and discuss some existing theoretical results. We provide background information on polynomial splines in Section 3.3. Section 3.4 includes the details of our discretization scheme, along with a discussion on the approximation power of splines. We present the multiresolution algorithm in Section 3.5 and illustrate our numerical results in Section 3.6.

3.2 Generalized Interpolation

In this section, we define and discuss the key components of our framework: the measurement operator, the regularization operator, the regularization norms, and the search space for the optimization problem. We then briefly review theoretical results available for this problem.

3.2.1 Continuous-Domain Framework

In generalized interpolation, the aim is to construct a function $s : \mathbb{R} \to \mathbb{R}$ that explains the measurements $\mathbf{y} \in \mathbb{R}^M$, with

$$\boldsymbol{\nu}(s) = \Big(\langle \nu_1, s\rangle, \dots, \langle \nu_M, s\rangle\Big) = \mathbf{y}, \tag{3.4}$$

where $\langle \nu_m, s\rangle$ represents the action of the linear functional $\nu_m : s \mapsto \langle \nu_m, s\rangle = \nu_m(s) \in \mathbb{R}$. When ν_m and s are ordinary functions defined over $\mathbb{R}$, the mth measurement is given by the Lebesgue integral $\langle \nu_m, s\rangle = \int_{\mathbb{R}} \nu_m(x)s(x)\mathrm{d}x$. In the pure interpolation problem, the measurement functionals are shifted Dirac distributions $\nu_m = \delta(\cdot - x_m)$, with the property that $\langle \delta(\cdot - x_m), s\rangle = s(x_m)$.

In order to specify the regularization operator L, we introduce the Schwartz space $\mathcal{S}(\mathbb{R})$ of smooth and rapidly decaying functions defined over $\mathbb{R}$. Its continuous dual is the space of tempered distributions, denoted by $\mathcal{S}'(\mathbb{R})$. In our framework, we focus on regularization operators of the form $\mathrm{L} = \mathrm{D}^{N_0} : \mathcal{S}'(\mathbb{R}) \to \mathcal{S}'(\mathbb{R})$, where D is the derivative operator extended to $\mathcal{S}'(\mathbb{R})$ [114, Chapter 3] and $N_0 \geq 1$. The null space of the operator D^{N_0} is $\mathcal{N}_{\mathrm{D}^{N_0}} = \mathrm{span}\{p_n\}_{n=1}^{N_0}$, with $p_n(x) = x^{n-1}$. The Green's function of D^{N_0} is denoted by $\rho_{\mathrm{D}^{N_0}}$; it satisfies the property that $\mathrm{D}^{N_0}\{\rho_{\mathrm{D}^{N_0}}\} = \delta$. The Green's function is not unique due to the existence of the null space.

Next, we specify the the continuous-domain L_p-norm. For a measurable function $w : \mathbb{R} \to \mathbb{R}$, the L_p-norm ($1 \leq p < \infty$) is defined as

$$\|w\|_{L_p} \triangleq \left(\int_{\mathbb{R}} |w(x)|^p \mathrm{d}x \right)^{\frac{1}{p}}, \tag{3.5}$$

while the L_∞-norm is defined as[2]

$$\|w\|_{L_\infty} \triangleq \operatorname*{ess\,sup}_{x\in\mathbb{R}} |w(x)|. \tag{3.6}$$

Equation (3.5) also specifies the L_p quasinorm for values of $p \in (0, 1)$. The Lebesgue space of functions $L_p(\mathbb{R}) = \{w : \mathbb{R} \to \mathbb{R} \mid \|w\|_{L_p} < \infty\}$, where $p \in [1, \infty]$, is a Banach space. Here, we also define the $\mathcal{M}$-norm used in gTV regularization, which is closely related to L_1 regularization, as

$$\|w\|_{\mathcal{M}} \triangleq \sup_{\varphi\in\mathcal{S}(\mathbb{R}), \|\varphi\|_\infty=1} \langle w, \varphi \rangle \tag{3.7}$$

for any $w \in \mathcal{S}'(\mathbb{R})$. The Banach space associated with $\|\cdot\|_{\mathcal{M}}$ is $\mathcal{M}(\mathbb{R}) = \{w \in \mathcal{S}'(\mathbb{R}) \mid \|w\|_{\mathcal{M}} < +\infty\}$. The $\mathcal{M}$-norm is an extension of the L_1-norm. Indeed, for any $w \in L_1(\mathbb{R})$, we have that

$$\|w\|_{\mathcal{M}} = \|w\|_{L_1}. \tag{3.8}$$

However, the Dirac impulse δ is included in $\mathcal{M}(\mathbb{R})$ with $\|\delta\|_{\mathcal{M}} = 1$ but does not belong to $L_1(\mathbb{R})$. Thus, we have that $L_1(\mathbb{R}) \subset \mathcal{M}(\mathbb{R})$.

Finally, we define the search spaces for the gTV-regularized and L_p-regularized problems as

$$\mathcal{M}^{(N_0)}(\mathbb{R}) = \{s \in \mathcal{S}'(\mathbb{R}) \mid \mathrm{D}^{N_0}\{s\} \in \mathcal{M}(\mathbb{R})\} \tag{3.9}$$

$$L_p^{(N_0)}(\mathbb{R}) = \{s \in \mathcal{S}'(\mathbb{R}) \mid \mathrm{D}^{N_0}\{s\} \in L_p(\mathbb{R})\}. \tag{3.10}$$

Here, we consider all generalized functions in $\mathcal{S}'(\mathbb{R})$ for which the regularization term is finite.

Now that we have described all the components involved in our regularized generalized-interpolation framework, we state the optimization problems that we consider in this

[2]The essential supremum is a generalization of the supremum in Lebesgue's theory of integration. For a measurable function $w : \mathbb{R} \to \mathbb{R}$, it is the smallest value $a \in \mathbb{R}$ such that $w(x) \leq a$ almost everywhere (i.e., everywhere except on a set of measure zero). The essential supremum is equivalent to the supremum for continuous functions.

work. They are

$$\mathcal{S}_{\mathcal{M}} = \arg\min_{s \in \mathcal{M}^{(N_0)}(\mathbb{R})} \|\mathrm{D}^{N_0}\{s\}\|_{\mathcal{M}} \quad \text{s.t.} \quad \boldsymbol{\nu}(s) = \mathbf{y} \tag{3.11}$$

$$\mathcal{S}_p = \arg\min_{s \in L_p^{(N_0)}(\mathbb{R})} \|\mathrm{D}^{N_0}\{s\}\|_{L_p} \quad \text{s.t.} \quad \boldsymbol{\nu}(s) = \mathbf{y}, \tag{3.12}$$

where $N_0 \geq 1$.

3.2.2 Theoretical Results

Before the discussion of theoretical results, we need to make some assumptions.

Assumptions 3.1. *In the following statements, the symbol $\mathcal{X}$ represents the search space $\mathcal{M}^{(N_0)}(\mathbb{R})$ or $L_p^{(N_0)}(\mathbb{R})$, depending on the problem at hand.*

i. The measurement operator $\boldsymbol{\nu}$ is weak-continuous on $\mathcal{X}$.*

ii. For the given measurements $\mathbf{y} \in \mathbb{R}^M$ and measurement operator $\boldsymbol{\nu}$, there exists at least one function $s_0 \in \mathcal{X}$ such that $\boldsymbol{\nu}(s_0) = \mathbf{y}$.

iii. The intersection of the null spaces of $\boldsymbol{\nu}$ and D^{N_0} is $\{0\}$.

Assumption (1.i) implies that the measurement functionals satisfy $\nu_m \in \mathcal{Y}$ for $m = 1, ..., M$, where the predual space $\mathcal{Y}$ is such that $\mathcal{X} = \mathcal{Y}'$. In practice, this imposes a minimum degree of regularity and decay on $\{\nu_m\}_{m=1}^M$. Assumption (1.ii) states a feasibility condition and is needed to ensure that the generalized interpolation problem is well-defined. If (1.i) holds and the ν_m are linearly independent, then (1.ii) is satisfied for any $\mathbf{y} \in \mathbb{R}^M$. Assumption (1.iii) ensures that the problem is well-posed over the null space of the regularization operator, where the penalization is immaterial. This can be checked by verifying that the matrix $\mathbf{P}$ with entries $[\mathbf{P}]_{m,n} = \langle \nu_m, p_n \rangle$ is full-rank.

For the gTV-regularized and L_2-regularized problems, there exist representer theorems that provide a parametric characterization of the possible range of solutions. In the case of L_2 regularization, the solution is unique, smooth, and lies in a finite-dimensional subspace that depends on the measurement and regularization operators [104]. The gTV problem can have infinitely many solutions, but the extreme points of the solution set $\mathcal{S}_{\mathcal{M}}$ are known to be splines whose type depends on the regularization operator only [101]. These splines have adaptive knots which are fewer than the number of measurements. On applying the operator D^{N_0} to these extreme points, we recover Dirac impulses at the knot locations, which implies a sparse N_0th order derivative. We refer to such solutions as the sparse solutions of the gTV problem.

Beside providing insights about the nature of the solutions, the representer theorems also play a role in the design of numerical methods to solve these problems. The parametric forms of the solution provided by the theorems are used for the discretization of the continuous-domain problems, leading to finite-dimensional optimization tasks which can be solved using standard optimization algorithms. A detailed comparison of L_2 versus gTV regularization can be found in [104]. The reader can refer to [104, 115] for the algorithms.

In this work, our main focus is on (3.12) with a general $p \geq 1$. This kind of a problem has been addressed in [106] for the case of pure interpolation, when the measurement functionals are Dirac impulses. Here, we state the result from [106] in a form that is compatible with our framework. When $p \in (1, \infty)$, there exists a unique solution s^* to the L_p-regularized interpolation problem. It satisfies

$$\mathrm{D}^{N_0}\{s^*\} = \frac{|v^*|^{q-1}}{\|v^*\|_{L_q}^{q-2}} \operatorname{sgn}(v^*), \tag{3.13}$$

where $\frac{1}{p} + \frac{1}{q} = 1$ and

$$v^*(x) = \sum_{m=1}^{M} a_m \rho_{\mathrm{D}^{N_0}}(x - x_m) + \sum_{n=1}^{N_0} b_n p_n(x) \tag{3.14}$$

is a polynomial spline with knots at the data points $\{x_m\}_{m=1}^M$, and where $\{a_m\}_{m=1}^M$ and $\{b_n\}_{n=1}^{N_0}$ are suitable sets of coefficients. On setting $p = 2$, we recover the result given in [104]. Equations (3.13)-(3.14) show that the N_0th derivative of the solution to our continuous-domain problem lies in a finite-dimensional manifold. The solution itself can then be obtained by taking an N_0-fold integral, subject to adequate boundary conditions. However, for $p \neq 2$, we have a nonlinear mapping in (3.13). This makes it difficult to interpret other effects of regularization on the solution. Moreover, due to this nonlinear mapping, these solutions do not readily lend themselves to a discretization scheme, unlike in the gTV and L_2 cases. Therefore, we propose a spline-based discretization scheme to numerically solve the L_p-regularized generalized-interpolation problem for $p \geq 1$.

3.3 Polynomial Splines

Polynomial splines of degree N_0 form an essential component of our discretization scheme. They are piecewise-defined functions where each piece is a polynomial of degree N_0. These pieces are connected in a manner such that the first $(N_0 - 1)$ derivatives of the function are continuous. The points where the pieces are connected are called knots. A cardinal polynomial spline of degree N_0 has its knots on the integer grid and can be expressed uniquely in the form of a B-spline expansion [116]

$$s(x) = \sum_{k \in \mathbb{Z}} c[k] \beta_+^{N_0}(x - k), \tag{3.15}$$

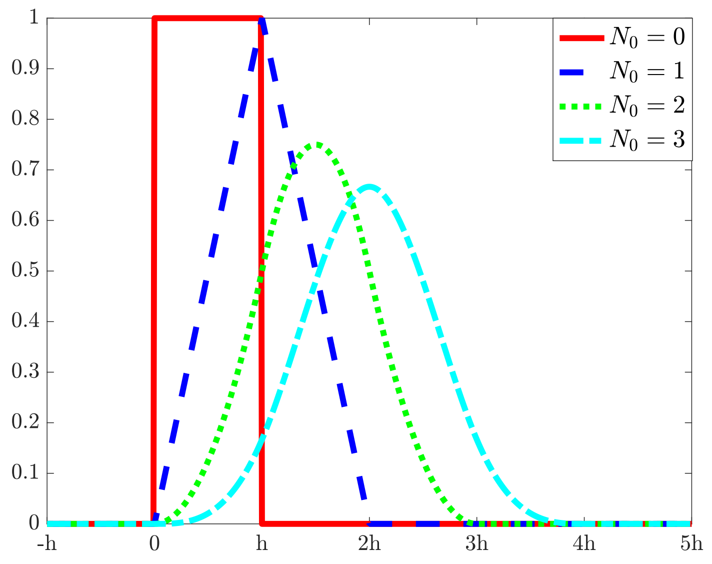


Figure 3.2: Causal B-splines $\beta_h^{N_0}(x)$ with scaling factor h.

where $\beta_+^{N_0}(x)$ is the causal B-spline of degree N_0 and $(c[k])_{k\in\mathbb{Z}}$ are the expansion coefficients. The causal B-spline of degree 0 is defined as:

$$\beta_+^0(x) = \begin{cases} 1, & \text{if } 0 \leq x < 1 \\ 0, & \text{otherwise,} \end{cases} \tag{3.16}$$

while the causal B-spline of degree N_0 is obtained by the (N_0+1)-fold convolution of $\beta_+^0(x)$ given by

$$\beta_+^{N_0}(x) = \underbrace{(\beta_+^0 * \beta_+^0 * \cdots * \beta_+^0)}_{N_0 \text{ convolutions}}(x). \tag{3.17}$$

We are interested in polynomial splines with knots located on a uniform grid of size h (in other words, the knots lie in $h\mathbb{Z}$). Such a spline of degree N_0 admits the B-spline expansion

$$s_h(x) = \sum_{k\in\mathbb{Z}} c_h[k]\beta_h^{N_0}(x - kh), \tag{3.18}$$

where $\beta_h^{N_0}(x) = \beta_+^{N_0}\left(\frac{x}{h}\right)$ is the causal scaled B-spline of degree N_0. It is uniquely specified by its coefficients $c_h = (c_h[k])_{k\in\mathbb{Z}}$. We illustrate in Figure 3.2 that $\beta_h^{N_0}(x)$ is compactly supported in $[0, (N_0+1)h]$. In fact, the B-spline $\beta_h^{N_0}(x)$ is the polynomial spline of degree N_0, with knots in $h\mathbb{Z}$, that has the shortest support [117].

Polynomial splines are closely linked to derivative operators of the form D^{N_0} $(N_0 \geq 1)$.

Table 3.1: The operator D^{N_0} and the scaled B-spline $\beta_h^{N_0-1}(x)$ and sequence $(d_{N_0}[k])_{k\in\mathbb{Z}}$ associated with it.

$\mathrm{L} = \mathrm{D}^{N_0}$	$\beta_h^{N_0-1}(x)$	$(d_{N_0}[0], \dots, d_{N_0}[N_0])$
D	$\beta_h^0(x) = \begin{cases} 1, & 0 \le x < h \\ 0, & \text{otherwise} \end{cases}$	$(1, -1)$
D^2	$\beta_h^1(x) = \begin{cases} x/h, & 0 \le x < h \\ (2h-x)/h, & h \le x < 2h \\ 0, & \text{otherwise} \end{cases}$	$(1, -2, 1)$
D^3	$\beta_h^2(x) = \begin{cases} x^2/2h, & 0 \le x < h \\ (-2x^2+6xh-3h^2)/2h^2, & h \le x < 2h \\ (3h-x)^2)/2h^2, & 2h \le x < 3h \\ 0, & \text{otherwise} \end{cases}$	$(1, -3, 3, -1)$

The operator D^{N_0} is associated with the scaled B-spline of degree $(N_0 - 1)$ according to

$$\mathrm{D}^{N_0}\{\beta_h^{N_0-1}\}(x) = \frac{1}{h^{N_0-1}} \sum_{k\in\mathbb{Z}} d_{N_0}[k]\delta(x - kh). \tag{3.19}$$

The sequence $(d_{N_0}[k])_{k\in\mathbb{Z}}$ is characterized by its z-transform

$$d_{N_0}(z) = (1 - z^{-1})^{N_0} \tag{3.20}$$

and is supported in $\{0, \dots, N_0\}$. In Table 3.1, we provide the explicit forms of $\beta_h^{N_0-1}(x)$ and $(d_{N_0}[k])_{k\in\mathbb{Z}}$ for $N_0 = 1, 2, 3$.

3.4 Discretization Scheme

3.4.1 Search Space

We discretize the continuous-domain problem (3.12) by restricting the search space to a suitable space of polynomial splines, defined as

$$L_{p,h}^{N_0}(\mathbb{R}) = \left\{ \sum_{k\in\mathbb{Z}} c[k]\beta_h^{N_0}(\cdot - kh) \ : \ c \in \ell_p^{N_0}(\mathbb{Z}) \right\}, \tag{3.21}$$

where $\beta_h^{N_0}$ is the scaled B-spline of degree N_0, $h > 0$ is the grid size, and

$$\ell_p^{N_0}(\mathbb{Z}) = \left\{ (c[k])_{k\in\mathbb{Z}} : (d_{N_0} * c) \in \ell_p(\mathbb{Z}) \right\}. \tag{3.22}$$

The choice of the search space $L_{p,h}^{N_0}(\mathbb{R})$ is guided by its exact discretization property which we discuss in Section 3.4.2. Moreover, the approximation power of splines ensures that, when h is sufficiently small, the search space $L_{p,h}^{N_0}(\mathbb{R})$ contains functions that are arbitrarily close to the solution of the unrestricted continuous-domain problem (3.12). We present a detailed argument for this in Section 3.4.4. The fact that $L_{p,h}^{N_0}(\mathbb{R})$ is represented in a B-spline basis is another advantage. B-splines are compactly supported and form a Riesz basis [118], thus resulting in a well-conditioned discretization.

3.4.2 Exact Discretization

The exact discretization property of the function space $L_{p,h}^{N_0}(\mathbb{R})$ stems from Proposition 3.1.

Proposition 3.1. *For any function $s \in L_{p,h}^{N_0}(\mathbb{R})$ with $p \in (0,\infty]$, we have that*

$$\|\mathrm{D}^{N_0}\{s\}\|_{L_p} = \left\| \frac{1}{h^{N_0-1/p}} (d_{N_0} * c) \right\|_{\ell_p}. \tag{3.23}$$

Proof. A scaled B-spline of degree N_0 can be expressed as

$$\beta_h^{N_0}(x) = \frac{1}{h}(\beta_h^{N_0-1} * \beta_h^0)(x). \tag{3.24}$$

Using (3.19) and (3.24), we deduce that

$$\mathrm{D}^{N_0}\{\beta_h^{N_0}\}(x) = \frac{1}{h^{N_0}} \sum_{k\in\mathbb{Z}} d_{N_0}[k]\beta_h^0(x-kh). \tag{3.25}$$

Therefore, for any $s \in L_{p,h}^{N_0}(\mathbb{R})$ it stands that

$$\mathrm{D}^{N_0}\{s\}(x) = \frac{1}{h^{N_0}} \sum_{k\in\mathbb{Z}} (d_{N_0} * c)[k]\beta_h^0(x-kh). \tag{3.26}$$

Equation (3.26) implies that $\mathrm{D}^{N_0}\{s\}$ is a piecewise-constant function. For $p \in (0,\infty)$, the following holds:

$$\begin{aligned}\|\mathrm{D}^{N_0}\{s\}\|_{L_p} &= \left(\int_{\mathbb{R}} \left| \frac{1}{h^{N_0}} \sum_{k\in\mathbb{Z}} (d_{N_0} * c)[k]\beta_h^0(x-kh) \right|^p \mathrm{d}x \right)^{\frac{1}{p}} \\ &= \left(\sum_{k\in\mathbb{Z}} h \left| \frac{1}{h^{N_0}} (d_{N_0} * c)[k] \right|^p \right)^{\frac{1}{p}} \\ &= \| \frac{1}{h^{N_0-1/p}} (d_{N_0} * c) \|_{\ell_p}. \end{aligned} \tag{3.27}$$

For the case $p = \infty$, we have that

$$\begin{aligned}\|\mathrm{D}^{N_0}\{s\}\|_{L_\infty} &= \operatorname*{ess\,sup}_{x\in\mathbb{R}} \left| \frac{1}{h^{N_0}} \sum_{k\in\mathbb{Z}} (d_{N_0} * c)[k]\beta_h^0(x - kh) \right| \\ &= \sup_{k\in\mathbb{Z}} \left| \frac{1}{h^{N_0}} (d_{N_0} * c)[k] \right| \\ &= \| \frac{1}{h^{N_0}} (d_{N_0} * c)\|_{\ell_\infty}. \end{aligned} \tag{3.28}$$

□

On plugging the parametric form (3.21) of any function $s \in L_{p,h}^{N_0}(\mathbb{R})$ into Problem (3.12) and using Proposition 3.1, we obtain the equivalent discrete problem

$$\mathcal{S}_{p,h} = \operatorname*{arg\,min}_{c\in\ell_p^{N_0}(\mathbb{Z})} \|\frac{1}{h^{N_0-1/p}}(d_{N_0} * c)\|_{\ell_p} \quad \text{s.t.} \quad \sum_{k\in\mathbb{Z}} c[k]\boldsymbol{\nu}(\beta_h^{N_0}(\cdot - kh)) = \mathbf{y} \tag{3.29}$$

The important thing to note here is that Problem (3.29) is *exactly* equivalent to the continuous-domain problem (3.12) restricted to the search space $L_{p,h}^{N_0}(\mathbb{R})$. In other words, by solving the above discrete problem, we effectively find a solution to the restricted continuous-domain problem, which is given by $\sum_{k\in\mathbb{Z}} c^*[k]\beta_h^{N_0}(\cdot - kh)$ with $c^* \in \mathcal{S}_{p,h}$. As indicated by Proposition 3.1, this discretization scheme is also valid for L_p quasinorm regularization with $p \in (0, 1)$. However, these values of p correspond to non-convex problems.

Interestingly, the function space $L_{1,h}^{N_0}(\mathbb{R})$ can also be used for discretizing the gTV problem (3.11), which then also happens to be equivalent to the $p = 1$ case.

Proposition 3.2. *For any function $s \in L_{1,h}^{N_0}(\mathbb{R})$, we have that*

$$\|\mathrm{D}^{N_0}\{s\}\|_{\mathcal{M}} = \|\mathrm{D}^{N_0}\{s\}\|_{L_1}. \tag{3.30}$$

Proof. Equation (3.26) implies that $\mathrm{D}^{N_0}\{s\}$ is piecewise-constant. Moreover, since $(d_{N_0} * c) \in \ell_1(\mathbb{Z})$, we conclude that $\mathrm{D}^{N_0}\{s\} \in L_1(\mathbb{R})$. The relationship between the $\mathcal{M}$-norm and L_1-norm (3.8) leads to (3.30). □

By restricting the search space in (3.11) to $L_{1,h}^{N_0}(\mathbb{R})$ and using Propositions 3.1 and 3.2, we obtain the discrete problem (3.29) with $p = 1$.

The salient and novel aspect of our method is the exact discretization of the continuous-domain problem. To the best of our knowledge, there is no prior work that discretizes L_p-regularized continuous-domain problems, with a general p, exactly. As mentioned earlier, the cases of $p = 2$ and gTV have also been handled in [104, 115]. However, those discretization schemes have been specifically derived from representer theorems for L_2

and gTV regularization, and unlike the method proposed in this paper, are not applicable for other values of p.

3.4.3 Finite-Dimensional Problem

In practice, we assume that the measurement functionals ν_m are supported over a finite interval $I_T = [0, T]$. Consequently, a finite number of B-spline expansion coefficients are now involved in the constraint term in (3.29). We denote the set of the indices of these coefficients by $K = \{k_{\min}, \dots, k_{\max}\}$; the cardinality of this set is $|K| = N$. We now state Proposition 3.3, which has been adapted from Lemma 3 in [115].

Proposition 3.3. *If the measurement functionals $\{\nu_m\}_{m=1}^{M}$ are supported in I_T, then a solution $c^* \in \mathcal{S}_{p,h}$ of Problem* (3.29) *is uniquely determined by the N coefficients $c^*|_K = (c^*[k_{min}], \dots, c^*[k_{max}])$.*

This result ensures that we only need to optimize over the N B-spline coefficients that affect the constraint (or data) term in (3.29). As described in [115], the expansion coefficients outside the interval of interest I_T can be set in a way such that all the regularization terms that they affect are nullified. This allows us to write the infinite-dimensional convolution in (3.29) as a matrix multiplication, leading to the finite-dimensional optimization problem

$$S_{p,h} = \arg\min_{\mathbf{c}\in\mathbb{R}^N} \|\mathbf{L}\mathbf{c}\|_{\ell_p} \quad \text{s.t.} \quad \mathbf{H}\mathbf{c} = \mathbf{y}, \tag{3.31}$$

where the system matrix $\mathbf{H} : \mathbb{R}^N \to \mathbb{R}^M$ is

$$\mathbf{H} = \begin{bmatrix} \vdots & & \vdots \\ \boldsymbol{\nu}(\beta_h^{N_0}(\cdot - k_{\min}h)) & \cdots & \boldsymbol{\nu}(\beta_h^{N_0}(\cdot - k_{\max}h)) \\ \vdots & & \vdots \end{bmatrix}, \tag{3.32}$$

and the regularization matrix $\mathbf{L} : \mathbb{R}^N \to \mathbb{R}^{N-N_0}$ is

$$\mathbf{L} = \frac{1}{h^{N_0 - \frac{1}{p}}} \begin{bmatrix} d_{N_0}[N_0] \cdots d_{N_0}[0] & 0 & \cdots & 0 \\ 0 \quad \ddots & \ddots & \ddots & \vdots \\ \vdots \quad \ddots \quad \ddots & & \ddots & 0 \\ 0 \quad \cdots \quad 0 & d_{N_0}[N_0] & \cdots & d_{N_0}[0] \end{bmatrix}. \tag{3.33}$$

The solutions $\mathbf{c}^* \in S_{p,h}$ and $c^* \in \mathcal{S}_{p,h}$ are related in the following manner: $\mathbf{c}^* = c^*|_K = (c^*[k_{\min}], \dots, c^*[k_{\max}])$. Proposition 3.3 implies that the solution to Problem (3.29) can be uniquely determined from $\mathbf{c}^*$. Thus, we conclude that Problem (3.31) is equivalent to the continuous-domain problem (3.12) ((3.11), respectively) restricted to the search space $L_{p,h}^{N_0}(\mathbb{R})$ ($L_{1,h}^{N_0}(\mathbb{R})$, respectively), in the sense that the continuous-domain solution can be fully described by $\mathbf{c}^*$.

3.4.4 Effect of the Grid Size

So far, we have seen that the solutions to our continuous-domain problems, when restricted to $L_{p,h}^{N_0}(\mathbb{R})$, can be obtained by simply solving the finite problem (3.31). Now, we look at the influence of the grid size h on these solutions. We define a linear projection operator for the function space $L_{p,h}^{N_0}(\mathbb{R})$ as

$$\mathcal{P}_{L_{p,h}^{N_0}}\{s\}(x) = \sum_{k\in\mathbb{Z}} \left\langle s, \frac{1}{h}\tilde{\beta}^{N_0}\left(\frac{\cdot}{h} - k\right)\right\rangle \beta_+^{N_0}\left(\frac{x}{h} - k\right), \tag{3.34}$$

where $\tilde{\beta}^{N_0}$ is a (generalized) function such that

$$\left\langle \beta_+^{N_0}(\cdot - p), \tilde{\beta}^{N_0}(\cdot - q)\right\rangle = \delta[p - q]. \tag{3.35}$$

The operator defined in (3.34) is a valid projection operator since it is idempotent. This can be shown by using the biorthonormality condition (3.35).

We now state Theorem 3.1, adapted from [119], which bounds the L_p-norm of the error between a function $s \in L_p^{(N_0)}(\mathbb{R})$ (the search space of the unrestricted continuous-domain problem, as defined in (3.9)) and its projection onto $L_{p,h}^{N_0}(\mathbb{R})$.

Theorem 3.1. *Let $\mathcal{P}_{L_{p,h}^{N_0}}$ be a linear projection operator for $L_{p,h}^{N_0}(\mathbb{R})$, as defined in* (3.34)*. When $p \in (1, \infty)$, the error of approximation for any $s \in L_p^{(N_0)}(\mathbb{R})$ is*

$$\|s - \mathcal{P}_{L_{p,h}^{N_0}}\{s\}\|_{L_p} = \mathcal{O}(h^{N_0}). \tag{3.36}$$

For a small-enough grid size h, the error of approximation for any $s \in L_p^{(N_0)}(\mathbb{R})$ will be negligible. Therefore, our restricted search space $L_{p,h}^{N_0}(\mathbb{R})$ will contain functions (projections) which are arbitrarily close to the solution of the unrestricted continuous-domain problem. Finally, to compute the solution to the restricted continuous-domain problem, we only need to solve the finite problem (3.31).

3.5 Multiresolution Algorithm

In this section, we discuss a multiresolution algorithm that computes a solution with the desired precision by gradually making the grid finer.

3.5.1 Solving the Finite Problem for a Fixed Grid Size

We first discuss the algorithm that we use to solve finite-dimensional problems of the form (3.31). As constrained-optimization problems are typically harder to solve numerically compared to their unconstrained counterparts, to make the optimization easier we consider the unconstrained version of (3.31) given by

$$S'_{p,h} = \arg\min_{\mathbf{c}\in\mathbb{R}^N} \left(\|\mathbf{y} - \mathbf{H}\mathbf{c}\|_2^2 + \tau\psi_p(\|\mathbf{L}\mathbf{c}\|_{\ell_p}) \right) \tag{3.37}$$

where $\tau \in \mathbb{R}^+$ is the regularization parameter and the function $\psi_p : \mathbb{R}^+ \to \mathbb{R}^+$ is defined as

$$\psi_p(x) = \begin{cases} x^p \text{ if } p \in [1, \infty), \\ x \;\text{ if } p = \infty. \end{cases} \tag{3.38}$$

Since ψ_p is monotonic over $\mathbb{R}^+$, the solution(s) to the constrained problem (3.31) can be obtained from (3.37) in the limit by taking $\tau \to 0$. Thus, we propose to solve our finite-dimensional problem (3.31) by solving (3.37) with a very small value of τ.

The case $p = 2$ is special since then the optimization problem (3.37) is quadratic and can be solved directly without the need for an iterative algorithm. The unique solution in this scenario can be obtained by solving the linear system of equations $(\mathbf{H}^T\mathbf{H} + \tau\mathbf{L}^T\mathbf{L})\mathbf{c}^* = \mathbf{H}^T\mathbf{y}$, which is obtained by setting to zero the gradient with respect to $\mathbf{c}$ of the cost functional in (3.37). This can be done by various methods, including direct matrix inversion.

For the values of $p \in [1, \infty] \setminus \{2\}$, we use the well-known ADMM [24] to solve Problem (3.37). The update rules for ADMM in our case are

$$\mathbf{c}^{k+1} = (\mathbf{H}^T\mathbf{H} + \frac{\rho}{2}\mathbf{L}^T\mathbf{L})^{-1}(\mathbf{H}^T\mathbf{y} + \frac{\rho}{2}\mathbf{L}^T(\mathbf{z}^k + \mathbf{u}^k)) \tag{3.39}$$

$$\mathbf{z}^{k+1} = \mathrm{prox}_{\tilde{\tau}\psi_p(\|\cdot\|_{\ell_p})}(\mathbf{L}\mathbf{c}^{k+1} + \mathbf{u}^k) \tag{3.40}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{L}\mathbf{c}^{k+1} - \mathbf{z}^{k+1}, \tag{3.41}$$

where $\mathbf{c}$ and $\mathbf{z}$ are the primal variables, $\mathbf{u}$ is the dual variable, $\rho > 0$ is the augmented-Lagrangian parameter and $\tilde{\tau} = \tau/\rho$. The proximal operator of a function g is defined as [120]

$$\mathrm{prox}_g(\mathbf{x}) = \arg\min_{\mathbf{u}} \left(\frac{1}{2}\|\mathbf{u} - \mathbf{x}\|_2^2 + g(\mathbf{u}) \right). \tag{3.42}$$

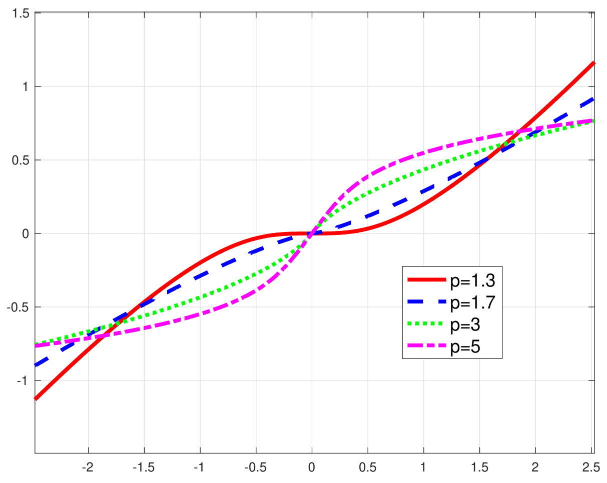


Figure 3.3: Lookup tables for the proximal operators of $|\cdot|^p$

For $p = \{1, \infty\}$, the proximal operators involved in (3.40) have the closed-form expressions

$$\mathrm{prox}_{\tilde{\tau}\|\cdot\|_{\ell_1}}(\mathbf{x}) = \mathrm{sgn}(\mathbf{x}) \otimes \max(|\mathbf{x}| - \tilde{\tau}, 0) \tag{3.43}$$

$$\mathrm{prox}_{\tilde{\tau}\|\cdot\|_{\ell_\infty}}(\mathbf{x}) = \mathbf{x} - \tilde{\tau}\mathrm{proj}_{\|\cdot\|_{\ell_1}\leq 1}(\mathbf{x}/\tilde{\tau}), \tag{3.44}$$

where the operators $\mathrm{sgn}(\cdot)$ and $\max(\cdot)$ are applied component-wise, $\otimes$ denotes component-wise multiplication, and the projection operator is

$$\mathrm{proj}_{\|\cdot\|_{\ell_1}\leq 1}(\mathbf{x}) = \underset{\mathbf{u}:\|\mathbf{u}\|_{\ell_1}\leq 1}{\arg\min}\ \|\mathbf{u} - \mathbf{x}\|_2^2. \tag{3.45}$$

This projector is computed as explained in [121]. Thus, the proximal operators can be computed efficiently for these two cases.

In general, we do not have a closed form expression for the proximal operator when $p \in (1, \infty)$. The additive separability of the function $\psi_p(\|\cdot\|_{\ell_p})$ can be used to observe that

$$[\mathrm{prox}_{\tilde{\tau}\psi_p(\|\cdot\|_{\ell_p})}(\mathbf{x})]_m = \mathrm{prox}_{\tilde{\tau}|\cdot|^p}([\mathbf{x}]_m). \tag{3.46}$$

Now, we only need to compute the proximal operator for the 1D function $\tilde{\tau}|\cdot|^p : \mathbb{R} \to \mathbb{R}$, which we do with the help of lookup tables (LUTs). We provide in Figure 3.3 a few examples of LUTs. An efficient implementation is achieved by exploiting properties of $\mathrm{prox}_{\tilde{\tau}|\cdot|^p}(\cdot)$ such as antisymmetry and monotonicity.

So far, we have seen that ADMM can be used to compute the unique solution to (3.37) when $p \in (1, \infty)$. When $p = \{1, \infty\}$, ADMM gives us one out of the possibly many solutions. In order to obtain a sparse solution for $p = 1$, we follow the procedure proposed

Algorithm 1 Multiresolution Algorithm

```
1: Input: $p$, $T$, $\mathbf{y}$, $\boldsymbol{\nu}$, $N_0$, $\tau$, $h_{\text{init}}$, $\epsilon$.
2: Output: $\mathbf{c}^*$
3: Initialization: $\mathbf{c} = \mathbf{0}$, $t = 0$, rel_error = $\epsilon + 1$, prev_cost = $+\infty$
4: while rel_error > $\epsilon$ do
5:     $h = h_{\text{init}}/2^t$
6:     Update $\mathbf{H}$, $\mathbf{L}$
7:     if $p = 2$ then
8:         $\mathbf{c} = (\mathbf{H}^T\mathbf{H} + \tau\mathbf{L}^T\mathbf{L})^{-1}\mathbf{H}^T\mathbf{y}$
9:     else
10:        $\mathbf{c} \leftarrow \text{ADMM}(\mathbf{c}_{\uparrow 2};\ p,\ \mathbf{y},\ \mathbf{H},\ \mathbf{L},\ \tau)$
11:    end if
12:    rel_error = |cost($\mathbf{c}$) − prev_cost| / prev_cost
13:    prev_cost = cost($\mathbf{c}$)
14:    $t \leftarrow t + 1$
15: end while
16: if $p = 1$ then
17:     $\mathbf{y}_\tau = \mathbf{Hc}$
18:     $\mathbf{c}^* = \text{Simplex}(\mathbf{y}_\tau,\ \mathbf{H},\ \mathbf{L})$
19: else
20:     $\mathbf{c}^* = \mathbf{c}$
21: end if
```

in [104]. The solution $\mathbf{c}^* \in S'_{p,h}$ obtained via ADMM is used to generate the measurements $\mathbf{y}_\tau = \mathbf{Hc}^*$. Using these "denoised" measurements, Problem (3.37) is then recast as a linear program which we solve using the simplex algorithm [122]. The simplex algorithm guarantees that we reach an extreme point of $S'_{1,h}$, which is sparse.

3.5.2 Grid Refinement

We begin with a coarse grid h_{init} and make it finer gradually until a further decrease of the grid size does not affect the solution much. At each iteration $t \in \mathbb{W}$, we pick a grid size $h_t = h_{\text{init}}/2^t$, splitting the grid from the previous iteration in half. We then solve the corresponding finite problem.

For this sequence of grid sizes, we observe that the search spaces are embedded like $L^{N_0}_{p,h_t}(\mathbb{R}) \subset L^{N_0}_{p,h_{t+1}}(\mathbb{R})$. This ensures that, by splitting the grid in half, we obtain a refined solution that is at least as good in terms of the cost function. Finally, we keep making the grid finer until the relative decrease in cost is less than some desired tolerance level ϵ. Another advantage of this embedding property is that the solution from the previous grid can be used as initialization for ADMM, which tends to improve the speed of convergence. This algorithm is adapted from the work in [115].

In Algorithm 1, $\mathbf{c}_{\uparrow 2}$ corresponds to the coefficients $\mathbf{c}$ modified to match a grid that is twice as fine as that of $\mathbf{c}$. The routine ADMM($\mathbf{c}_{\uparrow 2}$; p, $\mathbf{y}$, $\mathbf{H}$, $\mathbf{L}$, τ) runs ADMM on Problem (3.37) with $\mathbf{c}_{\uparrow 2}$ as the initialization while the routine Simplex($\mathbf{y}_\tau$, $\mathbf{H}$, $\mathbf{L}$) runs the simplex algorithm on the linear program obtained from Problem (3.37) by using the denoised measurements $\mathbf{y}_\tau$.

3.6 Numerical Experiments

We now present numerical results that allow us to identify certain properties of L_p-norm regularization and thus understand its effect. We have implemented our multiresolution algorithm using GlobalBioIm [123], a MATLAB library designed for solving inverse problems.

3.6.1 Setup

In our experiments, we have considered two types of measurement functionals.

- *Dirac Impulses:* In this setting, the given measurement operator takes the form $\boldsymbol{\nu}(s) = \Big(\langle \delta(\cdot - x_1), s\rangle, \dots, \langle \delta(\cdot - x_M), s\rangle\Big) = \Big(s(x_1), \dots, s(x_M)\Big)$, where the points $\{x_m\}_{m=1}^M$ lie within the interval I_T. This operator corresponds to the standard interpolation problem that was discussed in Section 3.1. We ensure that the points $\{x_m\}_{m=1}^M$ are pairwise distinct and that $M \geq N_0$, so that the operator $\boldsymbol{\nu}$ satisfies the condition $\mathcal{N}_{\mathrm{D}^{N_0}} \cap \mathcal{N}_{\boldsymbol{\nu}} = \{0\}$.

- *Dephased Cosines:* In this case, the measurement functionals are $\nu_1 = \mathbb{1}_{[0,T]}$ and $\nu_m = \cos(\omega_m x + \theta_m) \times \mathbb{1}_{[0,T]}$ for $m = \{2, 3, \dots, M\}$. This operator corresponds to a variant of the Fourier interpolation problem which is relevant to magnetic resonance imaging. In order to construct such an operator and the corresponding measurements for our experiments, we first generated a function s_0 and picked a threshold frequency $\omega_{\max}$ such that the spectrum of s_0 had little energy above $\omega_{\max}$. The frequencies ω_m were then drawn uniformly at random from $(0, \omega_{\max}]$ while the phases θ_m were drawn uniformly at random from $[0, \pi)$. This operator $\boldsymbol{\nu}$ was applied to s_0 to generate the measurements that we use in the experiments involving dephased cosines.

The regularization parameter was set to $\tau = 10^{-10}$ in the first two experiments and $\tau = 10^{-15}$ in the last two experiments. For all examples that we present in this section, the grid tolerance was set to $\epsilon = 10^{-3}$. In each example, we compute the solution for several values of $p \in [1, \infty]$.

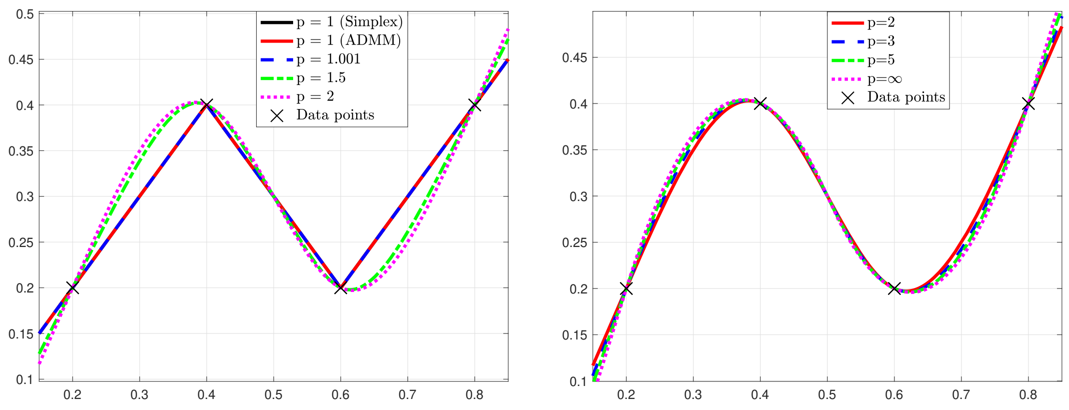


Figure 3.4: Unique gTV solution (L = D^2). The simplex and ADMM solutions are coincident.

3.6.2 Results

1) Continuum of Solutions and Sparsity: We first present two examples (Figures 3.4 and 3.5) to talk about the behavior of the solution as the value of p is changed. In these examples, the measurement functionals are Dirac impulses (interpolation problem) and the regularization operator is $\mathrm{L} = \mathrm{D}^2$. Both examples show that, as we vary p from ∞ to 1 (note that $p = 1$ corresponds to the gTV case), the solutions gradually move towards the (or one of the) gTV solution(s). For the example in Figure 3.4, the computed gTV solutions with and without applying the simplex are the same and resemble a linear spline with two knots, in agreement with [101]. It can be shown that this particular sparse solution is the unique solution to the gTV problem. In this case, we see that the solution for $p = 1.001$ is close to the unique sparse gTV solution.

By contrast, the configuration of the data points in Figure 3.5 is such that the gTV problem has multiple solutions. This can be seen in the plots as the solution obtained by running the simplex after ADMM is sparse (linear spline with three knots), while the solution obtained via ADMM only is non-sparse. Interestingly in this case, the solution for $p = 1.001$ is close to a non-sparse gTV solution. Based on the above observations and additional experiments of the same nature, we make several claims.

- There exists a continuum of solutions when p is varied from ∞ to 1.
- When the gTV problem has a unique solution, the continuum converges to that unique sparse solution as $p \to 1$.
- When the gTV problem has multiple solutions, the continuum converges to one of its non-sparse solutions as $p \to 1$.

We discuss two implications of our claims. Firstly, the existence of a continuum implies

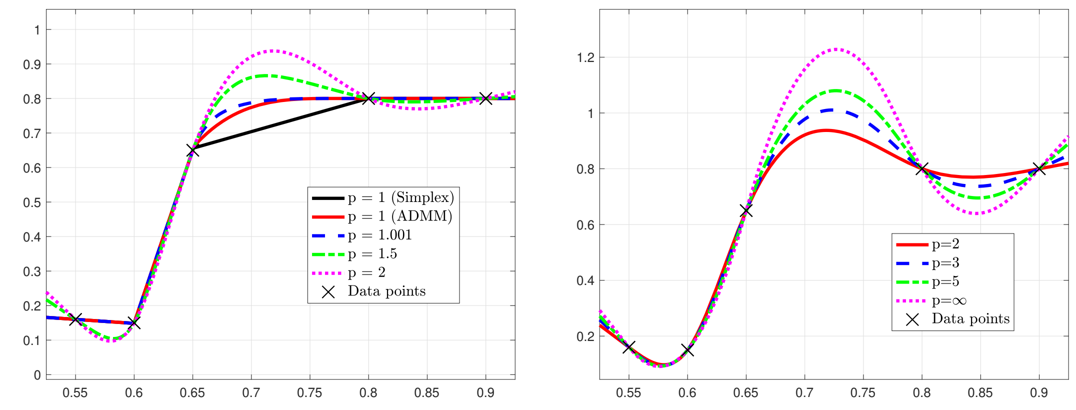


Figure 3.5: Multiple gTV solutions ($\mathrm{L} = \mathrm{D}^2$).

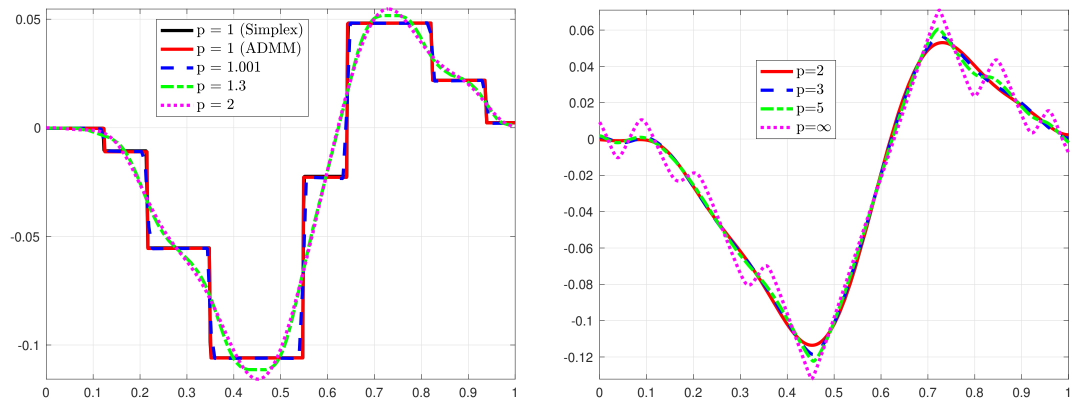


Figure 3.6: Dephased-cosine measurement functionals ($\mathrm{L} = \mathrm{D}, \quad M = 15$). For $p = 1$, the simplex and ADMM solutions are coincident.

that one can use L_p-norm regularization with $p \in (1, \infty)$, to “interpolate" between the properties of the gTV and L_∞ solutions. One such property is regularity or smoothness. In Figures 3.4 and 3.5, we observe that the smoothness of the solution reduces as p decreases. Secondly, we conclude that L_p-norm regularization with a small p can be used as a sparsity-promoting prior in settings where the gTV solution is guaranteed to be unique. This is in line with the use of discrete ℓ_p-norm regularization, with a small p, in compressed-sensing frameworks.

As further illustration, we also provide an example with the dephased-cosine measurement functionals. In this case, the regularization operator was $\mathrm{L} = \mathrm{D}$, leading to a piecewise-constant gTV solution in Figure 3.6. The continuum of solutions and change in regularity, as p is varied from ∞ to 1, is evident in this figure.

2) Gibbs-Like Oscillations: In the interpolation of step-like functions using splines, Gibbs-like oscillations are observed at the discontinuities [112, 124, 125]. We use the step and staircase functions (Figure 3.7) to investigate this effect in our L_p-regularized problem.

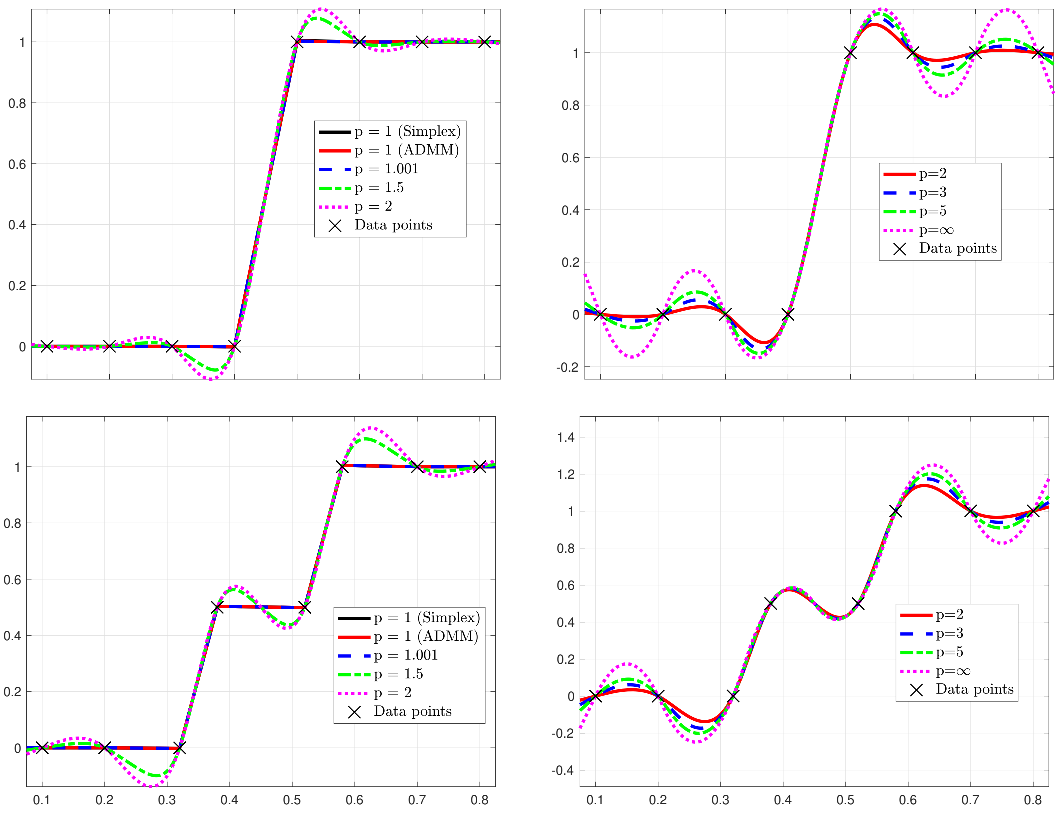


Figure 3.7: Illustration of Gibbs-like oscillations ($\mathrm{L} = \mathrm{D}^2$). For $p = 1$, the simplex and ADMM solutions are coincident.

In these cases, we observe that the solutions exhibit an oscillatory behavior (with an overshoot at the discontinuity) which decreases as p goes from ∞ to 1. Moreover, as p becomes smaller, the oscillatory effect of the discontinuity becomes more localized. We claim that

- L_p-norm regularization with a smaller p results in weaker Gibbs-like oscillations at the edges.

We would like to point out that the above claims exclude the special case of spatial interpolation with $\mathrm{L} = \mathrm{D}$. Here, all values of $p \in (1, \infty)$ generate the same solution, which is a linear spline with knots at the data points. This can be inferred from the theoretical result stated in Section 3.2.

3) Effect of N_0: We now discuss the influence of the operator $\mathrm{L} = \mathrm{D}^{N_0}$ which is the second component of our regularization term. In Figure 3.8, we present an example where we fix $p = 1.5$ and compute the solutions for different values of N_0. Our general observation is that

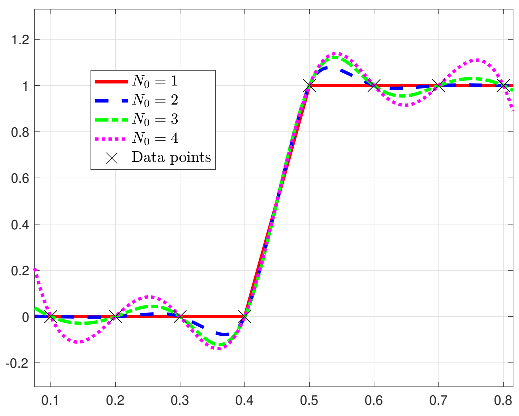


Figure 3.8: Effect of the regularization operator D^{N_0} for a fixed $p = 1.5$.

- For any $p \in [1, \infty]$, the solution becomes smoother and exhibits more oscillations as N_0 increases.

4) Comparison with Shannon's sinc interpolation: Consider a standard interpolation problem with uniformly spaced points

$$x_m = m\Delta, \quad m = 1, 2, \dots, M, \tag{3.47}$$

where $\Delta > 0$ is the spacing between any two consecutive points x_m, and measurements $\{y_m\}_{m=1}^{M}$. In this case, the well-known sinc interpolant is given by

$$s_{\mathrm{sinc}}(x) = \sum_{m=1}^{M} y_m \operatorname{sinc}\left(\frac{x - m\Delta}{\Delta}\right). \tag{3.48}$$

Remarkably, the variational formulation (3.12) of the above interpolation problem includes Shannon's sinc interpolation scheme as a special case corresponding to $p = 2$ and $N_0 \to \infty$ [126].

In many applications such as image scaling and image registration, smoother interpolating functions are desirable since they are well-behaved with well-defined multi-order derivatives. While $s_{\mathrm{sinc}}(x)$ is a highly regular function, unfortunately it also exhibits strong Gibbs-like oscillations at sharp transitions. On the other hand, as observed in the previous experiments, by controlling the values of p and N_0, L_p-regularized solutions can be made to achieve a balance between smoothness and oscillatory behaviour.

To illustrate this advantage of our framework, we consider interpolation of the data points from Figure 3.8. We compute the maximum overshoot (which is related to the extent of the oscillations) of the sinc interpolant and the L_p-regularized interpolant for several values of p and N_0, and we plot the results in Figure 3.9. For ease of comparison, we indicate the maximum overshoot for sinc interpolation, which is quite high, as a horizontal

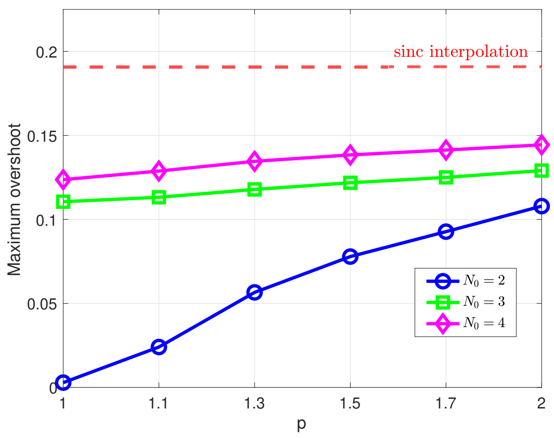


Figure 3.9: Maximum overshoot values for interpolation of the data points from Figure 3.8.

dashed line. The plots for the L_p-regularized solutions show that N_0 and p (more so when N_0 is small) can be varied to control the overshoots or oscillations, and balance them with the desired smoothness.

3.7 Summary

We have implemented a multiresolution algorithm to solve numerically the generalized-interpolation problem with L_p-norm regularization, along with its unconstrained variants. We have shown that an appropriate grid-based B-spline basis can be used to exactly discretize the (restricted) continuous-domain problem. Based on previous results from approximation theory and splines, we have argued that as the grid size goes to zero, the computed solution approaches the solution of the unrestricted continuous-domain problem. With the help of numerical results in the context of spatial and Fourier interpolation, we have established the existence of a continuum of solutions as p goes from ∞ to 1. Finally, we have made insightful observations about properties of the L_p-regularized solutions such as sparsity, regularity, and Gibbs-like oscillations.

3.8 Appendix

Consider the unconstrained optimization problem in (3.2):

$$\mathcal{S} = \arg\min_{s\in\mathcal{X}} \Bigg(\underbrace{\mathcal{D}\Big(\mathbf{y}, \boldsymbol{\nu}(s)\Big) + \tau\mathcal{R}(s)}_{\mathcal{J}(s)} \Bigg). \tag{3.49}$$

Here, we show that if $\mathcal{D}$ is strictly convex and $\mathcal{R}$ is convex, then all the solutions $s^* \in \mathcal{S}$ generate the same measurement vector $\mathbf{z}_0 = \boldsymbol{\nu}(s^*)$. The proof is adapted from [127] and is based on standard arguments in convex analysis.

Let $s_1^*, s_2^* \in \mathcal{S}$ be two solutions of (3.49) such that they produce different measurements *i.e.,* $\boldsymbol{\nu}(s_1^*) \neq \boldsymbol{\nu}(s_2^*)$. Let the minimum value of the objective function be $\mathcal{J}^* = \mathcal{J}(s_1^*) = \mathcal{J}(s_2^*)$. For a candidate function $s_c = \alpha s_1^* + (1-\alpha)s_2^*$, with $\alpha \in (0,1)$, we have

$$\begin{aligned}\mathcal{J}(s_c) &= \mathcal{D}\Big(\mathbf{y}, \boldsymbol{\nu}\Big(\alpha s_1^* + (1-\alpha)s_2^*\Big)\Big) + \tau\mathcal{R}\Big(\alpha s_1^* + (1-\alpha)s_2^*\Big) \\ &< \Bigg(\alpha\Bigg(\underbrace{\mathcal{D}\Big(\mathbf{y}, \boldsymbol{\nu}(s_1^*)\Big) + \tau\mathcal{R}(s_1^*)}_{\mathcal{J}^*}\Bigg) \\ &\qquad + (1-\alpha)\underbrace{\mathcal{D}(\mathbf{y}, \boldsymbol{\nu}(s_2^*)) + \tau\mathcal{R}(s_2^*)}_{\mathcal{J}^*}\Bigg) = \mathcal{J}^*. \end{aligned} \tag{3.50}$$

The above strict inequality is due to the fact that $\mathcal{D}$ is strictly convex and $\mathcal{R}$ is convex. The relation $\mathcal{J}(s_c) < \mathcal{J}^*$ is a contradiction and thus $\boldsymbol{\nu}(s_1^*) = \boldsymbol{\nu}(s_2^*) = \mathbf{z}_0$.

4 Sparse Stochastic Processes

[1]In this chapter, we present a benchmarking environment based on sparse stochastic processes [114] to objectively evaluate and compare the performance of reconstruction algorithms for linear inverse problems involving 1D signals. Our framework offers quantitative measures of the degree of optimality (in the mean-square-error sense) for any given reconstruction method. Since it is based on stochastic modelling, it provides access to unlimited amounts of data, which enables the proper benchmarking of NN-based approaches without having to worry about the representativity of the training data.

4.1 Introduction

NN-based methods that make use of prior information learned from a large collection of training data are now the focus of much of the current research in signal reconstruction. In several applications such as MRI, CT, optical imaging, and ultrasound, their gain over state-of-the-art classical methods is impressive. Specifically, they shine in extreme scenarios where one wishes to achieve more with fewer measurements. However, NN-based methods have certain limitations that currently hinder their further development.

Unlike the classical methods, which are backed by sound mathematics, the development of NN-based approaches is empirical. Expressivity is obtained through the composition of simple units, but the working of the whole is hard to comprehend and the architectural options are overwhelming (*e.g.,* depth, number of channels, size of the filters). In practice, one usually proceeds by trial and error using the training, validation, and testing errors as quantitative criteria. Further, the training of NNs is poorly understood and often difficult because of the underlying over-parameterization: getting a stochastic optimization algorithm to perform properly for a specific application typically requires a lot of adjustments and experimentation.

[1]This chapter is based on our work [128].

Beside the strain that this empirical approach exerts on developers, the performance greatly depends on the quality, cardinality, and representativity of the training dataset. The bottleneck with several applications (e.g., biomedical imaging) is often a limited access to large, representative datasets. This explains why the works that demonstrate the superiority of the NN-based approaches over the classical ones for signal reconstruction have used limited benchmarks so far.

4.1.1 Contributions

In this chapter, we present an objective environment to benchmark the performance of reconstruction algorithms for linear inverse problems, in particular, NN-based methods that require large amounts of training data.

We synthesize ground-truth signals and then simulate the measurement process (*e.g.,* convolution for deconvolution microscopy, Fourier sampling for MRI) in the presence of noise. Specifically, we consider a statistical framework where the underlying signals are realizations of 1D sparse stochastic processes (SSPs) [114]. Since the true statistical distribution of the signal is known exactly in our framework, the minimum-mean-square-error (MMSE) estimator is indeed optimal in the mean-square-error (MSE) sense. Therefore, we are able to provide statistical guarantees of optimality by specifying an upper limit on the reconstruction performance.

Our framework also provides training data for NN-based methods. Indeed, for some chosen stochastic signal model, we can produce datasets consisting of any desired number of signals or signal-measurement pairs for a given measurement model, which allows for an informed comparison of network architectures. Thus, the availability of the goldstandard (MMSE estimator) and training data make our benchmark a good ground for the tuning of NN architectures and for the identification of the best designs in a tightly controlled environment.

The MAP estimates of SSPs are solutions of optimization problems and can be computed efficiently. However, it has been observed that these MAP estimators are suboptimal in the MSE sense [34, 129], except in the Gaussian scenario where the MAP and MMSE estimators (generalized Wiener filter) coincide [15]. In this work, we focus on non-Gaussian signal models. In principle, the MMSE estimator involves the calculation of high-dimensional integrals, which are not numerically tractable in general. Thus, we develop efficient Gibbs-sampling-based algorithms to compute the MMSE estimators for specific classes of SSPs, with innovations following the Laplace, Student's t, and Bernoulli-Laplace distributions. To the best of our knowledge, no such working solution for generic linear inverse problems with SSPs has been presented in the literature.

Finally, we present experimental results that illustrate the usefulness of our framework. Specifically, we benchmark the performance of some well-known classical MPL estimators and end-to-end trained CNNs that perform direct nonlinear reconstructions, in the context of deconvolution and Fourier sampling for first-order SSPs. The CNNs that we consider are optimized by minimizing the MSE loss for training datasets. On one hand, when the innovations follow a Bernoulli-Laplace distribution, we observe that CNNs (with sufficient capacity and training data) outperform the sparsity-promoting MPL estimators, which are well-suited to these piecewise-constant signals. In fact, some of these CNNs achieve near-optimal MSE performance. On the other hand, our experiments with Student's t innovations indicate regimes where CNNs fail to reconstruct the signals well. More specifically, we observe that, when the tails of the Student's t distribution are made heavier (*i.e.,* when we move towards a Cauchy distribution), CNNs perform rather poorly.

The chapter is organized as follows: In Section 4.2, we describe a continuous-domain measurement model along with a way to discretize it. In Section 4.3, we introduce Lévy processes as stochastic models for our signals and we derive the probability distribution for samples of such processes. We then develop Gibbs samplers for Lévy processes associated with Laplace, Student's t, and Bernoulli-Laplace distributions in Section 4.4. Finally, we present experimental results in Section 4.5.

4.2 Measurement Model

In the proposed framework, we consider the recovery of a continuous-domain signal $s^\dagger : \mathbb{R} \to \mathbb{R}$ from a finite number M of measurements $\mathbf{y}^\dagger = (y^\dagger_m)_{m=1}^M$.

4.2.1 Continuous-Domain Measurement Model

We model the measurements $\mathbf{y}^\dagger = (y^\dagger_m)_{m=1}^M$ as

$$y^\dagger_m = \int_{\mathbb{R}} s^\dagger(t)\nu_m(t)\mathrm{d}t + n^\dagger[m], \tag{4.1}$$

where $(\nu_m)_{m=1}^M$ are linear functionals that describe the physics of the acquisition process and $n^\dagger[\cdot]$ is a realization of white Gaussian noise with variance σ_n^2. By choosing appropriate functionals $(\nu_m)_{m=1}^M$, we can study a variety of linear inverse problems such as denoising, deconvolution, inpainting, and Fourier sampling.

4.2.2 Discrete Measurement Model

We need to discretize (4.1) to obtain a computationally feasible model for the measurements. To that end, we consider a finite region of interest $\Omega = (0, T)$ of the signal and approximate it with

$$s_h^\dagger(t) = \sum_{k=1}^{K} s^\dagger(kh)\mathrm{sinc}\left(\frac{t}{h} - k\right), \tag{4.2}$$

where h is the sampling step and $K = \left(\lfloor \frac{T}{h} \rfloor - 1\right)$. When h is small enough, $s_h^\dagger$ is a good approximation of $s^\dagger$ within the interval Ω [130]. On introducing (4.2) into (4.1), we get that

$$\mathbf{y}^\dagger = \mathbf{H}\mathbf{s}^\dagger + \mathbf{n}^\dagger, \tag{4.3}$$

where $\mathbf{s}^\dagger = (s^\dagger(kh))_{k=1}^{K} \in \mathbb{R}^K$ contains equidistant samples of the signal, $\mathbf{H} : \mathbb{R}^K \to \mathbb{R}^M$ is the discrete system matrix with

$$[\mathbf{H}]_{m,k} = \int_{\mathbb{R}} \mathrm{sinc}\left(\frac{t}{h} - k\right)\nu_m(t)\mathrm{d}t, \tag{4.4}$$

and $\mathbf{n}^\dagger \in \mathbb{R}^M$ is the noise.

Thus, for any given signal samples $\mathbf{s}^\dagger \in \mathbb{R}^K$, we can simulate noisy measurements using (4.3). Next, we derive the discrete system matrices for deconvolution and Fourier sampling. Hereafter, we assume for simplicity that $h = 1$.

4.2.3 Deconvolution

In deconvolution, the measurements are acquired by sampling the result of the convolution between the signal and the point-spread function (PSF) ψ of the acquisition system, which we model by letting the measurement functionals be $\nu_m = \psi(m - \cdot)$. We assume that the cutoff frequency of ψ is $\omega_0 \leq \pi$, as this allows us to sample $(s^\dagger * \psi)$ on an integer grid without aliasing effects. In this case, The entries of the resulting system matrix $\mathbf{H}$ are given by

$$\begin{aligned}[\mathbf{H}]_{m,k} &= \int_{\mathbb{R}} \mathrm{sinc}(t - k)\psi(m - t)\mathrm{d}t \\ &= \psi(m - k).\end{aligned} \tag{4.5}$$

Here, $\mathbf{H}$ is a discrete convolution matrix whose entries are samples of the bandlimited PSF ψ.

4.2.4 Fourier Sampling

In Fourier sampling, the measurements are acquired by sampling the Fourier transform of the signal at arbitrary frequencies $\{\omega_m\}_{m=1}^{M}$. Accordingly, the measurement functionals are the complex exponentials $\nu_m = \mathrm{e}^{-\mathrm{j}\omega_m \cdot}$. Assuming that $|\omega_m| \leq \pi$, we get that

$$\begin{aligned}[\mathbf{H}]_{m,k} &= \int_{\mathbb{R}} \mathrm{sinc}(t-k)\mathrm{e}^{-\mathrm{j}\omega_m t}\mathrm{d}t \\ &= \mathrm{e}^{-\mathrm{j}\omega_m k}. \end{aligned} \tag{4.6}$$

Here, $\mathbf{H}$ is a discrete Fourier-like matrix, except that the frequencies ω_m do not necessarily lie on an uniform grid.

4.3 Stochastic Signal Model

In this section, we describe a continuous-domain stochastic model for the signal. We also derive the probability distribution for its samples.

4.3.1 Lévy Processes

In our framework, the underlying signals are realizations of a well-known class of first-order sparse stochastic processes: the Lévy processes [114, 131].

Definition 4.1 (Lévy process)**.** *A stochastic process (or collection of random variables)* $\mathsf{S} = \{\mathsf{S}(t) : t \in \mathbb{R}^+\}$ *is a Lévy process if*

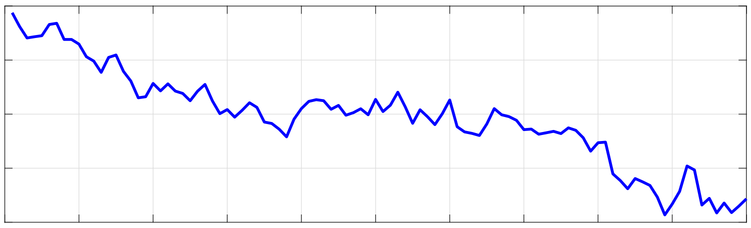
1. $\mathsf{S}(0) = 0$ *almost surely;*
2. *(independent increments) for any* $N \in \mathbb{N} \setminus \{0, 1\}$ *and* $0 \leq t_1 < t_2 \cdots < t_N < \infty$*, the increments* $\big(\mathsf{S}(t_2) - \mathsf{S}(t_1)\big), \big(\mathsf{S}(t_3) - \mathsf{S}(t_2)\big), \ldots, \big(\mathsf{S}(t_N) - \mathsf{S}(t_{N-1})\big)$ *are mutually independent;*
3. *(stationary increments) for any given step* h*, the increment process* $\mathsf{U}_h = \{\mathsf{S}(t) - \mathsf{S}(t-h) : t \in \mathbb{R}^+\}$ *is stationary;*
4. *(stochastic continuity) for any* $\epsilon > 0$ *and* $t \geq 0$

$$\lim_{h \to 0} \Pr\{|\mathsf{S}(t+h) - \mathsf{S}(t)| > \epsilon\} = 0.$$

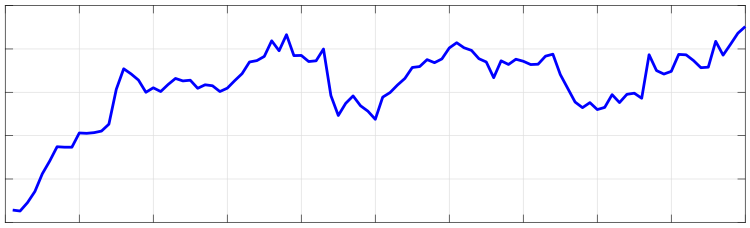
Lévy processes are closely linked to infinitely divisible (id) distributions.

Definition 4.2 (Infinite divisibility)**.** *A random variable* X *is infinitely divisible if, for any* $N \in \mathbb{N} \setminus \{0\}$*, there exist independent and identically distributed (i.i.d.) random variables* $\mathsf{X}_1, \ldots, \mathsf{X}_N$ *such that* $\mathsf{X} = \mathsf{X}_1 + \cdots + \mathsf{X}_N$*.*

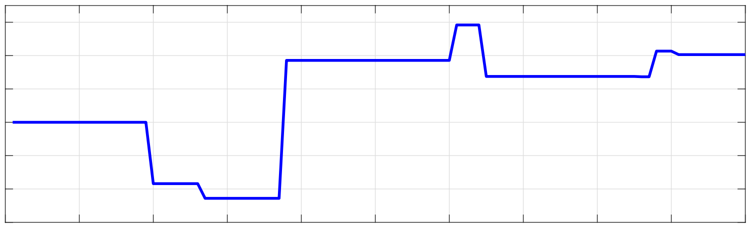
For any Lévy process S, the random variable $\mathsf{S}(t)$ for some $t > 0$ is infinitely divisible.



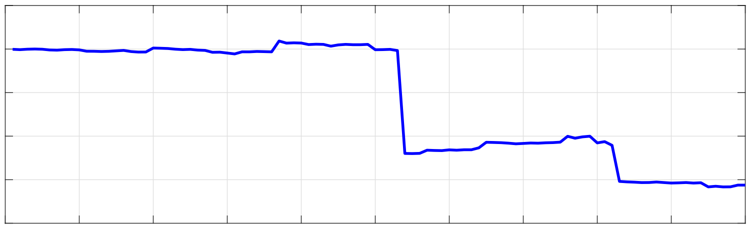
(a) Gaussian: $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$



(b) Laplace: $p(x) = \frac{b}{2} e^{-b|x|}$



(c) Bernoulli-Laplace: $p(x) = \lambda\delta(x) + (1-\lambda)\frac{b}{2} e^{-b|x|}$



(d) Student's t: $p(x) = \frac{\Gamma(\frac{\alpha+1}{2})}{\Gamma\left(\frac{\alpha}{2}\right)} \frac{1}{\sqrt{\pi}(1+x^2)^{\frac{\alpha+1}{2}}}$

Figure 4.1: Realizations of different Lévy processes as characterized by the corresponding infinitely divisible pdfs.

Moreover, its probability density function (pdf) is given by

$$p_{\mathsf{s}(t)}(x) = \int_{\mathbb{R}} \left(\int_{\mathbb{R}} p_{\mathsf{s}(1)}(y) \mathrm{e}^{\mathrm{j}\omega y} \mathrm{d}y \right)^t \mathrm{e}^{-\mathrm{j}\omega x} \frac{\mathrm{d}\omega}{2\pi}. \tag{4.7}$$

Conversely, for any id distribution with pdf p_{id}, it is possible to construct a Lévy process S such that $p_{\mathsf{S}(1)} = p_{\mathrm{id}}$. Thus, there is a one-to-one correspondence between Lévy processes and id distributions [131].

Among all id distributions, the pdf of the Gaussian distribution exhibits the fastest rate of decay at infinity. In this sense, we refer to the non-Gaussian, heavier-tailed members (*e.g.,* Laplace, Bernoulli-Laplace, Student's t, symmetric-alpha-stable) of the class of id distributions as sparse [132]. Indeed, some of these sparse distributions have a mass at the origin in their probability distribution (*e.g.,* Bernoulli-Laplace) and some of them are strongly compressible (*e.g.,* Student's t, symmetric-alpha-stable) [133].

The stochastic model of Lévy processes allows us to consider a variety of signals with different types of sparsity. In our framework, we focus on the subclass of Lévy processes associated with the Gaussian, Laplace, Bernoulli-Laplace and Student's t distributions. Some realizations of these processes are shown in Figure 4.1.

4.3.2 Discrete Stochastic Model

Now, we derive the pdf of the random vector $\mathbf{S} = (\mathsf{S}(k))_{k=1}^{K}$, which contains uniform samples of a Lévy process. Consider the stationary increment process $\mathsf{U}(t) = \{\mathsf{S}(t) - \mathsf{S}(t-1) : t \in \mathbb{R}^{+}\}$ whose first-order pdf p_{U} is the same as $p_{\mathsf{S}(1)}$ and so is infinitely divisible. Its samples $\mathbf{U} = (\mathsf{U}(k))_{k=1}^{K}$ can be expressed as

$$\mathbf{U} = \mathbf{D}\mathbf{S}, \tag{4.8}$$

where $\mathbf{D}$ is a finite-difference matrix of the form

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ & & \ddots & \ddots & \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix}. \tag{4.9}$$

Using (4.8) and the fact that the increments are independent, we obtain the pdf of the discrete signal as

$$p_{\mathbf{S}}(\mathbf{s}) = \prod_{k=1}^{K} p_{\mathsf{U}}\Big([\mathbf{D}\mathbf{s}]_k\Big). \tag{4.10}$$

Note that (4.8) can also be written as

$$[\mathbf{S}]_k = \sum_{n=1}^{k} [\mathbf{U}]_n, \qquad k = 1, \dots, K, \tag{4.11}$$

which gives us a direct way to generate samples of Lévy processes.

4.3.3 Extensions

In this work, we have considered inverse problems involving 1D signals that are modelled as realizations of Lévy processes with increments that follow the Gaussian, Laplace, Bernoulli-Laplace and Student's t distributions. Our framework can further be extended in a straightforward manner to include the more general signal model of continuous-domain first-order autoregressive processes [114, Chapter 7] driven by white noises associated with the aforementioned distributions. These AR(1) processes yield a discrete stochastic model that is similar to the one described in (4.10). There, the application of a suitable transformation matrix to the random vector containing equidistant samples of the process decouples it and generates a random vector (called the innovation or generalized increments) with i.i.d. entries. Thus, the MMSE estimation methods presented in Section 4.4 can be readily adapted for such AR(1) processes.

We can also directly extend the proposed framework to handle multidimensional signals for the particular stochastic model of continuous-domain AR Lévy sheets [134, Chapter 3], [114] associated with the Gaussian, Laplace, Bernoulli-Laplace and Student's t distributions. These are higher-dimensional generalizations (based on separable whitening operators) of the corresponding AR(1) processes and they result in desirable discrete models of the form (4.10). Unfortunately, the random vectors constructed from samples of other ("non-separable") higher-dimensional stochastic processes described in [114] cannot be fully decoupled by applying a linear transformation. This makes the task of designing schemes to evaluate their MMSE estimators very challenging. An alternate way of extending our framework could be to define a new class of continuous-domain multidimensional stochastic models using the spline-operator-based framework of [135, 136]. However, this approach would require substantial development of novel mathematical ideas and is thus not discussed further in this work.

4.4 MMSE Estimators for Sparse Lévy Processes

So far, we have introduced the signal and measurement models that allow us to generate our ground-truth signals and simulate their noisy measurements for a certain acquisition setup. Next, we focus on the MMSE estimator for the reconstruction problem at hand, which is to recover the signal $\mathbf{s}^\dagger$ from its measurements $\mathbf{y}^\dagger$.

Let $\mathbf{Y}$ be the underlying random vector for the measurements that takes values in $\mathbb{R}^M$. Since we have an AWGN model, using Bayes' rule, (4.3) and (4.10), we can write the pdf of the posterior distribution of $\mathsf{S}|\mathsf{Y} = \mathbf{y}^\dagger$ as

$$p_{\mathsf{S}|\mathsf{Y}}(\mathbf{s}|\mathbf{y}^\dagger) = \frac{p_{\mathsf{Y}|\mathsf{S}}(\mathbf{y}^\dagger|\mathbf{s})p_{\mathsf{S}}(\mathbf{s})}{\int_{\mathbb{R}^K} p_{\mathsf{Y}|\mathsf{S}}(\mathbf{y}^\dagger|\tilde{\mathbf{s}})p_{\mathsf{S}}(\tilde{\mathbf{s}})\mathrm{d}\tilde{\mathbf{s}}}$$
$$\propto \exp\left(-\frac{\|\mathbf{y}^\dagger - \mathbf{H}\mathbf{s}\|_2^2}{2\sigma_\mathrm{n}^2}\right)\prod_{k=1}^{K} p_{\mathsf{U}}\Big([\mathbf{D}\mathbf{s}]_k\Big). \tag{4.12}$$

The MMSE estimator is then given by

$$\mathbf{s}^*_{\mathrm{MMSE}}(\mathbf{y}^\dagger) = \arg\min_{\mathbf{s}\in\mathbb{R}^K}\left(\int_{\mathbb{R}^K} \|\tilde{\mathbf{s}} - \mathbf{s}\|_2^2 \; p_{\mathsf{S}|\mathsf{Y}}(\tilde{\mathbf{s}}|\mathbf{y}^\dagger)\mathrm{d}\tilde{\mathbf{s}}\right)$$
$$= \int_{\mathbb{R}^K} \tilde{\mathbf{s}} \; p_{\mathsf{S}|\mathsf{Y}}(\tilde{\mathbf{s}}|\mathbf{y}^\dagger)\mathrm{d}\tilde{\mathbf{s}}, \tag{4.13}$$

which is the mean of the posterior distribution $p_{\mathsf{S}|\mathsf{Y}}(\cdot|\mathbf{y}^\dagger)$. For a fixed stochastic model, the MMSE estimator is the optimal reconstructor in the MSE sense and thus serves as the goldstandard in our benchmarking framework. In the Gaussian case, the MMSE estimator is known to coincide with the MAP estimator and is straightforward to calculate [15, 137]. However, in the non-Gaussian case, we need to numerically evaluate the high-dimensional integral in (4.13), which is computationally challenging.

In the remainder of this section, we present efficient methods to compute the MMSE estimator for sparse Lévy processes with increments that follow the Laplace, Student's t, and Bernoulli-Laplace distributions, which constitutes a key contribution of this chapter.

4.4.1 Gibbs Sampling

In order to compute the integral in (4.13), one can generate samples $\{\mathbf{s}^{\dagger(q)}\}_{q=1}^{Q}$ from the posterior distribution $p_{\mathsf{S}|\mathsf{Y}}(\cdot|\mathbf{y}^\dagger)$ using an MCMC method and approximate $\mathbf{s}^*_{\mathrm{MMSE}}(\mathbf{y}^\dagger)$ by the empirical mean $\mathbf{s}^*_Q(\mathbf{y}^\dagger) = \frac{1}{Q}\sum_{q=1}^{Q}\mathbf{s}^{\dagger(q)}$. In this work, we propose to use the MCMC method called Gibbs sampling [138, 139] to first generate samples $\{\mathbf{u}^{\dagger(q)}\}_{q=1}^{Q}$ from the distribution $p_{\mathsf{U}|\mathsf{Y}}(\cdot|\mathbf{y}^\dagger)$. These can then be transformed in accordance with (4.11) to obtain the desired samples $\{\mathbf{D}^{-1}\mathbf{u}^{\dagger(q)}\}_{q=1}^{Q}$ from $p_{\mathsf{S}|\mathsf{Y}}(\cdot|\mathbf{y}^\dagger)$. We now give the gist of this algorithm.

Let X and Y be two random variables. Consider the task of generating samples from their joint distribution $p_{\mathsf{X},\mathsf{Y}}$. Gibbs sampling is advantageous whenever it is computationally difficult to sample from the joint distribution directly but the conditional distributions $p_{\mathsf{X}|\mathsf{Y}}(\cdot|y)$ and $p_{\mathsf{Y}|\mathsf{X}}(\cdot|x)$ are easy to sample from. The steps involved in this method are presented in Algorithm 2. They yield a Markov chain whose stationary distribution is indeed $p_{\mathsf{X},\mathsf{Y}}$ [139]. In practice, one discards some of the initial samples (burn-in period) to allow the chain to converge. Moreover, quantities (expectation integrals) based on the marginal distributions p_{X} and p_{Y} can be computed from the individual samples $\{x^{(q)}\}_{q=1}^{Q}$

Algorithm 2 Gibbs sampling

```
1: Input: $Q$ (number of samples), $B$ (burn-in period)
2: Initialization: $\left(\tilde{x}^{(0)}, \tilde{y}^{(0)}\right)$
3: for $q = 1, \dots, B + Q$ do
4:     Generate $\tilde{x}^{(q)}$ according to $p_{\mathsf{X}|\mathsf{Y}}\left( \cdot \,|\tilde{y}^{(q-1)}\right)$
5:     Generate $\tilde{y}^{(q)}$ according to $p_{\mathsf{Y}|\mathsf{X}}\left( \cdot \,|\tilde{x}^{(q)}\right)$
6: end for
7: Output: $\left\{\left(x^{(q)}, y^{(q)}\right)\right\}_{q=1}^{Q} = \left\{\left(\tilde{x}^{(q+B)}, \tilde{y}^{(q+B)}\right)\right\}_{q=1}^{Q}$
```

and $\{y^{(q)}\}_{q=1}^{Q}$, respectively.

Next, we present Gibbs sampling schemes for Lévy processes with Laplace, Student's t, and Bernoulli-Laplace increments. Our strategy is to introduce an auxiliary random vector $\mathbf{W}$ and perform Gibbs sampling for the joint distribution $p_{\mathbf{U},\mathbf{W}|\mathbf{Y}}(\cdot,\cdot|\mathbf{y}^{\dagger})$ [140, 141]. The key is to choose $\mathbf{W}$ such that the conditional distributions $p_{\mathbf{U}|\mathbf{W},\mathbf{Y}}(\cdot|\cdot,\mathbf{y}^{\dagger})$ and $p_{\mathbf{W}|\mathbf{U},\mathbf{Y}}(\cdot|\cdot,\mathbf{y}^{\dagger})$ can be sampled from in an efficient manner.

Hereafter, we assume that the noise variance σ_{n}^2 and the parameters of the signal model are known.

4.4.2 Laplace Increments

For Lévy processes with Laplace increments, we adapt the approach that was developed in [142].

The pdf for the Laplace distribution is

$$p_{\mathsf{U}}(u) = \frac{b}{2}\exp\Big(-b|u|\Big), \tag{4.14}$$

where b is the scale parameter. The density in (4.14) can be expressed as a scale mixture of normal distributions [143], as

$$p_{\mathsf{U}}(u) = \int_{\mathbb{R}} p_{\mathsf{U}|\mathsf{W}}(u|w)p_{\mathsf{W}}(w)\mathrm{d}w, \tag{4.15}$$

where

$$p_{\mathsf{U}|\mathsf{W}}(u|w) = \frac{1}{\sqrt{2\pi w}}\exp\left(-\frac{u^2}{2w}\right) \tag{4.16}$$

is the Gaussian pdf and

$$p_{\mathsf{W}}(w) = \frac{b^2}{2}\exp\Bigg(-\frac{b^2 w}{2}\Bigg)\mathbb{1}_+(w) \tag{4.17}$$

is a mixing exponential pdf[2] with $\lambda = 2/b^2$. This property allows us to define an auxiliary random vector $\mathbf{W}$ that takes values in $\mathbb{R}^K$ with i.i.d. entries following the distribution p_{W} in (4.17), such that

$$p_{\mathbf{U}|\mathbf{W}}(\mathbf{u}|\mathbf{w}) = \prod_{k=1}^{K} p_{\mathsf{U}|\mathsf{W}}\Big([\mathbf{u}]_k|[\mathbf{w}]_k\Big), \tag{4.18}$$

where $\mathbf{u}, \mathbf{w} \in \mathbb{R}^K$ and $p_{\mathsf{U}|\mathsf{W}}$ is shown in (4.16).

By the chain rule of probability (or the general product rule), the full joint distribution $p_{\mathbf{Y},\mathbf{U},\mathbf{W}}$ can be written as

$$\begin{aligned} p_{\mathbf{Y},\mathbf{U},\mathbf{W}}(\mathbf{y},\mathbf{u},\mathbf{w}) &= p_{\mathbf{Y}|\mathbf{U},\mathbf{W}}(\mathbf{y}|\mathbf{u},\mathbf{w})p_{\mathbf{U},\mathbf{W}}(\mathbf{u},\mathbf{w}) \\ &= p_{\mathbf{Y}|\mathbf{U}}(\mathbf{y}|\mathbf{u})p_{\mathbf{U}|\mathbf{W}}(\mathbf{u}|\mathbf{w})p_{\mathbf{W}}(\mathbf{w}), \end{aligned} \tag{4.19}$$

where $\mathbf{y} \in \mathbb{R}^M$. Consequently, the distribution $p_{\mathbf{U},\mathbf{W}|\mathbf{Y}}$ takes the form

$$\begin{aligned} p_{\mathbf{U},\mathbf{W}|\mathbf{Y}}(\mathbf{u},\mathbf{w}|\mathbf{y}) \propto\ &\exp\Bigg(-\frac{1}{2\sigma_{\mathrm{n}}^2}\|\mathbf{y}-\mathbf{A}\mathbf{u}\|_2^2\Bigg) \times \prod_{k=1}^{K}[\mathbf{w}]_k^{-\frac{1}{2}}\exp\Bigg(-\frac{[\mathbf{u}]_k^2}{2[\mathbf{w}]_k}\Bigg) \\ &\times \prod_{k=1}^{K}\frac{b^2}{2}\exp\Bigg(-\frac{b^2[\mathbf{w}]_k}{2}\Bigg)\mathbb{1}_+\Big([\mathbf{w}]_k\Big), \end{aligned} \tag{4.20}$$

where $\mathbf{A} \coloneqq \mathbf{H}\mathbf{D}^{-1}$.

Based on (4.20), the conditional distribution $p_{\mathbf{U}|\mathbf{W},\mathbf{Y}}$ is then obtained as

$$p_{\mathbf{U}|\mathbf{W},\mathbf{Y}}(\mathbf{u}|\mathbf{w},\mathbf{y}) \propto\ \exp\Bigg(-\frac{1}{2}\Bigg(\frac{1}{\sigma_{\mathrm{n}}^2}\|\mathbf{y}-\mathbf{A}\mathbf{u}\|_2^2 + \mathbf{u}^T\mathbf{C}_{\mathrm{L}}(\mathbf{w})\mathbf{u}\Bigg)\Bigg), \tag{4.21}$$

where $\mathbf{C}_{\mathrm{L}}(\mathbf{w})$ is a diagonal matrix with elements $\Big([\mathbf{w}]_k^{-1}\Big)_{k=1}^{K}$. Specifically, $p_{\mathbf{U}|\mathbf{W},\mathbf{Y}}(\cdot|\mathbf{w},\mathbf{y})$ is a multivariate Gaussian density with mean $\overline{\mathbf{u}} = \sigma_{\mathrm{n}}^{-2}\Big(\sigma_{\mathrm{n}}^{-2}\mathbf{A}^T\mathbf{A} + \mathbf{C}_{\mathrm{L}}(\mathbf{w})\Big)^{-1}\mathbf{A}^T\mathbf{y}$ and covariance matrix $\overline{\mathbf{R}} = \Big(\sigma_{\mathrm{n}}^{-2}\mathbf{A}^T\mathbf{A} + \mathbf{C}_{\mathrm{L}}(\mathbf{w})\Big)^{-1}$. There exist several methods for the efficient generation of samples from a multivariate Gaussian density [144–147].

[2]The pdf of the exponential distribution is

$$p_{\exp}(x) = (1/\lambda)\mathrm{e}^{-x/\lambda}\mathbb{1}_+(x),$$

where $\lambda > 0$ is the scale parameter.

The conditional distribution $p_{\mathsf{W}|\mathsf{U},\mathsf{Y}}$ is

$$p_{\mathsf{W}|\mathsf{U},\mathsf{Y}}(\mathbf{w}|\mathbf{u},\mathbf{y}) \propto \prod_{k=1}^{K} p_{\mathsf{W}|\mathsf{U},\mathsf{Y}}\Big([\mathbf{w}]_k|[\mathbf{u}]_k,\mathbf{y}\Big), \tag{4.22}$$

where

$$p_{\mathsf{W}|\mathsf{U},\mathsf{Y}}\Big(w|u,\mathbf{y}\Big) \propto \exp\left(-\frac{1}{2}\left(\frac{u^2}{w}+b^2 w\right)\right) \times w^{-\frac{1}{2}}\mathbb{1}_{+}(w) \tag{4.23}$$

belongs to the family of generalized inverse Gaussian distributions[3] with $\lambda_1 = b^2$, $\lambda_2 = u^2$ and $a = 0.5$. We can rely on the method proposed in [148] to draw samples from the pdf in (4.23).

To conclude, with the help of the conditional distributions derived in (4.21) and (4.23), we construct a blocked Gibbs sampler, where at at each iteration q, we generate $\mathbf{u}^{\dagger(q)}$ according to $p_{\mathsf{U}|\mathsf{W},\mathsf{Y}}\Big(\cdot|\mathbf{w}^{\dagger(q-1)},\mathbf{y}^{\dagger}\Big)$ and $[\mathbf{w}^{\dagger(q)}]_k$ according to $p_{\mathsf{W}|\mathsf{U},\mathsf{Y}}\Big(\cdot|[\mathbf{u}^{\dagger(q)}]_k,\mathbf{y}^{\dagger}\Big)$ for all $k \in \{1,\dots,K\}$. The collected samples $\{\mathbf{u}^{\dagger(q)}\}_q$ follow the desired distribution $p_{\mathsf{U}|\mathsf{Y}}(\cdot|\mathbf{y}^{\dagger})$.

4.4.3 Student's t Increments

The case of Student's t increments can be handled by adapting the method shown in [149], which is in fact similar to the one we described for Laplace increments.

The Student's t pdf is given by

$$p_{\mathsf{U}}(u) = \frac{\Gamma(\frac{\alpha+1}{2})}{\Gamma\Big(\frac{\alpha}{2}\Big)}\frac{1}{\sqrt{\pi}(1+u^2)^{\frac{\alpha+1}{2}}}, \tag{4.24}$$

where α is the number of degrees of freedom and controls the tail of the distribution, and where Γ denotes the gamma function. It can also be expressed as

$$p_{\mathsf{U}}(u) = \int_{\mathbb{R}} p_{\mathsf{U}|\mathsf{W}}(u|w)p_{\mathsf{W}}(w)\mathrm{d}w, \tag{4.25}$$

where

$$p_{\mathsf{U}|\mathsf{W}}(u|w) = \sqrt{\frac{w}{2\pi}}\exp\left(-\frac{wu^2}{2}\right) \tag{4.26}$$

[3]The pdf of the generalized inverse Gaussian distribution is

$$p_{\mathrm{gig}}(x) = \frac{(\lambda_1/\lambda_2)^{a/2}}{2K_a(\sqrt{\lambda_1\lambda_2})}x^{a-1}\mathrm{e}^{-(\lambda_1 x+\lambda_2/x)/2}\mathbb{1}_{+}(x),$$

where K_a is the modified Bessel function of the second kind, $\lambda_1 > 0$, $\lambda_2 > 0$, and $a \in \mathbb{R}$.

is a Gaussian pdf and

$$p_{\mathsf{W}}(w) = \frac{(0.5)^{\frac{\alpha}{2}}}{\Gamma(\frac{\alpha}{2})} w^{\frac{\alpha}{2}-1} \exp\left(-\frac{w}{2} \right) \mathbb{1}_+(w) \tag{4.27}$$

is the pdf of a gamma[4] distribution. Again, we introduce an auxiliary random vector $\mathsf{\mathbf{W}}$ that takes values in $\mathbb{R}^K$ whose i.i.d. entries follow p_{W} defined in (4.27). It is such that

$$p_{\mathsf{\mathbf{U}}|\mathsf{\mathbf{W}}}(\mathbf{u}|\mathbf{w}) = \prod_{k=1}^{K} p_{\mathsf{U}|\mathsf{W}}\Big([\mathbf{u}]_k | [\mathbf{w}]_k\Big), \tag{4.28}$$

where $\mathbf{u}, \mathbf{w} \in \mathbb{R}^K$ and $p_{\mathsf{U}|\mathsf{W}}$ is defined in (4.26).

Here, the distribution $p_{\mathsf{\mathbf{U}},\mathsf{\mathbf{W}}|\mathsf{\mathbf{Y}}}$ is given by

$$\begin{aligned} p_{\mathsf{\mathbf{U}},\mathsf{\mathbf{W}}|\mathsf{\mathbf{Y}}}(\mathbf{u}, \mathbf{w}|\mathbf{y}) \propto \ & \exp\left(-\frac{1}{2\sigma_{\mathrm{n}}^2} \|\mathbf{y} - \mathbf{A}\mathbf{u}\|_2^2 \right) \times \prod_{k=1}^{K} [\mathbf{w}]_k^{\frac{1}{2}} \exp\left(-\frac{[\mathbf{w}]_k [\mathbf{u}]_k^2}{2} \right) \\ & \times \prod_{k=1}^{K} [\mathbf{w}]_k^{\frac{\alpha}{2}-1} \exp\left(-\frac{[\mathbf{w}]_k}{2} \right) \mathbb{1}_+\Big([\mathbf{w}]_k\Big), \end{aligned} \tag{4.29}$$

where $\mathbf{y} \in \mathbb{R}^M$ and $\mathbf{A} \coloneqq \mathbf{H}\mathbf{D}^{-1}$.

Now, the conditional distribution $p_{\mathsf{\mathbf{U}}|\mathsf{\mathbf{W}},\mathsf{\mathbf{Y}}}$ turns out to be

$$p_{\mathsf{\mathbf{U}}|\mathsf{\mathbf{W}},\mathsf{\mathbf{Y}}}(\mathbf{u}|\mathbf{w}, \mathbf{y}) \propto \ \exp\left(-\frac{1}{2} \left(\frac{1}{\sigma_{\mathrm{n}}^2} \|\mathbf{y} - \mathbf{A}\mathbf{u}\|_2^2 + \mathbf{u}^T \mathbf{C}_{\mathrm{T}}(\mathbf{w}) \mathbf{u} \right) \right), \tag{4.30}$$

where $\mathbf{C}_{\mathrm{T}}(\mathbf{w})$ is a diagonal matrix with entries $\Big([\mathbf{w}]_k\Big)_{k=1}^{K}$. Similar to what we observed in the Laplace case, $p_{\mathsf{\mathbf{U}}|\mathsf{\mathbf{W}},\mathsf{\mathbf{Y}}}(\cdot|\mathbf{w}, \mathbf{y})$ is a multivariate Gaussian density with mean $\overline{\mathbf{u}} = \sigma_{\mathrm{n}}^{-2}\Big(\sigma_{\mathrm{n}}^{-2}\mathbf{A}^T\mathbf{A} + \mathbf{C}_{\mathrm{T}}(\mathbf{w})\Big)^{-1}\mathbf{A}^T\mathbf{y}$ and covariance matrix $\overline{\mathbf{R}} = \Big(\sigma_{\mathrm{n}}^{-2}\mathbf{A}^T\mathbf{A} + \mathbf{C}_{\mathrm{T}}(\mathbf{w})\Big)^{-1}$.

The distribution $p_{\mathsf{\mathbf{W}}|\mathsf{\mathbf{U}},\mathsf{\mathbf{Y}}}$ is again separable and takes the form

$$p_{\mathsf{\mathbf{W}}|\mathsf{\mathbf{U}},\mathsf{\mathbf{Y}}}(\mathbf{w}|\mathbf{u}, \mathbf{y}) \propto \prod_{k=1}^{K} p_{\mathsf{W}|\mathsf{U},\mathsf{\mathbf{Y}}}\Big([\mathbf{w}]_k | [\mathbf{u}]_k, \mathbf{y}\Big), \tag{4.31}$$

[4]The pdf of the gamma distribution is

$$p_{\mathrm{gam}}(x) = \frac{1}{\lambda_2^{\lambda_1}\Gamma(\lambda_1)} x^{\lambda_1 - 1} \mathrm{e}^{-x/\lambda_2} \mathbb{1}_+(x),$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$ are the shape and scale parameters, respectively.

where

$$p_{\mathsf{W}|\mathsf{U},\mathbf{Y}}\Big(w|u,\mathbf{y}\Big) \propto \exp\left(-\frac{(1+u)^2 w}{2}\right) \times w^{\frac{\alpha-1}{2}} \mathbb{1}_+(w). \tag{4.32}$$

is a gamma distribution with $\lambda_1 = \frac{\alpha+1}{2}$ and $\lambda_2 = \frac{2}{(1+u)^2}$, which can easily be sampled from.

4.4.4 Bernoulli-Laplace Increments

In [150], Gibbs sampling schemes have been designed for a deconvolution problem where the underlying signal is an i.i.d. spike train that follows the Bernoulli-Gaussian distribution. Unfortunately, the Bernoulli-Gaussian distribution is not infinitely divisible and so is not compatible with our framework of Lévy processes. While there exists some work [32] on Bernoulli-Laplace priors, according to the analysis presented in [150], their proposed sampler would have a tendency to get stuck in certain configurations. Thus, we build upon the method in [150] and develop a novel Gibbs sampler for Lévy processes with Bernoulli-Laplace increments.

The Bernoulli-Laplace pdf is

$$p_{\mathsf{U}}(u) = \lambda\delta(u) + (1-\lambda)\frac{b}{2}\exp\Big(-b|u|\Big), \tag{4.33}$$

where $\lambda \in (0,1)$ denotes the mass probability at the origin and b is a scale parameter. We can represent this same density as

$$p_{\mathsf{U}}(u) = \int_{\mathbb{R}} \left(\sum_{v=0}^{1} p_{\mathsf{U}|\mathsf{V},\mathsf{W}}(u|v,w)p_{\mathsf{V}}(v)\right)p_{\mathsf{W}}(w)\mathrm{d}w, \tag{4.34}$$

where

$$p_{\mathsf{V}}(v) = (\lambda)^{1-v}(1-\lambda)^v \quad \text{for } v \in \{0,1\} \tag{4.35}$$

is a Bernoulli distribution,

$$p_{\mathsf{W}}(w) = \frac{b^2}{2}\exp\left(-\frac{b^2 w}{2}\right)\mathbb{1}_+(w) \tag{4.36}$$

is an exponential pdf, and $p_{\mathsf{U}|\mathsf{V},\mathsf{W}}$ is defined such that

$$p_{\mathsf{U}|\mathsf{V},\mathsf{W}}(u|v=0,w) = \delta(u) \tag{4.37}$$

$$p_{\mathsf{U}|\mathsf{V},\mathsf{W}}(u|v=1,w) = \frac{1}{\sqrt{2\pi w}}\exp\left(-\frac{u^2}{2w}\right). \tag{4.38}$$

Based on this representation, we introduce two independent auxiliary random vectors $\mathbf{V}$

and W that take values in $\mathbb{R}^K$. Their elements are i.i.d. and follow the distributions p_{V} and p_{W}, as defined in (4.35) and (4.36), respectively. Further, these vectors satisfy

$$p_{\mathsf{U}|\mathsf{V},\mathsf{W}}(\mathbf{u}|\mathbf{v},\mathbf{w}) = \prod_{k=1}^{K} p_{\mathsf{U}|\mathsf{V},\mathsf{W}}\Big([\mathbf{u}]_k|[\mathbf{v}]_k,[\mathbf{w}]_k\Big), \tag{4.39}$$

where $\mathbf{u},\mathbf{v},\mathbf{w} \in \mathbb{R}^K$ and $p_{\mathsf{U}|\mathsf{V},\mathsf{W}}$ is defined in (4.37) and (4.38).

Here, the full joint distribution $p_{\mathsf{Y},\mathsf{U},\mathsf{V},\mathsf{W}}$ is given by

$$\begin{aligned} p_{\mathsf{Y},\mathsf{U},\mathsf{V},\mathsf{W}}(\mathbf{y},\mathbf{u},\mathbf{v},\mathbf{w}) &= p_{\mathsf{Y}|\mathsf{U},\mathsf{V},\mathsf{W}}(\mathbf{y}|\mathbf{u},\mathbf{v},\mathbf{w})p_{\mathsf{U},\mathsf{V},\mathsf{W}}(\mathbf{u},\mathbf{v},\mathbf{w}) \\ &= p_{\mathsf{Y}|\mathsf{U}}(\mathbf{y}|\mathbf{u})p_{\mathsf{U}|\mathsf{V},\mathsf{W}}(\mathbf{u}|\mathbf{v},\mathbf{w})p_{\mathsf{V}}(\mathbf{v})p_{\mathsf{W}}(\mathbf{w}), \end{aligned} \tag{4.40}$$

where $\mathbf{y} \in \mathbb{R}^M$. As a result, the distribution $p_{\mathsf{U},\mathsf{V},\mathsf{W}|\mathsf{Y}}$ takes the form

$$\begin{aligned} p_{\mathsf{U},\mathsf{V},\mathsf{W}|\mathsf{Y}}(\mathbf{u},\mathbf{v},\mathbf{w}|\mathbf{y}) \;\propto\; & \exp\left(-\frac{1}{2\sigma_{\mathrm{n}}^2}\|\mathbf{y}-\mathbf{A}\mathbf{u}\|_2^2\right) \times \prod_{k=1}^{K} p_{\mathsf{U}|\mathsf{V},\mathsf{W}}\Big([\mathbf{u}]_k|[\mathbf{v}]_k,[\mathbf{w}]_k\Big) \\ & \times \prod_{k=1}^{K} \lambda^{1-[\mathbf{v}]_k}(1-\lambda)^{[\mathbf{v}]_k} \times \prod_{k=1}^{K} \frac{b^2}{2}\exp\left(-\frac{b^2[\mathbf{w}]_k}{2}\right)\mathbb{1}_+([\mathbf{w}]_k), \end{aligned} \tag{4.41}$$

where $\mathbf{A} = \mathbf{H}\mathbf{D}^{-1}$.

Let us now introduce some notations. For any binary vector $\mathbf{q} \in \mathbb{R}^K$, let $\mathcal{I}_{\mathbf{q},0}$ and $\mathcal{I}_{\mathbf{q},1}$ denote sets of indices such that $[\mathbf{q}]_k = 0$ for $k \in \mathcal{I}_{\mathbf{q},0}$ and $[\mathbf{q}]_k = 1$ for $k \in \mathcal{I}_{\mathbf{q},1}$. Further, let $\mathbf{A}(\mathbf{q})$ be the matrix constructed by taking the columns of $\mathbf{A}$ corresponding to the indices in $\mathcal{I}_{\mathbf{q},1}$. We then define the matrix $\mathbf{B}(\mathbf{q},\mathbf{r}) = \sigma_{\mathrm{n}}^2\mathbf{I}+\mathbf{A}(\mathbf{q})\mathbf{C}_{\mathrm{BL}}(\mathbf{q},\mathbf{r})\mathbf{A}(\mathbf{q})^T$, where $\mathbf{r} \in \mathbb{R}^K$ is a vector with positive entries and $\mathbf{C}_{\mathrm{BL}}(\mathbf{q},\mathbf{r})$ is a diagonal matrix with entries $([\mathbf{r}]_k)_{k\in\mathcal{I}_{\mathbf{q},1}}$. Here, we also introduce the vector $\mathbf{q}_{(-k)} \in \mathbb{R}^{K-1}$ that contains all the entries of $\mathbf{q}$ except the kth one, so that $\mathbf{q}_{(-k)} = ([\mathbf{q}]_1,\dots,[\mathbf{q}]_{k-1},[\mathbf{q}]_{k+1},\dots,[\mathbf{q}]_K)^T$. Similarly, for a random vector Q that takes values in $\mathbb{R}^K$, we have $\mathsf{Q}_{(-k)} = ([\mathsf{Q}]_1,\dots,[\mathsf{Q}]_{k-1},[\mathsf{Q}]_{k+1},\dots,[\mathsf{Q}]_K)^T$. Lastly, for $q \in \{0,1\}$, we define the vector $\mathbf{q}^q_{(-k)} \in \mathbb{R}^K$ such that $\mathbf{q}^q_{(-k)} = ([\mathbf{q}]_1,\dots,[\mathbf{q}]_{k-1},q$ $,[\mathbf{q}]_{k+1},\dots,[\mathbf{q}]_K)^T$.

First, we look at the conditional distribution $p_{\mathsf{U}|\mathsf{V},\mathsf{W},\mathsf{Y}}$. From (4.37) and (4.41), we deduce that any sample from $p_{\mathsf{U}|\mathsf{V},\mathsf{W},\mathsf{Y}}(\cdot|\mathbf{v},\mathbf{w},\mathbf{y})$ takes the value of zero at the indices in $\mathcal{I}_{\mathbf{v},0}$. If we define $(\mathsf{U}_1|\mathsf{V}=\mathbf{v},\mathsf{W}=\mathbf{w},\mathsf{Y}=\mathbf{y}) = ([\mathsf{U}|\mathsf{V}=\mathbf{v},\mathsf{W}=\mathbf{w},\mathsf{Y}=\mathbf{y}]_k)_{k\in\mathcal{I}_{\mathbf{v},1}}$, then we get

$$p_{\mathsf{U}_1|\mathsf{V},\mathsf{W},\mathsf{Y}}(\mathbf{u}_1|\mathbf{v},\mathbf{w},\mathbf{y}) \propto \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma_{\mathrm{n}}^2}\|\mathbf{y}-\mathbf{A}(\mathbf{v})\mathbf{u}_1\|_2^2 + \mathbf{u}_1^T\mathbf{C}_{\mathrm{BL}}(\mathbf{v},\mathbf{w})\mathbf{u}_1\right)\right), \tag{4.42}$$

where $\mathbf{u}_1 \in \mathbb{R}^{|\mathcal{I}_{\mathbf{v},1}|}$. Thus, $p_{\mathsf{U}_1|\mathsf{V},\mathsf{W},\mathsf{Y}}$ is a multivariate Gaussian density with mean $\overline{\mathbf{u}_1} =$

$\sigma_{\mathrm{n}}^{-2}\Big(\sigma_{\mathrm{n}}^{-2}\mathbf{A}(\mathbf{v})^T\mathbf{A}(\mathbf{v})+\mathbf{C}_{\mathrm{BL}}(\mathbf{v},\mathbf{w})\Big)^{-1}\mathbf{A}(\mathbf{v})^T\mathbf{y}$ and covariance matrix $\overline{\mathbf{R}} = \Big(\sigma_{\mathrm{n}}^{-2}\mathbf{A}(\mathbf{v})^T\mathbf{A}(\mathbf{v})+\mathbf{C}_{\mathrm{BL}}(\mathbf{v},\mathbf{w})\Big)^{-1}$.

The conditional distribution $p_{\mathsf{W}|\mathsf{U},\mathsf{V},\mathsf{Y}}$ takes the form

$$p_{\mathsf{W}|\mathsf{U},\mathsf{V},\mathsf{Y}}(\mathbf{w}|\mathbf{u},\mathbf{v},\mathbf{y}) \propto \prod_{k=1}^{K} p_{\mathsf{W}|\mathsf{U},\mathsf{V},\mathsf{Y}}\Big([\mathbf{w}]_k|[\mathbf{u}]_k,[\mathbf{v}]_k,\mathbf{y}\Big), \tag{4.43}$$

where $p_{\mathsf{W}|\mathsf{U},\mathsf{V},\mathsf{Y}}$ is given by

$$p_{\mathsf{W}|\mathsf{U},\mathsf{V},\mathsf{Y}}(w|u,v=0,\mathbf{y}) \propto \frac{b^2}{2}\exp\left(-\frac{b^2 w}{2}\right)\mathbb{1}_{+}(w) \tag{4.44}$$

$$p_{\mathsf{W}|\mathsf{U},\mathsf{V},\mathsf{Y}}(w|u,v=1,\mathbf{y}) \propto \exp\left(-\frac{1}{2}\left(\frac{u^2}{w}+b^2 w\right)\right) \times w^{-\frac{1}{2}}\mathbb{1}_{+}(w). \tag{4.45}$$

The densities in (4.44) and (4.45) correspond to the exponential distribution with $\lambda = 2/b^2$ and the generalized inverse Gaussian distribution with $\lambda_1 = b^2$, $\lambda_2 = u^2$, and $a = 0.5$.

Next, inspired by the work in [150], we consider sampling from the marginalized conditional distribution of $[\mathsf{V}]_k|\mathsf{V}_{(-k)} = \mathbf{v}_{-(k)}, \mathsf{W} = \mathbf{w}, \mathsf{Y} = \mathbf{y}$ in a sequential manner as this can allow for a more efficient exploration of configurations of $\mathsf{V}|\mathsf{Y} = \mathbf{y}$. More specifically, at each iteration q, we will draw $[\mathbf{v}^{(q)}]_k$ from the distribution $p_{[\mathsf{V}]_k|\mathsf{V}_{(-k)},\mathsf{W},\mathsf{Y}}\Big(v|\mathbf{v}^{(q)}_{(-k)},\mathbf{w}^{(q)},\mathbf{y}\Big)$, where $\mathbf{v}^{(q)}_{(-k)} = \Big([\mathbf{v}^{(q)}]_1,\ldots,[\mathbf{v}^{(q)}]_{k-1},[\mathbf{v}^{(q-1)}]_{k+1},\ldots,[\mathbf{v}^{(q-1)}]_K\Big)$ and $k \in \{1,\ldots,K\}$.

The marginalized posterior distribution $p_{\mathsf{V},\mathsf{W}|\mathsf{Y}}$ is given by

$$p_{\mathsf{V},\mathsf{W}|\mathsf{Y}}(\mathbf{v},\mathbf{w}|\mathbf{y}) \propto p_{\mathsf{Y}|\mathsf{V},\mathsf{W}}(\mathbf{y}|\mathbf{v},\mathbf{w})p_{\mathsf{V}}(\mathbf{v})p_{\mathsf{W}}(\mathbf{w}), \tag{4.46}$$

where

$$p_{\mathsf{Y}|\mathsf{V},\mathsf{W}}(\mathbf{y}|\mathbf{v},\mathbf{w}) = \int_{\mathbb{R}^K} p_{\mathsf{Y}|\mathsf{U},\mathsf{V},\mathsf{W}}(\mathbf{y}|\mathbf{u},\mathbf{v},\mathbf{w})p_{\mathsf{U}|\mathsf{V},\mathsf{W}}(\mathbf{u}|\mathbf{v},\mathbf{w})\mathrm{d}\mathbf{u}. \tag{4.47}$$

It can be shown that (4.46) and (4.47) lead to

$$\begin{aligned} p_{\mathsf{V},\mathsf{W}|\mathsf{Y}}(\mathbf{v},\mathbf{w}|\mathbf{y}) &\propto |\mathbf{B}(\mathbf{v},\mathbf{w})|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\mathbf{y}^T\mathbf{B}(\mathbf{v},\mathbf{w})^{-1}\mathbf{y}\right) \\ &\times \prod_{k=1}^{K}\lambda^{1-[\mathbf{v}]_k}(1-\lambda)^{[\mathbf{v}]_k} \times \prod_{k=1}^{K}\frac{b^2}{2}\exp\left(-\frac{b^2[\mathbf{w}]_k}{2}\right)\mathbb{1}_{+}([\mathbf{w}]_k). \end{aligned} \tag{4.48}$$

From (4.48), we see that $p_{[\mathbf{V}]_k|\mathbf{V}_{(-k)},\mathbf{W},\mathbf{Y}}$ is a Bernoulli distribution with

$$p_{[\mathbf{V}]_k|\mathbf{V}_{(-k)},\mathbf{W},\mathbf{Y}}(v|\mathbf{v}_{(-k)},\mathbf{w},\mathbf{y}) = \left(1+\exp\left(-\frac{1}{2}\left(h\left(1-v;\mathbf{v}_{(-k)},\mathbf{w},\mathbf{y}\right) - h\left(v;\mathbf{v}_{(-k)},\mathbf{w},\mathbf{y}\right)\right)\right)\right)^{-1}, \tag{4.49}$$

where

$$h\left(v;\mathbf{v}_{(-k)},\mathbf{w},\mathbf{y}\right) = \mathbf{y}^T\mathbf{B}\left(\mathbf{v}^v_{(-k)},\mathbf{w}\right)^{-1}\mathbf{y} + \log\left(\left|\mathbf{B}\left(\mathbf{v}^v_{(-k)},\mathbf{w}\right)\right|\right) + 2v\log\left(\frac{\lambda}{1-\lambda}\right). \tag{4.50}$$

To conclude, using the conditional distributions derived above, we construct a Gibbs sampler, where in each iteration q, we generate $\mathbf{w}^{\dagger(q)}$ according to $p_{\mathbf{W}|\mathbf{U},\mathbf{V},\mathbf{Y}}\left(\cdot|\mathbf{u}^{\dagger(q-1)},\mathbf{v}^{\dagger(q-1)},\mathbf{y}^\dagger\right)$, $[\mathbf{v}^{\dagger(q)}]_k$ according to $p_{[\mathbf{V}]_k|\mathbf{V}_{(-k)},\mathbf{W},\mathbf{Y}}\left(\cdot|\mathbf{v}^{\dagger(q)}_{(-k)},\mathbf{w}^{\dagger(q)},\mathbf{y}^\dagger\right)$ for all $k\in\{1,\ldots,K\}$ and $\mathbf{u}^{\dagger(q)}$ according to $p_{\mathbf{U}|\mathbf{V},\mathbf{W},\mathbf{Y}}\left(\cdot|\mathbf{v}^{\dagger(q)},\mathbf{w}^{\dagger(q)},\mathbf{y}^\dagger\right)$. This particular order of updates is important as it yields a partially collapsed Gibbs sampler [151] where the stationary distribution is still $p_{\mathbf{U},\mathbf{V},\mathbf{W}|\mathbf{Y}}(\cdot,\cdot,\cdot|\mathbf{y}^\dagger)$.

4.5 Numerical Experiments

In our experiments, we benchmark the performance of some popular signal reconstruction schemes, including a CNN-based method, on deconvolution and Fourier sampling problems with Lévy processes associated with the Bernoulli-Laplace and Student's t distributions.

4.5.1 Signal Models

We consider a signal vector $\mathbf{s}^\dagger\in\mathbb{R}^{100}$ that contains samples of a Lévy process whose increments follow the Bernoulli-Laplace or Student's t distribution.

Bernoulli-Laplace increments

The Bernoulli-Laplace pdf (4.33) is characterized by the parameters λ and b, where λ determines the mass probability at the origin and b represents the scale of the Laplace component. We perform experiments for models corresponding to $\lambda\in\{0.6,0.7,0.8,0.9\}$. The scale parameter is set to $b=1$ for each case.

Student't t increments

The Student's t pdf (4.24) is parameterized by α, which controls the tails of the distribution. We conduct experiments for $\alpha \in \{1, 3, 5, 39\}$.

4.5.2 Measurement Models

We consider both deconvolution and Fourier sampling problems for each of the above-described signal models.

Deconvolution

As shown in Section 4.2.3, the system matrix $\mathbf{H}$ for deconvolution is a discrete convolution matrix. Accordingly, we construct $\mathbf{H} : \mathbb{R}^{100} \to \mathbb{R}^{88}$ such that

$$\mathbf{H} = \begin{bmatrix} [\mathbf{h}]_{13} \cdots [\mathbf{h}]_1 & 0 & \cdots & 0 \\ 0 \quad \ddots & \ddots & \ddots & \vdots \\ \vdots \quad \ddots \ \ddots & & \ddots & 0 \\ 0 \quad \cdots \quad 0 & [\mathbf{h}]_{13} & \cdots & [\mathbf{h}]_1 \end{bmatrix}, \tag{4.51}$$

where $\mathbf{h} \in \mathbb{R}^{13}$ consists of the central samples of a truncated Gaussian PSF with variance $\sigma_0^2 = 4$.

Fourier Sampling

For Fourier sampling in 1D, which is reminiscent of MRI, the forward model $\mathbf{H}$ resembles a discrete Fourier matrix (see Section 4.2.4). Thus, in order to construct $\mathbf{H}$, we first sample $M' = 16$ rows of the DFT matrix. The first row of the DFT matrix (DC component) is always kept, while the remaining ones are selected in a quasi-random fashion with a denser sampling at low frequencies. We then create the real system matrix $\mathbf{H} : \mathbb{R}^{100} \to \mathbb{R}^{M}$, where $M = 2M' - 1$, by separating the real and imaginary parts.

In both measurement models, the AWGN variance σ_{n}^2 is chosen such that the (average) measurement SNR is around 30 dB.

4.5.3 Reconstruction Algorithms

For each combination of the signal and measurement models, we compare the performance of some classical MPL estimators, a CNN-based scheme and the MMSE estimator. We generate validation and test datasets, each consisting of 1,000 pairs of ground-truth signals and their noisy measurements. Further, in order to train the CNNs, we also synthesize a repository $\mathcal{T}$ containing a large number of training examples.

Classical methods

We consider the MPL estimators

$$\mathbf{s}^*_{\ell_2}(\mathbf{y}^\dagger) = \arg\min_{\mathbf{s}\in\mathbb{R}^K} \left(\|\mathbf{y}^\dagger - \mathbf{H}\mathbf{s}\|_2^2 + \tau\|\mathbf{D}\mathbf{s}\|_2^2 \right), \tag{4.52}$$

$$\mathbf{s}^*_{\ell_1}(\mathbf{y}^\dagger) = \arg\min_{\mathbf{s}\in\mathbb{R}^K} \left(\|\mathbf{y}^\dagger - \mathbf{H}\mathbf{s}\|_2^2 + \tau\|\mathbf{D}\mathbf{s}\|_1 \right), \tag{4.53}$$

and

$$\mathbf{s}^*_{\log}(\mathbf{y}^\dagger) = \arg\min_{\mathbf{s}\in\mathbb{R}^K} \left(\|\mathbf{y}^\dagger - \mathbf{H}\mathbf{s}\|_2^2 + \tau\sum_{k=1}^{K} \log\left(1 + \left([\mathbf{D}\mathbf{s}]_k\right)^2\right)\right), \tag{4.54}$$

where $\tau \in \mathbb{R}_+$. For each of these methods, the same regularization parameter τ is used for the entire test dataset. This particular value of τ is the one that yields the lowest MSE for the validation dataset.

These estimators are implemented in MATLAB using GlobalBioIm [123]—a library for solving inverse problems. Specifically, the ℓ_2 estimator is expressed in closed-form as

$$\mathbf{s}^*_{\ell_2}(\mathbf{y}^\dagger) = \left(\mathbf{H}^T\mathbf{H} + \tau\mathbf{D}^T\mathbf{D}\right)^{-1}\mathbf{H}^T\mathbf{y}^\dagger. \tag{4.55}$$

The ℓ_1 and log estimators are computed iteratively using ADMM. Since the cost functional in (4.54) is non-convex, we initialize ADMM for computing the log estimate with the ℓ_1 estimate so that it can reach a better local minimum.

CNN-based method

The concept here is to train a CNN as a regressor that maps an initial low-quality reconstruction $\mathbf{s}^*_{\text{init}}(\mathbf{y}^\dagger)$ to a high-quality one [41–43, 152, 153]. The architecture of the CNN used in our experiments is based on the well-known denoising network DnCNN [154] and is described in Figure 4.2 and Table 4.1.

First, we build a training dataset of cardinality M_T by taking the first M_T examples $\{\mathbf{s}^\dagger_m, \mathbf{y}^\dagger_m\}_{m=1}^{M_T}$ from the repository $\mathcal{T}$. We then train the model by minimizing the MSE loss function

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M_T}\sum_{m=1}^{M_T} \left\|\mathbf{s}^\dagger_m - \mathrm{CNN}_{\boldsymbol{\theta}}\left(\mathbf{s}^*_{\text{init}}(\mathbf{y}^\dagger_m)\right)\right\|_2^2, \tag{4.56}$$

where $\boldsymbol{\theta}$ represents the learnable parameters of the network, with the help of the ADAM optimizer [155]. The CNN is trained for 1,000 epochs with a batch size of 256 and a weight decay of γ. The initial learning rate is set as 10^{-2}. For some duration of the training (first

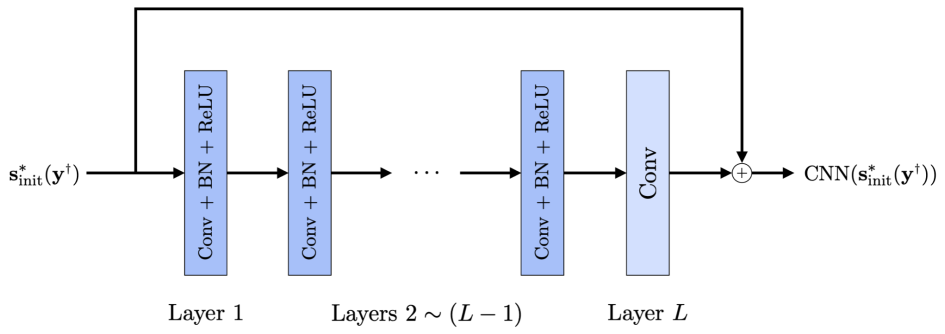

$\mathbf{s}^*_{\text{init}}(\mathbf{y}^\dagger)$
Conv + BN + ReLU
Conv + BN + ReLU
···
Conv + BN + ReLU
Conv
$\text{CNN}(\mathbf{s}^*_{\text{init}}(\mathbf{y}^\dagger))$
Layer 1
Layers $2 \sim (L-1)$
Layer L

Figure 4.2: Architecture of the CNN, where BN denotes the operation of batch normalization.

Table 4.1: Convolution Layers.

Layer	Filter size	Input channels	Output channels
1	$(F \times 1)$	1	C
$2 \sim (L-1)$	$(F \times 1)$	C	C
L	$(F \times 1)$	C	1

600 epochs for deconvolution and first 750 epochs for Fourier sampling), it is decreased by a factor of 0.5 every 50 epochs. We choose the initial low-quality reconstruction to be $\mathbf{s}^*_{\text{init}}(\mathbf{y}^\dagger) = \mathbf{H}^T\mathbf{y}^\dagger$ for the deconvolution problems. In the case of Fourier sampling, $\mathbf{s}^*_{\text{init}}(\mathbf{y}^\dagger)$ is the zero-filled reconstruction. All the CNN-based reconstruction schemes are implemented in PyTorch.

Goldstandard (MMSE estimator)

Our MMSE estimators are implemented in MATLAB, according to the methods detailed in Section 4.4. There, we set the number of samples as $Q = 8{,}000$ and the burn-in period as $B = 3{,}000$ for signals with Bernoulli-Laplace increments. We use $Q = 15{,}000$ and $B = 5{,}000$ for signals associated with the Student's t distribution.

4.5.4 Results

We present our results for all the test datasets in Figures 4.3, 4.4, 4.6 and 4.5. For the sake of clarity, instead of the MSE, we display the "MSE optimality gap" which is the difference between the MSE obtained by a specific method and the MSE attained by the MMSE estimator. In these figures, the CNNs are labelled using the tuple (F, C, L, M_T, γ), where F is the filter size, C is the number of channels, L is the number of layers, M_T

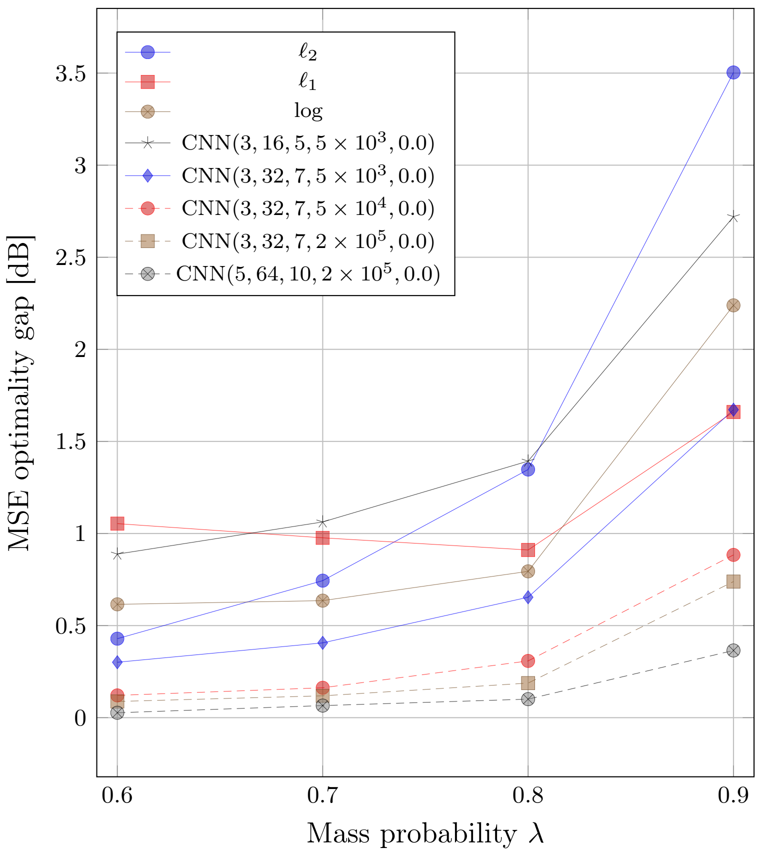


Figure 4.3: Deconvolution of Lévy processes with Bernoulli-Laplace increments.

is the cardinality of the training dataset and γ is the weight decay. For the interested reader, we also provide information about the computation times required by all the methods in the supplementary material.

Lévy processes with Bernoulli-Laplace increments

Here, we summarize our observations for both the deconvolution and Fourier sampling experiments (Figures 4.3 and 4.4).

The sparsity-promoting ℓ_1 estimator, which corresponds to the popular TV regularization, is known to be well-suited to piecewise-constant Lévy processes with Bernoulli-Laplace increments. As the value of λ increases, these signals become sparser and exhibit fewer jumps. Consequently, we observe that the ℓ_1 estimator performs better than the ℓ_2 estimator. The log estimator also promotes sparse solutions [29] and we see that it performs well for these piecewise-constant signals. However, despite the good fit, there is still some gap between the MSE attained by the ℓ_1 and log, and MMSE estimators.

The performance of the CNN-based method improves as we increase the capacity of the

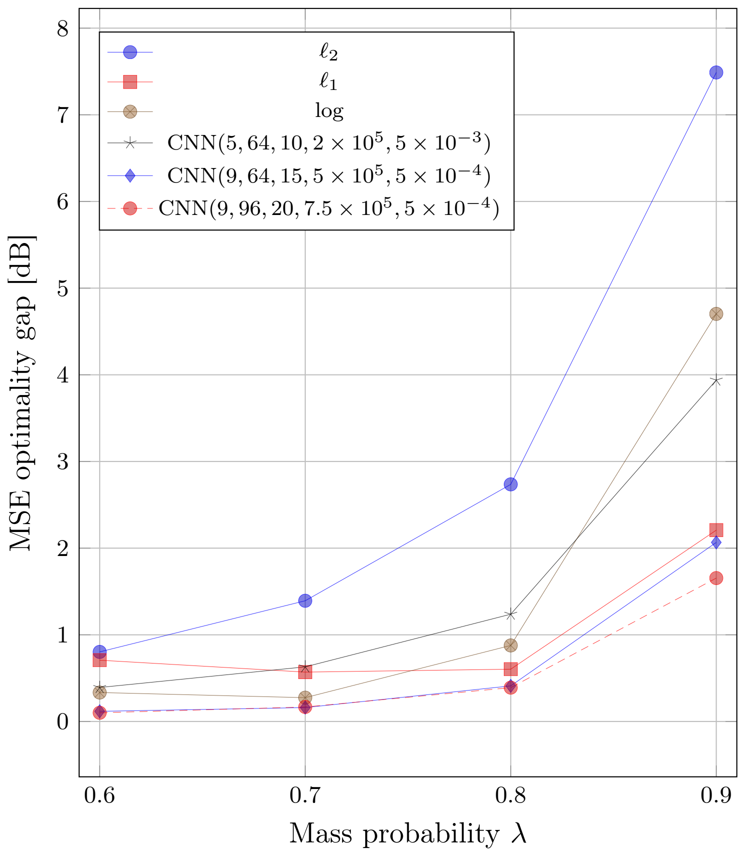


Figure 4.4: Fourier sampling of Lévy processes with Bernoulli-Laplace increments.

CNN and the amount of training data. In fact, with sufficient capacity and training data, they outperform the ℓ_1 and log estimators. Remarkably, some of the CNNs achieve a near-optimal MSE.

Lévy processes with Student's t increments

The parameter α allows us to consider a wide range of signals. As $\alpha \to \infty$, we approach the Gaussian regime. The other extreme is $\alpha = 1$, which corresponds to the super heavy-tailed (sparse) Cauchy distribution. This scenario can be particularly challenging for the correct setting of algorithm parameters. Due to the heavy tails of the Cauchy distribution, the validation and test datasets may contain signals with a vastly different range of values. Consequently, for a given model-based method, the regularization parameter τ that is chosen to yield the lowest MSE for the validation dataset may differ significantly from the value τ^* that achieves the lowest MSE on the test dataset. Thus, in Figures 4.6 and 4.5, we also include the performance of model-based methods when their regularization parameter is tuned for optimal MSE performance on the test dataset directly. These "boosted" model-based methods are labelled as ℓ_2^*, ℓ_1^* and log*.

In Figures 4.6 and 4.5, we can see that the ℓ_2 estimator is optimal for a large value of α. As the value of α decreases, the performance of the ℓ_2 estimator deteriorates and becomes worse than that of the ℓ_1 estimator. For all the cases, the log estimator attains reasonable MSE values. Note that for the deconvolution experiment involving Cauchy signals, there is a significant gap between the MSE values obtained by the ℓ_2 and ℓ_1 and ℓ_2^* and ℓ_1^* estimators, respectively. Interestingly, the log estimator is less affected by this issue.

Finally, for both deconvolution and Fourier sampling problems, CNNs with sufficient capacity and training data perform well up to $\alpha = 3$, after which there seems to be a steep transition and their performance drops sharply. In fact, for Cauchy signals, we observe that the training process for these CNNs is quite unstable—the training loss marginally decreases and seems to converge, and the networks do not generalize to the validation (or test) datasets. We believe that this last example poses an open challenge for designing robust neural-network-based schemes that can handle signals following (super) heavy-tailed distributions.

4.6 Summary

We have introduced a controlled environment, based on sparse stochastic processes (SSPs), for the objective benchmarking of reconstruction algorithms, including NN-based methods that require lots of training data, in the context of linear inverse problems. We have developed efficient posterior sampling schemes to compute the minimum-mean-square-error estimators for specific classes of SSPs. These yield the upper limit on reconstruction performance and allow us to provide a measure of statistical optimality. We have highlighted the abilities of our framework by benchmarking some popular classical MPL estimators and convolutional neural-network (CNN) architectures for deconvolution and Fourier-sampling problems. In particular, we have observed that, while CNNs outperform the sparsity-based MPL estimators and achieve a near-optimal performance in terms of mean-square error for a wide range of conditions, they can sometimes fail too, especially for signals with heavy-tailed innovations.

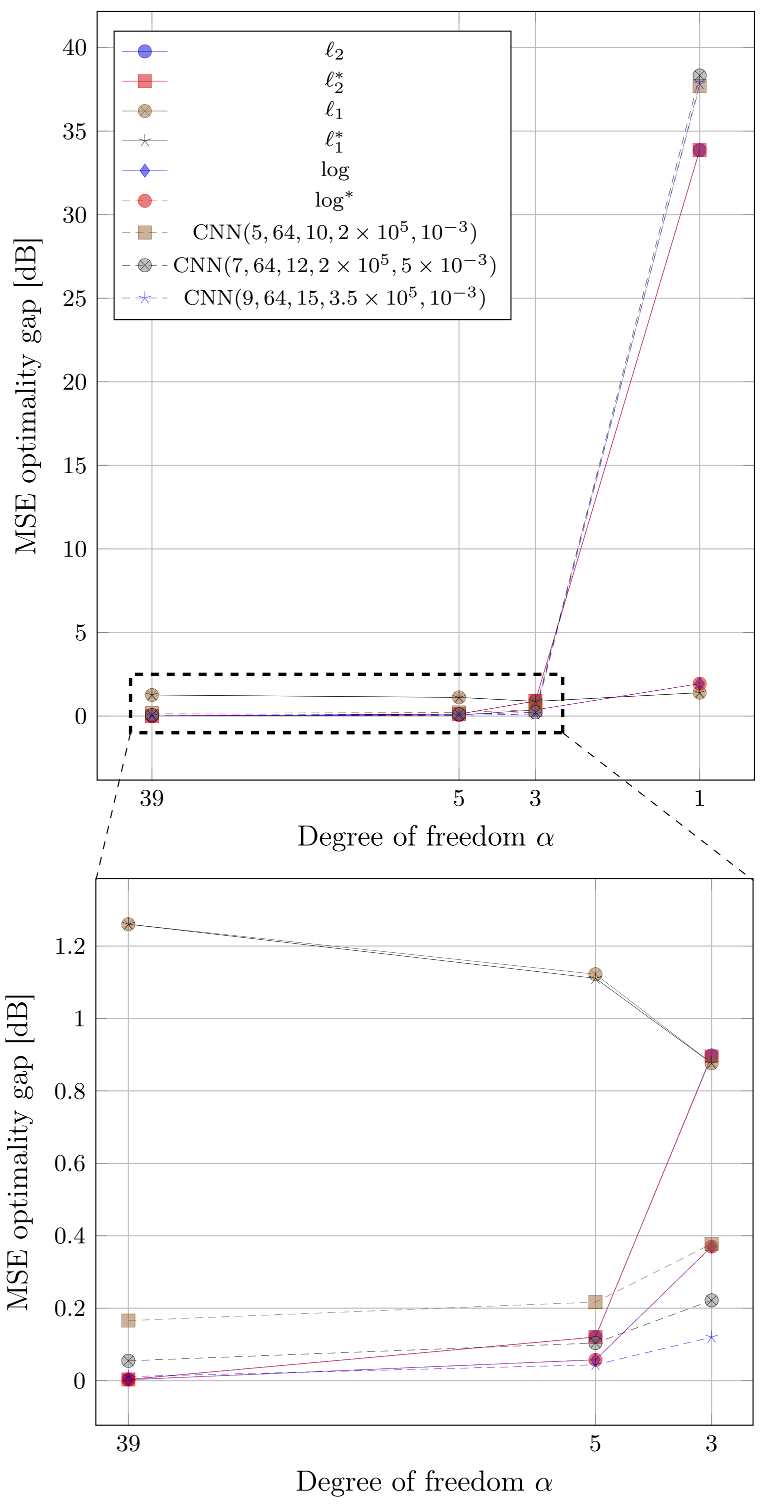


Figure 4.5: Fourier sampling of Lévy processes with Student's t increments. The figure at the bottom is a zoomed-in version of the dotted rectangular box shown in the figure at the top.

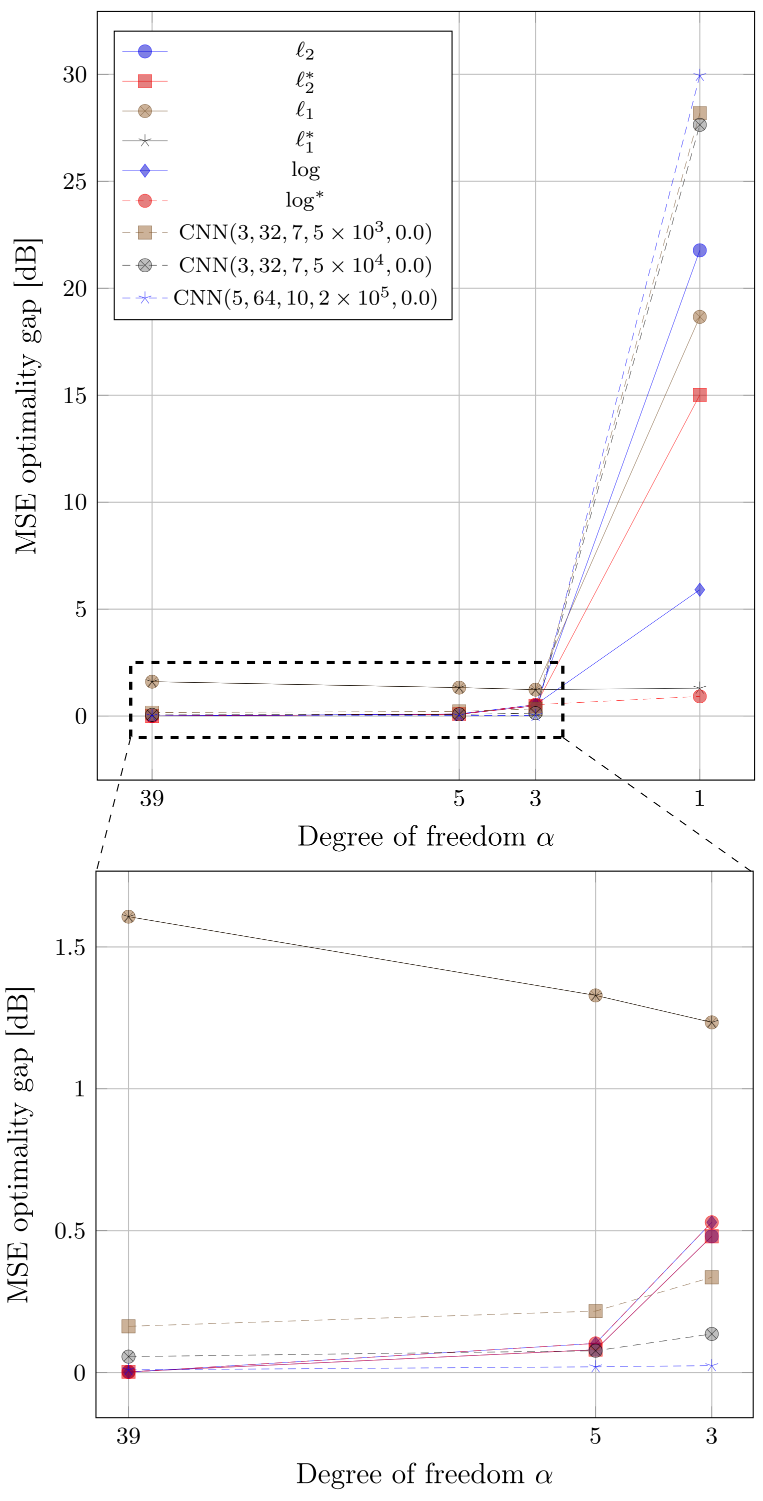


Figure 4.6: Deconvolution of Lévy processes with Student's t increments. The figure at the bottom is a zoomed-in version of the dotted rectangular box shown in the figure at the top.

Part II The Neural Network Revolution

5 Convergent Iterative Image-Reconstruction Methods

[1]In this chapter, we focus on the development of universal neural-network-based approaches within the penalized-likelihood-based estimation paradigm for solving linear inverse problems in imaging. Here, we first present an efficient module for learning continuous piecewise-linear activation functions in neural networks (Section 5.1). We then deploy this module to train 1-Lipschitz denoising convolutional neural networks (Section 5.2) and learnable convex regularizers (Section 5.3), both of which can be used to design provably convergent iterative image-reconstruction methods.

5.1 Learning Activation Functions in Neural Networks

[2]In this section, we present an efficient computational solution for learning component-wise activation functions in neural networks.

5.1.1 Introduction

During the past decade, deep neural networks (DNNs) have evolved into a major player for machine learning. They have been found to outperform the traditional techniques of statistical learning [160] (*e.g.,* kernel methods, support-vector machines, random forests) in many real-world applications that include image classification [161], speech recognition [162], image segmentation [163], and medical imaging [41].

The basic principle behind DNNs is to construct powerful learning architectures via the composition of simple basic modules; that is, linear (or affine) transformations and pointwise nonlinearities [40]. The qualifier "deep" refers to the depth (or number of layers) of such a composition which is typically much larger than one. Formally, a typical feedfoward DNN is a map $\mathbf{f}_{\boldsymbol{\theta}} : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ that admits a factorized representation of the

[1]This chapter is based on our works [156–159].
[2]This section is based on our work [156].

form

$$\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) : \boldsymbol{A}_L \circ \cdots \circ \boldsymbol{\sigma}_\ell \circ \boldsymbol{A}_\ell \circ \cdots \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{A}_1(\mathbf{x}), \tag{5.1}$$

where L is the depth of the neural net and $\boldsymbol{\theta}$ is a list of parameters that collects all adjustable quantities. Specifically, a given layer ℓ of the network is characterized by

1. a linear transformation $\mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell} : \mathbf{x} \mapsto \boldsymbol{A}_\ell(\mathbf{x}) = \mathbf{W}_\ell \mathbf{x}$, where $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ is a matrix of weights, and
2. the pointwise responses of its neurons

$$\boldsymbol{\sigma}_\ell(\mathbf{x}) = \Big(\sigma_{\ell,1}(x_1), \cdots, \sigma_{\ell,N_\ell}(x_{N_\ell})\Big),$$

where the scalar map $\sigma_{\ell,n} : \mathbb{R} \to \mathbb{R}$ is the activation function of the neuron indexed by (ℓ, n).

In essence, $\mathbf{W}_\ell$ encodes the strength of the neural connections from the previous layer, while $\boldsymbol{\sigma}_\ell$ represents the (parallel) responses of the N_ℓ neurons at layer ℓ. In the conventional setup, the response of the individual neurons is fixed and takes the form

$$\sigma_{\ell,n}(x) = \sigma(x - b_{\ell,n}), \tag{5.2}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a common activation function—typically, a sigmoid or a rectified linear unit (ReLU)—and $b_{\ell,n} \in \mathbb{R}$ is an adjustable bias [39]. In summary, the parameters $\boldsymbol{\theta}$ associated with the DNN in (5.1) are composed of the linear weights of $\mathbf{W}_\ell$ and the biases $\mathbf{b}_\ell \in \mathbb{R}^{N_\ell}$, $\ell = 1, \dots, L$.

The topic of this work deviates from the standard paradigm in the sense that it explores the option of adapting the responses of the individual neurons in an attempt to further improve the performance of such systems. In other words, instead of assigning a single bias parameter to each neuron as in (5.2), we investigate the possibility of redesigning or adjusting the activation functions $\sigma_{\ell,n} : \mathbb{R} \to \mathbb{R}$ on a neuron-by-neuron basis. While the typical way in which this can be achieved is via the introduction of a suitable parametrization, which may be linear or nonlinear, we will see that one can also formulate the problem in a functional framework with the help of a suitable regularization [164]. At any rate, the main point is that this augmented form of training results in a more difficult optimization problem and that it calls for more powerful algorithms.

The purpose of this work is to unify the parametric and functional approaches by representing the neural activation functions in terms of B-spline basis functions. This is possible as long as we restrict ourselves to the class of deep spline neural networks[3], which

[3]The denomination "deep spline neural network" refers to a DNN whose activation functions are linear splines, which includes ReLUs.

cover the complete family of continuous piecewise-linear (CPWL) mappings [165–167]. Our approach builds on the intimate connection between ReLU networks and splines, which has been observed by a number of authors [164, 168–171]. The spline interpretation is actually present at two levels: (i) the fact that such DNNs are describable as hierarchical splines and (ii) the property that the global response is CPWL, which allows one to interpret them as piecewise perceptrons [169]. While the local linear (perceptron-like) behavior of deep spline networks is both reassuring and enlightening, the part that is less obvious is the global continuity of the response, which ensures that the linear pieces (facets of polytopes) are seamlessly joined together.

The section is organized as follows: We start with a review of prior work on neural design in Section 5.1.2. In Section 5.1.3, we explain the main theoretical results on deep spline networks; namely, the CPWL property and the fact that they are optimal with respect to $\mathrm{TV}^{(2)}$ regularization. We then introduce our parametrization and optimization framework in Section 5.1.4 and present experimental results in Section 5.1.5.

5.1.2 Prior Work

We now briefly review the prior works on the design of neural activation functions, which can be broadly classified into three categories.

Inspiration from Neurophysiology

The traditional activation function for neural networks inspired by neurophysiology is a saturating sigmoid whose sharpness can be tuned for best performance [172]. Since splines have the ability to encode arbitrary functions, they can be used to generate a much richer variety of activation functions, which can then be optimized for best performance. Relevant examples of parametric activation function models for traditional neural networks include B-spline receptive fields [173], Catmull-Rom cubic splines [174, 175], and smooth piecewise polynomials [176].

Link with Iterative Soft-Thresholding Algorithms

One can make an interesting connection between neural networks and sparse-encoding techniques [20, 177] by considering the unrolled version of an iterative soft-thresholding algorithm (ISTA) [44, 178]. This connection suggests that the activation function fulfills the role of the nonlinearity in classical ISTA [21, 179]. Incidentally, the canonical nonlinearity associated with ℓ_1 minimization is an antisymmetric linear spline, which can be expressed as a linear combination of two ReLUs. In recent years, researchers have considered more general parametric nonlinearities whose weights are learned during training. Such models involve linear combinations of Gaussian radial-basis functions [45] and cubic B-splines [180, 181].

ReLU Variations

While many (fixed) activation functions σ in (5.2) have been considered in the literature, the preferred choice that has emerged over the years is the rectified linear unit $\mathrm{ReLU}(x) = (x)_+ \triangleq \max(0, x)$ [182]. In particular, it has been observed that ReLUs facilitate training [39]. Two ReLU variants, by order of improving performance, are "leaky ReLU" [183], in which the vanishing part of the response is replaced by one with a fixed nonzero linear slope, and "parametric ReLU" (PReLU) [184], where the linear slope is learnable. Also related to ReLU is Agostinelli *et al.*'s model of adaptive piecewise-linear (APL) units [185]. It results in an activation function that is a linear spline with a small fixed number of knots and has been found to outperform plain ReLU activation functions. Another instance is [186], where piecewise-linear units with learnable parameters are used as activation functions.

5.1.3 Theoretical Justification of Spline Activation Functions

Many of the state-of-the-art DNNs rely on ReLU activation functions or some variant thereof. Beside the issue of practical efficiency, a key feature of ReLU networks is that they result in a global continuous and piecewise-linear (CPWL) input-output relation. This is a fundamental property that generalizes to a wider class of spline activation functions and that also ensures that deep ReLU networks have universal approximation properties [187–189].

Deep Neural Nets as High-Dimensional Splines

A polynomial spline of degree 1 is a one-dimensional function that is continuous and piecewise-linear. In fact, the simplest nontrivial example of polynomial spline of degree 1 is $x \mapsto (x - b_k)_+ = \mathrm{ReLU}(x - b_k)$, which is made up of two linear pieces separated by a single knot at b_k. The concept is generalizable to higher dimensions [190, 191].

Definition 5.1 (CPWL function). *A function $f : \mathbb{R}^{N_0} \to \mathbb{R}$ is continuous piecewise-linear if*

1. *it is continuous $\mathbb{R}^{N_0} \to \mathbb{R}$;*
2. *its domain $\mathbb{R}^{N_0} = \bigcup_{k=1}^{K} P_k$ can be partitioned into a finite set of non-overlapping polytopes P_k over which it is affine.*

Likewise, a vector-valued function $\mathbf{f} = (f_1, \dots, f_N) : \mathbb{R}^{N_0} \to \mathbb{R}^N$ is CPWL if each of the component functions f_n is CPWL.

What is truly remarkable with CPWL functions is that they remain CPWL through the operations that typically occur in a deep neural network [166, 167]. Specifically,

1. any linear combination of CPWL functions is CPWL;
2. the composition of any two CPWL functions is CPWL;
3. the max or min of two CPWL functions is CPWL.

Since the functions $\boldsymbol{A}_\ell$ in (5.1) are trivially CPWL, the resulting DNN is CPWL whenever the pointwise nonlinearities $\boldsymbol{\sigma}_\ell$ are CPWL, for instance when they are piecewice-linear splines, which is indeed the case for deep ReLU networks. It is therefore perfectly legitimate to interpret deep ReLU networks—and, by extension, deep spline networks—as multidimensional splines of polynomial degree 1.

Variational Optimality of Deep Spline Networks

Lesser known is the property that the CPWL behavior can also be enforced indirectly through the use of an appropriate regularization [164]. To that end, one simply augments the cost functional that is used to train the network by an additive second-order total-variation regularization term for each adjustable activation function.

In our framework, we consider deep neural networks $\mathbf{f}_{\mathrm{deep}} : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ composed of L layers with the generic feedforward architecture described by (5.1).

The linear transformation in layer ℓ, represented by the matrix $\mathbf{W}_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, is associated with some free (adjustable) parameters $\boldsymbol{\phi}_\ell \in \mathbb{R}^{N_{\mathrm{lin},\ell}}$. In order to specify the latter, one has to distinguish between two configurations. When the layer is fully connected, $\boldsymbol{\phi}_\ell$ is the vectorized version of $\mathbf{W}_\ell$, which amounts to a total of $N_{\mathrm{lin},\ell} = N_{\ell-1} \times N_\ell$ tunable weights. The other important configuration is that of a convolutional layer where $\boldsymbol{\phi}_\ell$ contains much fewer convolution filter weights than $N_{\ell-1} \times N_\ell$. Similarly, to share nonlinearities across neurons, we specify each nonlinear mapping $\boldsymbol{\sigma}_\ell : \mathbb{R}^{N_\ell} \to \mathbb{R}^{N_\ell}$ by the vector $\mathbf{g}_\ell = (g_{\ell,1}, \dots, g_{\ell,N_{\mathrm{nonlin},\ell}})$ of adjustable activation functions $g_{\ell,n} : \mathbb{R} \to \mathbb{R}$, where $N_{\mathrm{nonlin},\ell} \in \mathbb{N}$ denotes the number of unique activation functions used in layer ℓ. For example, in a fully connected layer, it can be advantageous to use an independent activation function for each neuron. In this case, $N_{\mathrm{nonlin},\ell} = N_\ell$ and $g_{\ell,n} = \sigma_{\ell,n}$. By contrast, for convolutional layers, it is natural to use a single activation function per feature map, so that $N_{\mathrm{nonlin},\ell}$ will typically match the number of channels. When the same nonlinearity is shared across all channels, one has that $N_{\mathrm{nonlin},\ell} = 1$.

With this extended notation, given a dataset $\{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^{M}$, the training of the network is formulated as the functional optimization problem

$$\min_{\substack{\boldsymbol{\phi}_\ell \in \mathbb{R}^{N_{\mathrm{lin},\ell}} \\ \mathbf{g}_\ell \in \mathrm{BV}^{(2)}(\mathbb{R})^{N_{\mathrm{nonlin},\ell}}}} \sum_{m=1}^{M} \mathrm{E}(\mathbf{f}_{\mathrm{deep}}(\mathbf{x}_m), \mathbf{y}_m) + \sum_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell\|_2^2 + \sum_{\ell=1}^{L-1} \lambda_\ell \mathrm{TV}^{(2)}(\mathbf{g}_\ell), \tag{5.3}$$

where $\mathrm{E} : \mathbb{R}^{\mathrm{N_L}} \times \mathbb{R}^{\mathrm{N_L}} \to \mathbb{R}+$ is an arbitrary proper convex function and $\mathrm{TV}^{(2)}(\mathbf{g}) = \mathrm{TV}^{(2)}(g_1, \dots, g_N) = \sum_{n=1}^{N} \mathrm{TV}^{(2)}(g_n)$, where

$$\mathrm{TV}^{(2)}(g_n) = \left\|\mathrm{D}^2 g_n\right\|_{\mathcal{M}} \triangleq \sup_{\varphi \in \mathcal{S}(\mathbb{R}):\ \|\varphi\|_\infty \leq 1} \langle g_n, \frac{\mathrm{d}^2 \varphi}{\mathrm{d}x^2} \rangle \tag{5.4}$$

is the second-order total variation of the component function $g_n : \mathbb{R} \to \mathbb{R}$. Let us remark that the two first terms in (5.3) are the standard criteria used to train deep neural networks. The first (data loss) quantifies the goodness of fit, while the second (the so-called weight decay) favors solutions with a smaller amplitude of the linear weights $\boldsymbol{\phi}_\ell$. The novel element here is the additional optimization over the individual neuronal activation functions $\mathbf{g}_\ell$, which is made possible because of the inclusion of the third term: the sum of the second-order total variations of the trainable nonlinearities. Since this regularization only penalizes the second derivative of the activation function, it favors simple solutions—preferably linear or with "sparse" second derivatives—while ensuring that the activations be differentiable almost everywhere, which is essential for the backpropagation algorithm. For further explanation on the regularization functional $\mathrm{TV}^{(2)}$ and the definition of the search space $\mathrm{BV}^{(2)}(\mathbb{R})$, we refer to Appendix 5.4.1.

Unser's representer theorem for DNNs states that (5.3) admits a global minimizer (deep spline network) with neuronal activation functions of form

$$x \mapsto g_{\ell,n}(x) = b_{0,\ell,n} + b_{1,\ell,n} x + \sum_{k=1}^{K_{\ell,n}} a_{k,\ell,n}(x - \tau_{k,\ell,n})_+ \tag{5.5}$$

with $K_{\ell,n} \leq (M-2)$ and $\mathrm{TV}^{(2)}(g_{\ell,n}) = \sum_{k=1}^{K_{\ell,n}} |a_{k,\ell,n}| = \|\mathbf{a}_{\ell,n}\|_1$. Thus, the optimal activation functions are *adaptive* piecewise-linear splines. Specifically, every nonlinearity has a parametric description that is given by (5.5). It is characterized by its number $K = K_{\ell,n}$ of knots, the knot locations $\tau_1, \dots, \tau_K$, and the linear weights $\mathbf{b} \in \mathbb{R}^2, \mathbf{a} \in \mathbb{R}^K$, where we have dropped the network indices (ℓ, n) for simplicity. While Characterization (5.5) is elegant, it does not tell one how to determine the underlying parameters. We thus now face a more challenging optimization problem. The main complication is the allocation of knots—the determination of $K_{\ell,n}$ and the locations $\tau_{k,\ell,n}$ on a neuron-by-neuron basis—which is now also part of the problem.

Let us mention that we can also handle the case where the nonlinear mapping is shared across the layers, so that $\boldsymbol{\sigma}_1 = \cdots = \boldsymbol{\sigma}_L = \boldsymbol{\sigma}$ and $\boldsymbol{\sigma}$ is specified by a vector $\mathbf{g} =$

$(g_1, \dots, g_{N_{\text{nonlin}}})$ of adjustable scalar maps. Here, the training problem is formulated as

$$\min_{\substack{\boldsymbol{\phi}_\ell \in \mathbb{R}^{N_{\text{lin},\ell}} \\ \mathbf{g} \in \mathrm{BV}^{(2)}(\mathbb{R})^{N_{\text{nonlin}}}}} \sum_{m=1}^{M} \mathrm{E}(\mathbf{f}_{\text{deep}}(\mathbf{x}_m), \mathbf{y}_m) + \lambda \mathrm{TV}^{(2)}(\mathbf{g}) + \sum_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell\|_2^2. \tag{5.6}$$

By adapting Unser's representer theorem, we can show that the optimal shared activation functions have the same form as in (5.5).

Remarkably, the parametric form that results from the functional minimization of (5.3) is compatible with the model proposed by Agostinelli *et al.* [185]. They represent the activation functions as $g_{\ell,n}(x) = h_{\ell,n}(x - b_{\ell,n})$, where $h_{\ell,n}$ is an APL unit of the form

$$x \mapsto h_{\ell,n}(x) = (x)_+ + \sum_{k=1}^{K} a_{k,\ell,n}(-x + \tau_{k,\ell,n})_+. \tag{5.7}$$

Here, the number K of knots is fixed beforehand; the bias $b_{\ell,n}$, weights $a_{k,\ell,n}$, and knot locations $\tau_{k,\ell,n}$ are learnable parameters. While (5.7) bears a close resemblance to (5.5), there are a few key differences that we highlight here.

1. The justification of APL units in [185] is empirical while the spline parametrization of (5.5) is based on a global functional optimization.
2. The APL units involve a fixed ReLU positioned at 0, and so, unlike (5.5), they cannot reproduce all affine functions of the form $b_0 + b_1 x$.
3. The number of spline knots in APL units is fixed (and is the same for all neurons), whereas it is adaptive in our approach. In fact, the determination of $K_{\ell,n}$ is part of the optimization problem that we consider.
4. The ReLU weights of the APL units are either not constrained, or slightly regularized through some empirical ℓ_2-norm weight decay. By contrast, in our approach, the theory dictates the use of a sparsity-promoting ℓ_1-regularization. In fact, as we shall see in Section 5.1.4, the ℓ_1-norm regularization is of great practical significance as it allows us to control $K_{\ell,n}$ by removing unnecessary knots.

5.1.4 Optimization of Activation Functions

Convex Proxy for Shallow Networks

The major difficulty in optimizing the DNN with respect to the spline parameters in (5.5) is that the number $K = K_{\ell,n}$ of knots is unknown and that the activation model is nonlinear with respect to the knot locations $\tau_k = \tau_{k,\ell,n}$. Our workaround is to place a fixed but highly redundant set of knots on a uniform grid with a step size T. We then

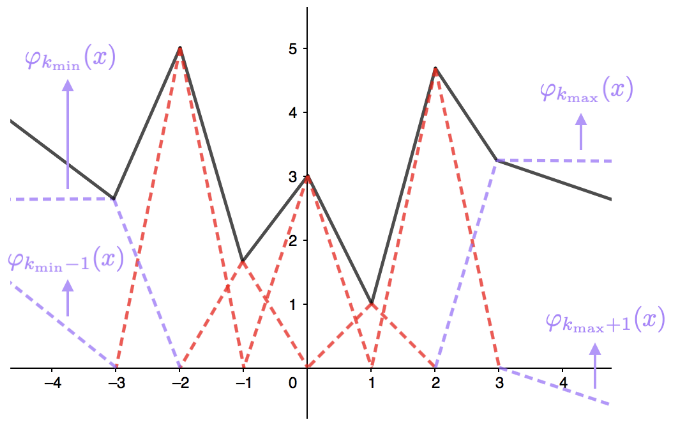


Figure 5.1: Decomposition of a deep spline activation function (solid line) in terms of B-spline basis functions (dashed lines), as expressed by (5.9) with $T = 1$. The basis is composed of $(K - 2)$ triangular functions, which are compactly supported and shifted replicates of each other, plus 4 one-sided outside functions. The key property is that the evaluation of $\sigma(x)$ for any fixed $x \in \mathbb{R}$ involves no more than two basis functions.

rely on the sparsifying effect of ℓ_1-minimization to nullify the coefficients of $\mathbf{a} = (a_k)$ that are not needed. This amounts to representing the spline activation functions by

$$\sigma(x) = b_0 + b_1 x + \sum_{k=k_{\min}}^{k_{\max}} a_k (x - kT)_+, \tag{5.8}$$

with $\mathrm{TV}^{(2)}(\sigma) = \|\mathbf{a}\|_1$. The consideration of the linear model (5.8), thereafter referred to as "gridded ReLU," gives rise to a classical ℓ_1-optimization problem that can be handled by most neural-network software frameworks. In the case of a shallow network with $L = 1$, it even results in a convex problem that is reminiscent of the LASSO [192]. We also note that (5.8) can be made arbitrarily close to (5.5) by taking T sufficiently small. While the solution $\mathbf{a}$ is expected to be sparse, with few active knots, the downside of the approach is that the underlying representation is cumbersome and badly conditioned due to the exploding behavior of the basis functions $(\cdot - kT)_+$ at infinity.

From ReLUs to B-Splines

While the direct connection with ℓ_1-minimization in (5.8) is very attractive, the less favorable aspect of the model is that its computational cost is proportional to the underlying number of ReLUs (or spline knots); that is, $K = (k_{\max} - k_{\min} + 1)$, which can be arbitrarily large depending on the value of T. Here, we propose a way to bypass this limitation by switching to another equivalent but maximally localized basis: the B-splines.

Our model takes the form

$$\sigma(x) = \sum_{k=k_{\min}-1}^{k_{\max}+1} c_k \varphi_k \left(\frac{x}{T}\right), \tag{5.9}$$

which involves triangular-shaped basis functions that are rescaled versions of B-splines defined on an integer grid. As illustrated in Figure 1, the central bases for $k = (k_{\min} + 1)$ to $(k_{\max} - 1)$ are shifted replicates of the compactly supported linear B-spline

$$\varphi_k(x) = \beta^1(x - k), \text{ for } k_{\min} < k < k_{max}, \tag{5.10}$$

where

$$\beta^1(x) = (x+1)_+ - 2(x)_+ + (x-1)_+ = \begin{cases} 1 - |x|, & x \in [-1, 1] \\ 0, & \text{otherwise.} \end{cases} \tag{5.11}$$

The four remaining boundary basis functions are one-sided splines that allow the activation function defined in (5.9) to exhibit a linear behavior at both ends, for $x < k_{\min}T$ as well as for $x > k_{\max}T$. Specifically, we have that

$$\varphi_{k_{\min}-1}(x) = (-x + k_{\min})_+ = \begin{cases} k_{\min} - x, & x < k_{\min} \\ 0, & \text{otherwise} \end{cases} \tag{5.12}$$

$$\begin{aligned} \varphi_{k_{\min}}(x) &= (-x + k_{\min} + 1)_+ - (-x + k_{\min})_+ \\ &= \begin{cases} 1, & x \le k_{\min} \\ 1 - (x - k_{\min}), & x \in (k_{\min}, k_{\min} + 1) \\ 0, & x \ge k_{\min} + 1 \end{cases} \end{aligned} \tag{5.13}$$

$$\begin{aligned} \varphi_{k_{\max}}(x) &= (x - k_{\max} + 1)_+ - (x - k_{\max})_+ \\ &= \begin{cases} 0, & x \le k_{\max} - 1 \\ x - k_{\max} + 1, & x \in (k_{\max} - 1, k_{\max}) \\ 1, & x \ge k_{\max} \end{cases} \end{aligned} \tag{5.14}$$

$$\varphi_{k_{\max}+1}(x) = (x - k_{\max})_+ = \begin{cases} 0, & x \le k_{\max} \\ x - k_{\max}, & x > k_{\max}. \end{cases} \tag{5.15}$$

The B-spline model defined in (5.9) has the same knots as those of the gridded ReLU representation given by (5.8). It also has the same number of degrees of freedom; namely, $K + 2 = (k_{\max} + 1) - (k_{\min} - 1) + 1$. By using the property that the φ_k can all be expanded in terms of integer shifts of ReLUs (see the central term of (5.10)-(5.15)), we can show that the two sets of basis functions span the same subspace. In doing so, we obtain a formula for the retrieval of the a_k and, hence, the $\mathrm{TV}^{(2)}(\sigma)$—in terms of the second-order difference of the c_k (see Appendix 5.4.2). While the gridded ReLU and B-spline models (5.8) and (5.9) are mathematically equivalent, the advantage of (5.9) is that there are at most two active basis functions at any given point $x = x_0$, independently of the step size T. This has important implications for the efficiency and scalability of both the evaluation of the DNN at a given point $\mathbf{x}_m$ and the computation of its gradient with respect to c_k (as opposed to a_k in the equivalent ReLU representation). Details of our implementation of the B-spline model are given in Appendix 5.4.2.

5.1.5 Experimental Results

In this section, we illustrate the capabilities of the proposed learning framework. Our main intent is to assess the benefit of optimizing the activation functions and to demonstrate the following claims:

1. The use of learned activation functions tends to improve the testing performance.
2. More complex activation functions can allow for simpler/smaller networks.
3. Learning with gridded ReLUs yields good performance for small values of K. However, the time and memory required for learning explodes as K grows.
4. The B-spline configuration is easy to train and is scalable in time and memory as K grows. Hence, it has the ability to learn more complex activation functions, which then typically also translates into better performance.

Further, we investigate the effect of the regularization parameter λ on the number of active knots in the learned spline activation functions and the performance of the neural network.

We consider both classification and signal-recovery (deconvolution) problems to highlight the versatility of our approach. The code (in PyTorch) is available on GitHub[4].

Classification

1. Area Classification

First, we discuss a simple two-class classification example with input dimension $N_0 = 2$. It allows us to obtain a better understanding of our learning scheme and to illustrate our claims visually.

[4]https://github.com/joaquimcampos/DeepSplines

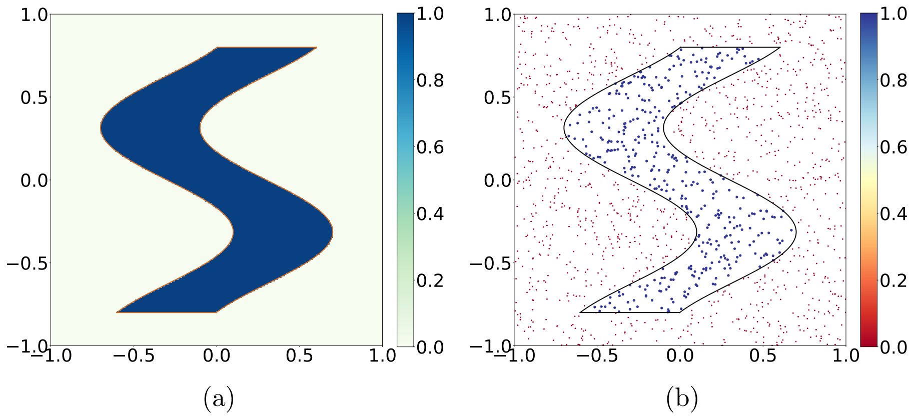


Figure 5.2: Ground truth and training dataset.

Setup

The task is to classify points in the two-dimensional space $[-1,1] \times [-1,1]$ as lying inside or outside an S shape (see Figure 5.2a). Mathematically, this region is represented by the binary function $f : [-1,1] \times [-1,1] \mapsto \{0,1\}$ given by

$$f(x_1,x_2) = \begin{cases} 1, & |x_1 - g(x_2)| \leq 0.3 \text{ and } |x_2| < 0.8, \\ 0, & \text{otherwise,} \end{cases} \tag{5.16}$$

where $g(x) = 0.4\sin(-5x)$. We generate training and validation datasets with $M = 1{,}500$ data points each. The coordinates $\mathbf{x}_m = (x_{1,m}, x_{2,m})$ of the data points are sampled from a uniform distribution on $[-1,1] \times [-1,1]$ and the labels y_m are assigned according to (5.16). The training dataset is shown in Figure 5.2b.

We tackle this problem using a fully connected network with N_h hidden layers, which takes a 2D input $\mathbf{x} = (x_1, x_2)$ and outputs a real value $\hat{f}(\mathbf{x}) \in [0,1]$. The number of neurons in each hidden layer is W; thus, the layer descriptor $(N_0, \dots, N_L)$ of the network is of the form $(2, W, \dots, W, 1)$. In the B-spline network, spline activation functions with $K = 19$ knots on a grid of size $T = 0.1$ are used as nonlinearities after each linear step, except the final one which involves a fixed sigmoid activation function. In the adaptive piecewise-linear unit (APLU) network, the nonlinearities take the form (5.7) with the number of adjustable knots set to $K = 19$. We compare the performance of our, ReLU, PReLU, and APLU networks on a test dataset that consists of 40,000 points that lie on a 2D grid of width 0.01×0.01 in $[-1,1] \times [-1,1]$. To evaluate the performance of these networks on a dataset, the output values $\hat{f}$ are quantized into predictions

$$\hat{f}_{\text{pred}}(\mathbf{x}) = \begin{cases} 1, & \hat{f}(\mathbf{x}) > 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{5.17}$$

The classification accuracy is computed as

$$\text{accuracy (\%)} = \frac{\#\text{ correct predictions}}{\#\text{ total predictions}} \times 100. \tag{5.18}$$

The binary cross-entropy loss is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{m=1}^{M}\Big((-y_m)\log(\hat{f}(\mathbf{x}_m)) - (1-y_m)\log(1-\hat{f}(\mathbf{x}_m))\Big), \tag{5.19}$$

where $\boldsymbol{\theta}$ represents the parameters of the network. This loss is chosen for the training process. In all the networks, the weights are initialized using Xavier's initialization [193]. For the B-spline network, half of the spline activation functions are initialized with σ_{abs} and the other half with σ_{soft}, where

$$\sigma_{\text{abs}}(x) = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{5.20}$$

$$\sigma_{\text{soft}}(x) = \begin{cases} x + \frac{1}{2}, & x \leq -\frac{1}{2} \\ 0, & x \in (-\frac{1}{2}, \frac{1}{2}) \\ x - \frac{1}{2}, & x \geq \frac{1}{2}. \end{cases} \tag{5.21}$$

This initialization is based on the fact that any function can be represented as the sum of an even and an odd function. In the APLU network, the ReLU weights $a_{k,\ell,n}$ and knot locations $\tau_{k,\ell,n}$ are initialized by randomly sampling them from zero-mean Gaussian distributions with standard deviations 0.1 and 1, respectively. The loss function is minimized over a total of 500 epochs using the ADAM optimizer [194]. The initial learning rate, set to 10^{-3}, is decreased by a factor of 10 at the epochs 440 and 480. A small batch size of 10 is helpful to avoid local minima.

Comparison with ReLU, PReLU, and APLU Networks

We compare in Figure 5.3 and Table 5.1 the performance of the ReLU, PReLU, B-spline, and APLU networks for three different architectures. For the B-spline networks, the optimal values of μ_ℓ and λ_ℓ, in terms of the performance for the validation dataset, are found using the method described in Appendix 5.4.3. The weight decays for the ReLU, PReLU, and APLU networks are tuned with the help of a grid search. In the APLU network, an ℓ_2-norm penalty with scaling factor 10^{-3} is also applied to the activation function parameters $(a_{k,\ell,n}, \tau_{k,\ell,n})$. With these optimal hyperparameters, the networks are then retrained 9 times independently. The median performance (over these 9 runs) for the test dataset is reported in Figure 5.3 and Table 5.1.

In the interest of fairness, in Table 5.1 we also mention the number of parameters associated with the networks. A fully connected network with N_h hidden layers has

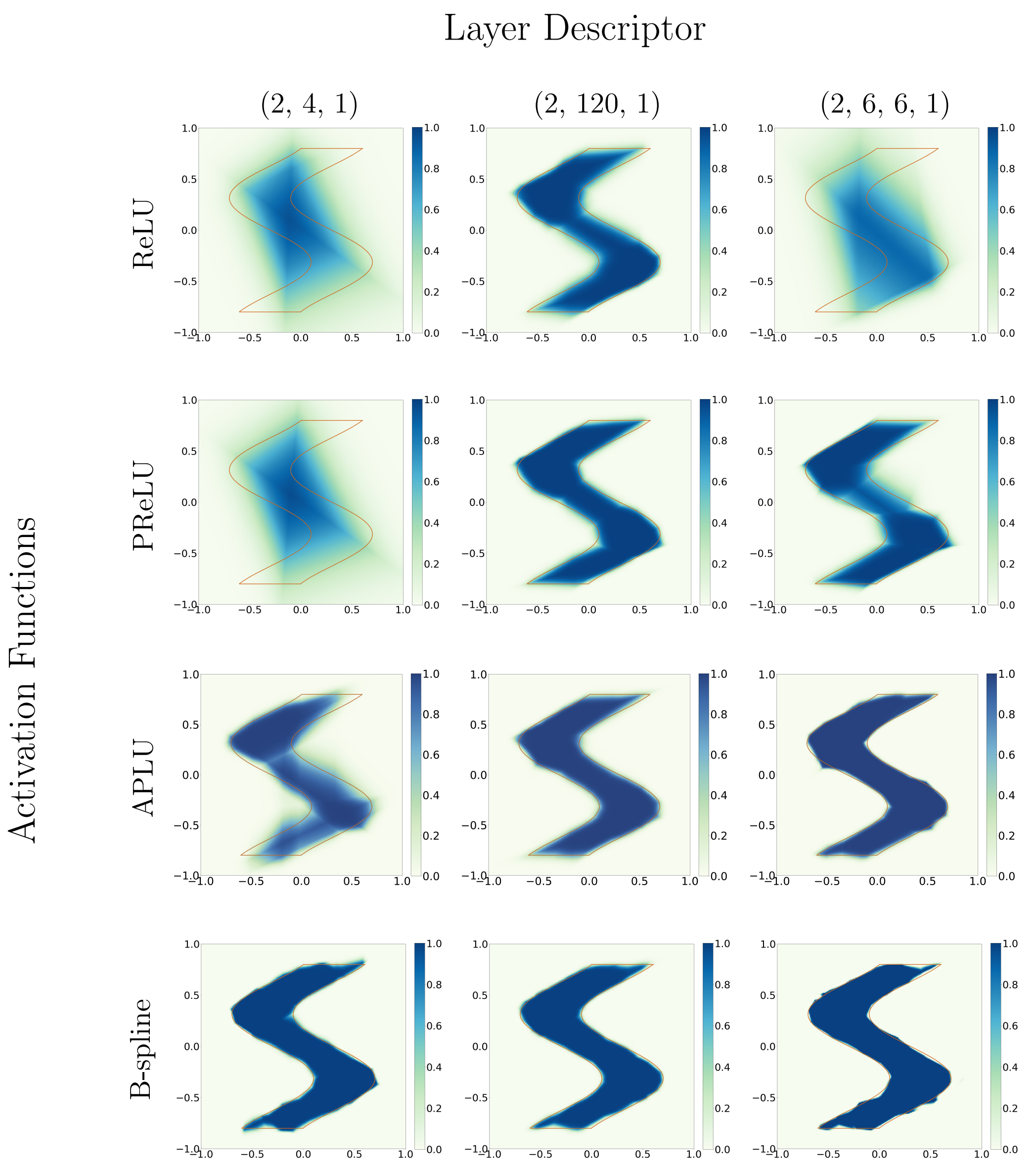


Figure 5.3: Learned probability maps for the area-classification problem.

$3W + (N_h - 1)W^2$ linear weights and 1 bias parameter for the fixed sigmoid activation function. The network also has some additional parameters that depend on the choice of the activation function. The ReLU networks have N_hW biases while, in addition to these biases, the PReLU networks have N_hW learnable parameters that represent the linear slopes of the PReLU activation functions in $\mathbb{R}^-$. In the B-spline networks, the number of additional parameters (per activation function) is equal to the number of active knots in the learned linear-spline nonlinearity plus the 2 coefficients that determine its linear (null-space) component. Lastly, the APLU networks have $(2K + 1)$ additional parameters per activation function, where the number of adjustable knots K was set to

Table 5.1: Number of parameters and classification error rate.

	Architecture	N_{param}	Error rate (%)
ReLU	(2,4,1)	17	17.02
	(2,120,1)	481	2.59
	(2,6,6,1)	67	15.39
PReLU	(2,4,1)	21	17.00
	(2,120,1)	601	1.87
	(2,6,6,1)	79	2.89
APLU	(2,4,1)	169	5.15
	(2,120,1)	5041	1.64
	(2,6,6,1)	523	1.42
B-spline	(2,4,1)	68	1.66
	(2,120,1)	822	1.40
	(2,6,6,1)	171	1.60

19 beforehand.

For the simplest architecture $(2, 4, 1)$, we observe that the B-spline and APLU networks outperform the ReLU and PReLU models which lack capacity and perform rather poorly. This demonstrates that the learning of activation functions improves the accuracy; more so, if the activation function has reasonably many learnable parameters.

Remarkably, the simplest B-spline network outperforms the ReLU and PReLU networks with richer architectures despite having fewer parameters. This is because it is capable of learning more complex activation functions. This, in turn, translates into an overall map that is more faithful to the gold standard—the ideal S shape. This suggests that, instead of making the architecture of the network more complex, for example by including more neurons in the initial layers and/or adding more layers, one can increase the accuracy by relying on more sophisticated, learnable nonlinearities.

The results of Table 5.1 also illustrate the advantages of our learning scheme over the APL units. For the architecture $(2, 4, 1)$, the B-spline network yields a better accuracy than the APLU network even though it has fewer parameters. One possible explanation is that the APL units, which have a fixed number of knots, face difficulties in optimizing their knot locations, whereas the adaptive B-splines bypass this problem with the help of a grid and sparsity-promoting ℓ_1-regularization. Another possible reason could be the ill-conditioned nature of expansion (5.7), in the sense that a small perturbation of one ReLU coefficient

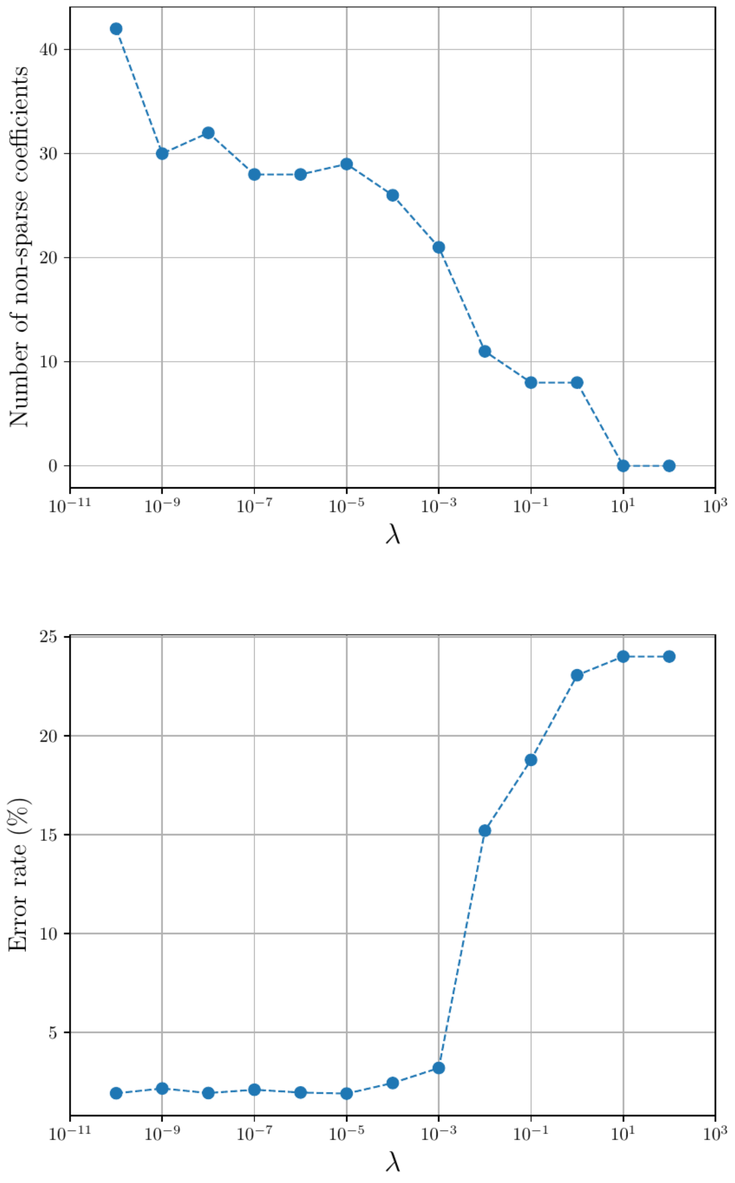


Figure 5.4: Effect of λ on the number of active knots and on the classification error.

has a nonlocal effect on the activation function, which makes the optimization task more challenging. For the other two richer architectures, we get similar performances for the APLU and B-spline networks. However, the B-spline networks require fewer knots.

Effect of the Regularization Parameter λ

We consider now a B-spline network with layer descriptor $(2, 4, 1)$ for the area-classification task. The weight decay is fixed as $\mu_1 = \mu_2 = 10^{-4}$ and λ is varied in the interval $[10^{-10}, 10^{2}]$. For each value of λ, 10 independent models are trained on the training dataset. The median number of total active knots[5] and the classification error (on the test dataset) of the corresponding model are shown as functions of λ in Figure 5.4.

The number of active knots decreases (or, equivalently, the sparsity of the learned activation functions increases) as λ increases, which means that the hyperparameter λ controls the complexity of the network. The performance of the network remains (nearly) constant, up to a critical value of λ, after which it begins to deteriorate. This is crucial since it suggests that, by carefully tuning λ, we can obtain simpler networks that still perform well.

2. CIFAR-10 and CIFAR-100

Now, we look at the application of the proposed learning scheme to the classification of standard datasets such as CIFAR [195]. We consider two network architectures—the network-in-network [196] (NIN) and a deep residual network [197] (ResNet32) for the CIFAR-10 and CIFAR-100 classification tasks. Each dataset consists of 50,000 training images and 10,000 test images of size (32×32).

First, we compare the performance of the B-spline, ReLU, and APLU networks. We then also demonstrate the advantages of our B-spline solution over its gridded ReLU counterpart. In the B-spline networks (NIN and ResNet), we use spline nonlinearities with $K = 49$ knots on a grid of size $T = 0.16$. We rely on one activation function per output channel for the convolutional layers and one spline activation function per output unit for the fully connected layers. For the APLU networks[6] (NIN and ResNet), we set the number of adjustable knots to $K = 1$, with one APL activation function per output unit for the convolutional layers as well as the fully connected layers. All networks include a softmax unit in the final layer and are trained by minimizing the categorical cross-entropy loss.

For each dataset, 5,000 samples are reserved for validation during training, while the remaining 45,000 samples are augmented as in [197]. The weights in the NINs are initialized by random sampling from a Gaussian distribution with zero mean and a standard deviation of 0.05. The weights in the ResNets are initialized using He's recipe [184]. The B-spline activation functions are initialized as leaky ReLUs while the APL units are initialized in the same manner as in the area-classification experiment. For the B-spline NIN, B-spline ResNet, and APLU ResNet, the parameters of the activation functions are updated using the ADAM optimizer [194] with an initial learning rate of 10^{-3}. The remaining network parameters are updated using an SGD optimizer with an initial learning rate of 10^{-1}. For the APLU NIN, an SGD optimizer with an initial learning rate of 10^{-1} is used to update all the learnable parameters. The NINs are trained for 320 epochs with a batch size of 128. The learning rate is decreased by a factor of 10

[5]The number of total active knots is the sum of the number of active knots or ReLUs in each learned activation function in the network.

[6]The reported configurations are the ones that were found to give the best performance.

Table 5.2: NIN error rates on CIFAR-10 and CIFAR-100.

Activation function	CIFAR-10	CIFAR-100
ReLU	8.78%	32.44%
APLU	8.71%	31.74%
B-spline	8.29%	30.43%

Table 5.3: ResNet error rates on CIFAR-10 and CIFAR-100.

Activation function	CIFAR-10	CIFAR-100
ReLU	6.31%	29.02%
APLU	6.45%	28.85%
B-spline	6.02%	28.24%

in epochs 80, 160, and 240. The ResNets are trained for 300 epochs with a batch size of 128 while the learning rate is divided by 10 in epochs 150, 225, and 262, following the training scheme in [198].

Comparison with ReLU and APLU Networks
For the ReLU networks (NIN and ResNet), we deploy a grid search to optimize the weight decays in terms of the performance on the validation dataset. For the B-spline and APLU networks, we use the same weight decays as those found for the corresponding ReLU networks, and we perform grid searches to find the optimal values of λ and the ℓ_2-norm penalty scaling factor. We then use the optimal hyperparameters and retrain the networks N_T times independently on the complete training datasets, with 50,000 samples. We set $N_T = 5$ for the NINs and $N_T = 9$ for the ResNets. Finally, we compute the error rates over the test datasets. The median test errors are reported in Table 5.2 and Table 5.3. We see that the B-spline networks outperform the ReLU and APLU networks here as well. Surprisingly, the APLU ResNet is slightly inferior to the ReLU ResNet for the CIFAR-10 dataset. It turns out that, for residual networks with APL units, a similar observation has been made in [199].

B-splines vs. gridded ReLUs vs. APLUs
In this experiment[7], we record the memory consumption and computation time (per epoch) for the B-spline, gridded ReLU, and APLU ResNets.

As we see in Table 5.4, the time/memory consumption during forward and backward propagation for gridded ReLUs and APLUs explodes with the number of knots. This is because the point evaluation of an activation function requires a summation over

[7]This experiment was run on a TITAN X (Pascal) GPU with 12196 MB of memory.

Table 5.4: B-splines *vs.* gridded ReLUs *vs.* APLUs

Architecture, Nb. coefficients	Memory (megabytes)	Time per epoch (seconds)
B-splines, $K = 9$	1132	44.92
B-splines, $K = 29$	1133	41.89
B-splines, $K = 499$	1299	41.19
Gridded ReLUs, $K = 9$	3313	49.86
Gridded ReLUs, $K = 29$	9616	81.21
APLUs, $K = 9$	3316	49.72
APLUs, $K = 29$	9618	87.34

For the gridded ReLU and APLU networks, the maximum number of knots allowed by the GPU memory is 31.

all contributing ReLUs, which results in a time complexity of $O(K)$. Moreover, the corresponding intermediate values need to be stored for backpropagation.

For B-splines, by contrast, each evaluation only requires the coefficients of two adjacent basis functions, since the $\varphi_{k,T}$ have minimal overlap, leading to a time complexity of $O(1)$. Accordingly, one only needs to store the coefficients and the index of the two active basis functions.

Signal Recovery

We further illustrate the benefits of learning the activation functions through the application of convolutional neural networks (CNNs) to inverse problems [37]. Here, the goal is to recover a signal $\mathbf{s} \in \mathbb{R}^d$ from its (noisy) measurements $\mathbf{y} \in \mathbb{R}^m$ given by

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}, \tag{5.22}$$

where $\mathbf{H} : \mathbb{R}^d \mapsto \mathbb{R}^m$ is a linear operator that describes the measurement acquisition process and $\mathbf{n} \in \mathbb{R}^m$ is an additive noise. In our experiments, we consider CNN-based regression schemes that relate an initial estimate of the signal to the desired estimate of the signal [41, 43]. Specifically, we compare the performance of standard ReLU CNNs with B-spline CNNs in a deconvolution task.

1. Setup
We consider the recovery of piecewise-constant statistical signals $\mathbf{s} \in \mathbb{R}^{100}$ that satisfy the discrete innovation model

$$\mathbf{u} = \mathbf{D}\mathbf{s}, \tag{5.23}$$

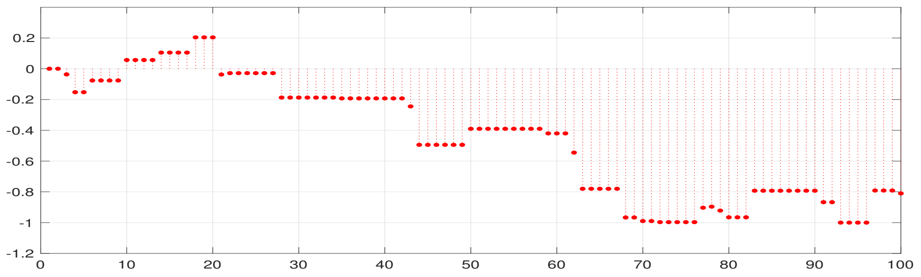


Figure 5.5: Piecewise-constant signal generated according to (5.25).

where $\mathbf{D} \in \mathbb{R}^{100\times 100}$ is a finite-difference matrix and $\mathbf{u} \in \mathbb{R}^{100}$ is a sparse random vector with independent and identically distributed entries that are drawn from the Bernoulli-Laplace distribution

$$p_U(u) = (0.6)\delta(u) + (0.4)\frac{1}{2}\mathrm{e}^{-|u|}. \tag{5.24}$$

Under appropriate boundary conditions, we can invert (5.23) and derive the synthesis formula

$$s_k = \sum_{q=1}^{k} u_q, \qquad k = 1, 2, \ldots, 100, \tag{5.25}$$

which has been used for our experiments. The dynamic range of each generated signal $\mathbf{s}$ is adjusted so that its values lie in $[-1, 1]$. An example of such a signal is shown in Figure 5.5.

The noiseless measurement vector $\mathbf{y}_0 \in \mathbb{R}^{88}$ is obtained by convolving the signal $\mathbf{s}$ with a Gaussian kernel of standard deviation $\sigma = 2$ and support $(6\sigma + 1) \times 1$. The resulting discrete-system matrix $\mathbf{H} \in \mathbb{R}^{88\times 100}$, such that $\mathbf{y}_0 = \mathbf{Hs}$, is

$$\mathbf{H} = \begin{bmatrix} h_{13} \cdots h_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots\ \ddots & \vdots \\ \vdots & \ddots\ \ddots & \ddots & 0 \\ 0 & \cdots & 0 & h_{13} \cdots h_1 \end{bmatrix}, \tag{5.26}$$

where $\mathbf{h} \in \mathbb{R}^{13}$ denotes the truncated Gaussian kernel. Finally, we add a white Gaussian noise $\mathbf{n} \in \mathbb{R}^{88}$ to the noiseless measurements $\mathbf{y}_0$ such that the input SNR, defined as

$$\mathrm{SNR}(\mathbf{y}_0 + \mathbf{n}) = 20 \log_{10}\left(\|\mathbf{y}_0\|_2/\|\mathbf{n}\|_2\right), \tag{5.27}$$

is equal to 20dB.

Our training dataset for the CNN-based approaches consists of M_t = 10,000 samples. Meanwhile, the validation and test datasets contain 1,000 samples each.

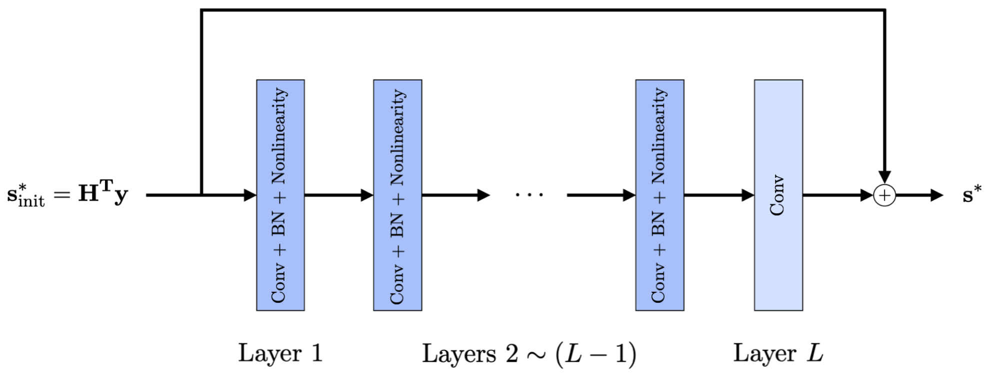


Figure 5.6: Architecture of the convolutional neural network. In a ReLU CNN, the nonlinearity is the ReLU, while in a B-spline CNN, the nonlinearity is a learnable linear spline.

Table 5.5: Convolution Layers

Layer number	(Filter size, number of input channels, number of output channels)
1	$(3 \times 1, 1, C)$
$2 \sim (L-1)$	$(3 \times 1, C, C)$
L	$(3 \times 1, C, 1)$

Similar to the work in [41, 43], we train CNNs to learn a mapping from an initial estimate of the signal (in our case $\mathbf{s}^*_{\text{init}} = \mathbf{H}^T\mathbf{y}$) to the desired estimate $\mathbf{s}^*$ of the signal. The architecture of the network is shown in Figure 5.6 and the details of the convolutional layers are provided in Table 5.5. For all our experiments, the number of channels is set as $C = 5$. In the B-spline CNN, we have learnable linear-spline activation functions with $K = 49$ knots on a grid of size $T = 0.1$.

The loss function used for training is

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{m=1}^{M} \|\mathbf{s}_m - \mathbf{s}^*_m(\boldsymbol{\theta})\|_2^2, \tag{5.28}$$

where $\boldsymbol{\theta}$ represents the parameters of the network. All the activation functions in the B-spline CNN are initialized as leaky ReLUs with negative slopes set to 0.1. The loss function is minimized using the ADAM optimizer. The networks are trained for 150 epochs with a batch size of 20. For ReLU CNNs, the initial learning rate is set as 10^{-2} and is decreased by a factor of 0.5 in the epochs $[25, 50, 75, 100, 125]$. The same learning rate schedule is also used for B-spline CNNs with $L \leq 7$. For B-spline CNNs with more layers ($L > 7$), the initial learning rate is 10^{-3} and is decreased by a factor of 0.5 in the epochs $[50, 75, 100, 125]$.

2. Results and Discussion

In our experiments, we compare the CNN-based approaches with the total-variation (TV) method [25] given by

$$\mathbf{s}^*_{\mathrm{TV}} = \arg\min_{\mathbf{s}\in\mathbb{R}^N} \left(\|\mathbf{y} - \mathbf{H}\mathbf{s}\|_2^2 + \tau\|\mathbf{D}\mathbf{s}\|_1 \right), \tag{5.29}$$

where $\tau \in \mathbb{R}_+$ is a parameter that controls the regularization strength. It is known to promote piecewise-constant solutions and is well matched to the signals that we consider here. In order to make a fair comparison with the CNNs, we use the same regularization parameter τ in the TV method for every signal in the test dataset. This value of τ is the one that gives the best performance in terms of the mean-square error or, equivalently,

$$\mathrm{SNR}(\mathbf{s}^*, \mathbf{s}) = 20\log_{10}\left(\|\mathbf{s}\|_2 / \|\mathbf{s}^* - \mathbf{s}\|_2 \right) \tag{5.30}$$

for the validation dataset.

Sharing versus Unsharing

We consider four configurations for the B-spline CNN. The first is the fully shared network, where a single learnable spline activation function is shared across all layers and channels. The second and third are the channel (layer, respectively) shared network, where the nonlinearity is shared only across channels (layers, respectively). The fourth is the unshared network, which has an independent nonlinearity in each layer and channel.

In a first experiment, we compare the performance of these four configurations. We fix the number of layers to $L = 4$. In the B-spline CNN, we rely on our hyperparameter-tuning method (see Appendix 5.4.3) to find the optimal μ_ℓ and λ_ℓ in terms of performance for the validation dataset. The weight decay for the ReLU CNN is chosen via grid search. Using the optimal values, we retrain the networks 9 times independently; the median SNR over these 9 runs is shown for the test dataset in Table 5.6, where B-CNN means B-spline CNN.

We provide the number of parameters for the networks in Table 5.6. A ReLU CNN with L layers, C channels, and filter size $(w \times 1)$, has $2wC + (L-2)wC^2$ convolutional-filter weights, 1 bias term for the last convolutional layer, and $2(L-1)C$ batch-normalization parameters. The additional parameters in the B-spline CNN are the total number of active knots in the learned spline activation functions.

We observe that all four versions of the B-spline CNN achieve a higher SNR than the ReLU CNN. This further supports our claim that learning the activation functions tends to improve the performance of the network. Moreover, as expected, configurations with a greater number of parameters perform better. The option of sharing the learnable spline

Table 5.6: Sharing *versus* unsharing of the linear spline activation functions in B-spline CNNs ($L = 4$).

Method	N_{param}	SNR (dB)	Time per epoch (seconds)
ReLU CNN	211	14.95	4.59
Fully shared B-CNN	253	15.09	16.05
Channel shared B-CNN	346	15.16	17.04
Layer shared B-CNN	456	15.23	15.90
Unshared B-CNN	830	15.36	17.78

nonlinearities makes our framework flexible and allows us to benefit from the increased capacity of the network while introducing fewer additional parameters. Also, note that Table 5.6 confirms that the running times for the different versions of the B-spline CNN are nearly the same.

Increase in the Depth of the Networks
Next, we compare the ReLU CNN and the fully shared B-spline CNN when the number of layers increases. The procedure of the previous experiment is followed again to set the hyperparameters and to report the performance of the test dataset (see Table 5.7).

We observe that the CNN-based approaches outperform the TV method, despite it being particularly well matched to the piecewise-constant signals that we consider. This shows the advantage of using learning-based methods over model-based ones when sufficient training data is available. For all values of L, the B-spline CNN achieves a higher SNR than the ReLU CNN. However, this improvement in performance diminishes as L increases and is negligible for $L \geq 10$. We believe that, when the network is sufficiently deep, the ReLU CNN has a sufficient representation power and so the additional capacity offered by the B-spline CNN does not translate into better performance. The main takeaway here is that learning the activation functions results in a noticeable improvement in performance for simpler/smaller networks, which are desirable for a number of reasons such as better interpretability of the networks, computational efficiency, and controlled Lipschitz constants [200].

5.1.6 Summary

We have presented an efficient computational solution to train deep neural networks with learnable activation functions. Specifically, we have focused on deep spline networks. They form a superset of the traditional ReLU networks and are known to be optimal with respect to the second-order total variation of the adjustable nonlinearities. We have tackled the resulting difficult joint-optimization problem by representing the linear-spline

Table 5.7: Performance of deep networks.

Method	Layers	Parameters	Performance
ReLU CNN	4	211	14.95
	5	296	15.27
	6	381	15.47
	7	466	15.68
	8	551	15.74
	10	721	15.80
	15	1146	15.84
Fully shared B-CNN	4	253	15.09
	5	339	15.34
	6	430	15.59
	7	503	15.73
	8	587	15.79
	10	760	15.81
	15	1196	15.85
TV	-	-	14.92

nonlinearities in terms of B-spline basis functions and by expressing the second-order total-variation regularization as an ℓ_1-penalty, thus unifying the parametric and functional approaches for the learning of activation functions. The proposed B-spline representation was instrumental in making the training of the DNN computationally feasible. Indeed, any computation concerning the activation functions involves only two basis elements per data point. Finally, we have demonstrated the benefits of our framework through experiments in the context of classification and deconvolution problems. In particular, we have observed that our method compares favorably to the traditional ReLU networks, the improvement being more pronounced for simpler/smaller networks.

5.2 Lipschitz-Constrained Neural Networks for Plug-and-Play Reconstruction

[8]In this section, we build upon the previously presented framework of deep spline neural networks (DSNNs) to design expressive 1-Lipschitz denoising networks, which can be deployed within provably convergent plug-and-play reconstruction schemes.

5.2.1 Introduction

In linear inverse problems, the goal is to reconstruct an image $\mathbf{s} \in \mathbb{R}^d$ from measurements $\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \in \mathbb{R}^m$. The linear operator[9] $\mathbf{H}\colon \mathbb{R}^d \to \mathbb{R}^m$ models the acquisition system and $\mathbf{n} \in \mathbb{R}^m$ is a realization of additive white Gaussian noise. Here, the MPL estimator for the image is given by

$$\mathbf{s}^* = \arg\min_{\mathbf{s}\in\mathbb{R}^d} \left(\frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{s}\|_2^2 + \tau\mathcal{R}(\mathbf{s}) \right), \tag{5.31}$$

where the regularization term $\mathcal{R}\colon \mathbb{R}^d \to \mathbb{R}_+$ imposes some prior knowledge about the image $\mathbf{s}$ and $\tau \in \mathbb{R}_+$ is a tunable hyperparameter. The cost functional in (5.31) is typically minimized using proximal algorithms such as forward-backward splitting (FBS) [87] and the alternating direction method of multipliers (ADMM) [24].

As mentioned in Chapter 2, the main idea in the Plug-and-Play (PnP) priors framework [51, 201] is to replace the proximal operator of $\mathcal{R}$ in the iterations of proximal algorithms with some off-the-shelf denoiser, even though it might not correspond to an explicit regularization term. This implicit regularization approach has been shown to yield better results than classical sparsity-promoting methods for a variety of inverse problems since it allows the use of powerful denoisers such as NLM [202], WNNM [203], BM3D [204], and neural networks [57, 62, 205]. However, the delicate point that remains is ensuring the convergence of these algorithms, which is non-trivial but essential for sensitive applications (e.g., the ones encountered in medical imaging).

There exist several works that analyze conditions on the denoiser under which PnP algorithms are guaranteed to converge [206–210]. For example, Ryu *et al.* [57] show that PnP-FBS and PnP-ADMM provably converge to fixed points if the denoiser obeys an appropriate Lipschitz condition. They then propose a practical way to enforce the derived Lipschitz constraint while training neural network denoisers. However, their analysis requires the data-fidelity term to be strongly convex and this unfortunately rules out ill-posed inverse problems. In order to design convergent PnP schemes for ill-posed problems, stricter conditions need to be enforced on the denoiser. More specifically, it has been

[8]This section is based on our works [157, 158].

[9]We assume that the inverse problem is ill-posed, that is, $\mathbf{H}$ is non-invertible.

shown that averagedness (firm nonexpansiveness) of the denoiser is sufficient to guarantee fixed point convergence of PnP-FBS (PnP-ADMM) [54, 55]. The design and training of constrained neural networks to satisfy the averagedness or firm nonexpansiveness conditions is a challenging task and remains an active area of research [55, 63].

In this work, we focus on the problem of training 1-Lipschitz[10] (nonexpansive) neural networks in order to construct averaged denoisers that can be used within PnP-FBS. Specifically, we consider networks where the Lipschitz constant of each layer (linear and nonlinear) is controlled to be one. Henceforth, we refer to such networks as *1-Lip neural networks.*

There are several ways to impose constraints on the linear layers. The most popular one is spectral normalization [211], where the ℓ_2 operator norm of each weight matrix is set to one. The required spectral norms are computed via power iterations. To take this idea even further, [212] and [213] have restricted the weight matrices to be orthonormal in fully connected layers.

The use of ReLU activation functions in that setting, however, appears to be overly constraining: it has been shown that 1-Lip ReLU networks cannot even represent simple functions such as the absolute value function under 2-norm constraints on the linear layers [212], as well as ∞-norm constraints [214]. This observation justifies the development of new activation functions specifically tailored to 1-Lip architectures. Currently, the most popular one is GroupSort (GS), proposed by [212], where the pre-activations are split into groups that are sorted in ascending order. This results in a multivariate and gradient-norm-preserving (GNP) activation function. The authors provide empirical evidence that GS outperforms ReLU on several tasks such as Wasserstein-1 distance estimation, robust classification, and function fitting under Lipschitz constraints.

Here, we propose an alternative way to boost the performance of 1-Lip neural networks via the use of component-wise 1-Lipschitz learnable-linear-spline (LLS) activation functions. Since the LLS activation functions presented in Section 3.1 are generally not Lipschitz-constrained, we present an efficient method to explicitly control their Lipschitz constant. Further, we also develop a normalization module that modulates the scale of each LLS activation function without changing their Lipschitz constant and thus increases their flexibility. We perform a systematic comparison of the proposed framework with other 1-Lip architectures (including ReLU and GS) for function fitting, Wasserstein-1 distance estimation, and CT and MRI reconstruction within the PnP framework. Our results show that our framework outperforms all the competing activation functions.

The section is organized as follows: We begin with a brief description of PnP-FBS in Section 5.2.2. In Section 5.2.3, we present existing 1-Lip architectures. We then introduce

[10]An operator $\mathrm{T}\colon \mathbb{R}^K \to \mathbb{R}^K$ is L-Lipschitz if $\|\mathrm{T}(\mathbf{x}) - \mathrm{T}(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^K$. The smallest value of L is called the Lipschitz constant of T. In this section, we only consider $\|\cdot\|$ to be the 2-norm (also known as the Euclidean norm).

our method in Section 5.2.4 and present experimental results in Section 5.2.5.

5.2.2 Plug-and-Play Forward-Backward Splitting (PnP-FBS)

The iterates for PnP-FBS corresponding to the optimization problem in (5.31) are given by

$$\mathbf{s}_{k+1} = \mathrm{D}\Big(\mathbf{s}_k - \alpha \mathbf{H}^T(\mathbf{H}\mathbf{s}_k - \mathbf{y})\Big), \tag{5.32}$$

where $\mathrm{D} : \mathbb{R}^d \to \mathbb{R}^d$ is the chosen denoiser and α is the stepsize.

Fixed-Point Convergence

A standard set of sufficient conditions to guarantee fixed-point convergence of the iterations (5.32) is that

1. D is averaged, namely $\mathrm{D} = \beta\mathrm{N} + (1-\beta)\mathrm{Id}$ where $\beta \in (0,1)$ and $\mathrm{N}\colon \mathbb{R}^d \to \mathbb{R}^d$ is a nonexpansive mapping;
2. $\alpha \in [0, 2/\|\mathbf{H}\|^2)$;
3. the update operator in (5.32) has a fixed point.

Note that in general, Condition (1) is not sufficient to ensure that D is the proximal operator of some convex regularizer $\mathcal{R}$. Hence, its interpretability is still limited. Condition (2) implies that $\mathbf{s} \mapsto \big(\mathbf{s} - \alpha\mathbf{H}^T(\mathbf{H}\mathbf{s} - \mathbf{y})\big)$ is averaged. As averagedness is preserved through composition, the iterates are updated by the application of an averaged operator (see [55] for details). With Condition (3), the convergence of the iterations (5.32) follows from Opial's convergence theorem.

Stability of the Reconstruction Map in the Measurement Domain

Beyond convergence, we can also show the stability of the reconstruction map in the measurement domain.

Proposition 5.1. *Let $\mathbf{s}_1^*$ and $\mathbf{s}_2^*$ be fixed points of the PnP-FBS algorithm* (5.32) *for the measurements $\mathbf{y}_1$ and $\mathbf{y}_2$, respectively. If the denoiser is averaged with $\beta \leq 1/2$, then for any $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^m$, we have that*

$$\|\mathbf{H}\mathbf{s}_1^* - \mathbf{H}\mathbf{s}_2^*\|_2 \leq \|\mathbf{y}_1 - \mathbf{y}_2\|_2. \tag{5.33}$$

Further, if we slightly increase the constraints on D, we get the result of Proposition 5.2

Proposition 5.2. *Let* $\mathbf{s}_1^*$ *and* $\mathbf{s}_2^*$ *be fixed points of the PnP-FBS algorithm* (5.32) *for the measurements* $\mathbf{y}_1$ *and* $\mathbf{y}_2$*, respectively. If* D *is* K*-Lipschitz with* $K < 1$ *(if it is contractive), then the reconstruction process is stable for any* $\mathbf{H}$*, in the sense that, for any* $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^m$*,*

$$\|\mathbf{s}_1^* - \mathbf{s}_2^*\|_2 \ \leq \ \frac{\alpha\|\mathbf{H}\|K}{1-K}\|\mathbf{y}_1 - \mathbf{y}_2\|_2. \tag{5.34}$$

The proofs for these propositions are given in Appendix 5.4.4. Overall, with PnP-FBS, we can expect better data consistency than the one provided by the end-to-end neural network frameworks that attempt to directly map the measurements $\mathbf{y}$ to the image $\mathbf{s}$. Those attempts are known to suffer from stability issues [215] and, more importantly, have been found to remove or hallucinate tumors [216], which is unacceptable in the context of diagnostic imaging. Relations (5.33) and (5.34) are protection against such hallucinations. They tell us that, if two sets of measurements are close to each other, then the corresponding reconstructions must also be close to each other.

5.2.3 1-Lipschitz Neural Networks

In this work, we consider feedforward neural networks $\mathbf{f}_{\boldsymbol{\theta}} \colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ of the form

$$\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) : \boldsymbol{A}_L \circ \cdots \circ \boldsymbol{\sigma}_\ell \circ \boldsymbol{A}_\ell \circ \cdots \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{A}_1(\mathbf{x}), \tag{5.35}$$

where each $\boldsymbol{A}_\ell \colon \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \dots, L$, is a linear layer given by

$$\boldsymbol{A}_\ell(\mathbf{x}) = \mathbf{W}_\ell \mathbf{x} + \mathbf{b}_\ell, \tag{5.36}$$

with weight matrices $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell, N_{\ell-1}}$ and bias vectors $\mathbf{b}_\ell \in \mathbb{R}^{N_\ell}$. The model incorporates fixed or learnable nonlinear activation functions $\boldsymbol{\sigma}_\ell \colon \mathbb{R}^{N_\ell} \to \mathbb{R}^{N_\ell}$. For component-wise activation functions, we have that $\boldsymbol{\sigma}_\ell(\mathbf{x}) = (\sigma_{\ell,n}(x_n))_{n=1}^{N_\ell}$ with individual scalar activation functions $\sigma_{\ell,n} \colon \mathbb{R} \to \mathbb{R}$. The complete set of parameters of the network is denoted by $\boldsymbol{\theta}$.

A straightforward way to control $\mathrm{Lip}(\mathbf{f}_{\boldsymbol{\theta}})$ is to use the sub-multiplicativity of the Lipschitz constant for the composition operation, which yields the estimate

$$\mathrm{Lip}(\mathbf{f}_{\boldsymbol{\theta}}) \leq \mathrm{Lip}(\boldsymbol{A}_L) \prod_{\ell=1}^{L-1} \mathrm{Lip}(\boldsymbol{\sigma}_\ell)\,\mathrm{Lip}(\boldsymbol{A}_\ell). \tag{5.37}$$

Consequently, one can obtain a bound for $\mathrm{Lip}(\mathbf{f}_{\boldsymbol{\theta}})$ by controlling the Lipschitz constant of each linear layer and of each activation function.

Lipschitz-Constrained Linear Layers

It is known that the Lipschitz constant of the linear layer $\boldsymbol{A}_\ell$ is equal to the largest singular value of its weight matrix. In our experiments, we constrain the weight matrices $\mathbf{W}_\ell$ in two ways.

- **Spectral Normalization:** This method rescales each linear layer by dividing its weight matrix by its largest singular value. The latter is estimated via power iterations. This method was introduced for fully connected networks in [211] and later generalized for convolutional layers in [57].
- **Orthonormalization:** Here, the weight matrices $\mathbf{W}_\ell$ are forced to be orthonormal, so that $\mathbf{W}_\ell^T \mathbf{W}_\ell$ is the identity matrix. Unlike spectral normalization, which only constrains the largest singular value, this method forces all the singular values to be one. Various implementations of orthonormalization have been proposed to handle both fully connected [212] and convolutional layers [217, 218].

Lipschitz-Constrained Activation Functions

Here, we shortly introduce all activation functions that we compare against LLS.

- **ReLU:** The ReLU activation function is component-wise and 1-Lipschitz. It is given by $\mathrm{ReLU}(\mathbf{x}) = (\max(0, x_n))_{n=1}^N$.
- **Absolute Value:** The absolute value (AV) activation function is component-wise, 1-Lipschitz, and GNP. It is given by $\mathrm{AV}(\mathbf{x}) = (|x_n|)_{n=1}^N$.
- **Parametric ReLU:** The parametric ReLU (PReLU) activation function [184] is component-wise. 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, an easy way to make it 1-Lipschitz is to clip the parameters $(a_n)_{n=1}^N$ in $[-1, 1]$.
- **GroupSort:** This activation function [212] separates the pre-activations into groups of size k and then sorts each group in ascending order. Hence, it is locally a permutation and is therefore GNP and 1-Lipschitz. If the group size is 2, this activation function is called MaxMin. 1-Lip MaxMin and GS neural networks are universal approximators for 1-Lipschitz functions in a specific setting where the first weight matrix satisfies $\|\mathbf{W}_1\|_{2,\infty} \leq 1$ and all other weight matrices satisfy $\|\mathbf{W}_l\|_\infty \leq 1$ [212, Theorem 3].
- **Householder:** The householder (HH) activation function [219] separates the pre-activations into groups of size 2, and for any $\mathbf{x} \in \mathbb{R}^2$, computes

$$\mathrm{HH}_{\mathbf{v}}(\mathbf{x}) = \begin{cases} \mathbf{x}, & \mathbf{v}^T\mathbf{x} > 0 \\ \left(\mathbf{I} - 2\mathbf{v}\mathbf{v}^T\right)\mathbf{x}, & \mathbf{v}^T\mathbf{x} \leq 0, \end{cases} \tag{5.38}$$

where $\mathbf{v} \in \mathbb{R}^2$ is a learnable parameter with $\|\mathbf{v}\|_2 = 1$. The HH activation function is always 1-Lipschitz and GNP.

For these choices, Proposition 5.3 holds. The proof is given in Appendix 5.4.5.

Proposition 5.3. *On any compact set $D \subset \mathbb{R}^{N_0}$, 1-Lip neural networks with AV, PReLU, GS, or HH activation functions can represent the same set of functions.*

By contrast to Proposition 5.3, 1-Lip ReLU networks are less expressive and can only represent a subset of these functions.

5.2.4 1-Lipschitz Deep Spline Neural Networks

As a first step towards Lipschitz-constrained deep spline neural networks, [200] added a term in the training loss that penalizes a loose bound of the Lipschitz constant of the LLS activation functions. This approach, however, does not offer a strict control of the overall Lipschitz constant of the network. Here, we instead present a method to explicitly control the Lipschitz constant of each LLS activation function.

As described in Section 5.4.2, we represent an LLS activation function σ in a B-spline basis. It is fully described by the vector of its B-spline coefficients $\mathbf{c} \in \mathbb{R}^K$ and the stepsize T. In practice, we choose a high number K and a small stepsize T. We then ensure that a simple activation function is learned by using $\mathrm{TV}^{(2)}$ regularization given by $\mathrm{TV}^{(2)}(\sigma) = \|\mathbf{Lc}\|_1$, where $\mathbf{L}$ is the second-order finite-difference matrix defined in (5.97). The Lipschitz constant of σ is given by $\mathrm{Lip}(\sigma) = \frac{1}{T}\|\mathbf{Dc}\|_\infty$, where

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & -1 & 1 \end{bmatrix}. \tag{5.39}$$

Overall, we aim to impose strict bounds on the first-order finite differences of the coefficients $\mathbf{c}$, and we seek to sparsify their second-order finite differences.

To ensure that every activation function σ is 1-Lipschitz, the absolute difference between any two consecutive coefficients must be at most T. Hence, the corresponding set of feasible coefficients is given by $\{\mathbf{c} \in \mathbb{R}^K : \|\mathbf{Dc}\|_\infty \leq T\}$. A first attempt at a minimization over this set was made in our earlier work [157]. There, we used a method that divides each activation function by its maximum slope after each training step. In Section 5.2.4, we present an alternative projection scheme that is better suited to optimization and yields a much better performance in practice, while being just as fast. Additionally, we introduce a scaling parameter for each activation function, which facilitates the training and increases the performance of the network even further at a negligible computational cost.

Constrained Coefficients

The textbook approach to maintain the 1-Lipschitz property throughout an iterative minimization scheme would be to determine the least-squares projection onto $\{\mathbf{c} \in \mathbb{R}^K : \|\mathbf{D}\mathbf{c}\|_\infty \leq T\}$ at each iteration. This operation would preserve the mean of $\mathbf{c}$, as shown in Appendix 5.4.6. Unfortunately, its computation is very expensive as it requires to solve a quadratic program after each training step and for each activation function. As substitute, we introduce a simpler projection $\mathrm{P}_{\mathrm{Lip}}$ that also preserves the mean while being much faster to compute. In brief, $\mathrm{P}_{\mathrm{Lip}}$ computes the finite-differences, clips them, sums them and adds a constant to the preservation of the mean.

Let us denote the Moore–Penrose pseudoinverse of $\mathbf{D}$ by $\mathbf{D}^\dagger$ and the vector of ones by $\mathbf{1} \in \mathbb{R}^K$. Further, we define the component-wise operation

$$\mathrm{Clip}_{[T_1,T_2]}(x) = \begin{cases} T_1, & x < T_1 \\ x, & x \in [T_1, T_2] \\ T_2, & x > T_2. \end{cases} \tag{5.40}$$

Proposition 5.4. *The operation* $\mathrm{P}_{\mathrm{Lip}}$ *defined as*

$$\mathrm{P}_{\mathrm{Lip}}(\mathbf{c}) = \mathbf{D}^\dagger \,\mathrm{Clip}_{[-T,T]}(\mathbf{D}\mathbf{c}) + \mathbf{1}\frac{1}{K}\sum_{k=1}^{K} c_k \tag{5.41}$$

has the following properties:

1. *it is a projection onto the set* $\{\mathbf{c} \in \mathbb{R}^K : \|\mathbf{D}\mathbf{c}\|_\infty \leq T\}$*;*
2. *it is almost-everywhere differentiable with respect to* $\mathbf{c}$*;*
3. *it preserves the mean of* $\mathbf{c}$*.*

The proof of Proposition 5.4 can be found in Appendix 5.4.6.

In gradient-based optimization, one usually handles domain constraints by projecting the variables back onto the feasible set after each gradient step. However, this turned out to be inefficient for neural networks in our experiments. Instead, we parameterize the LLS activation functions directly with $\mathrm{P}_{\mathrm{Lip}}(\mathbf{c})$, which leads to unconstrained training. This strategy is in line with the popular spectral normalization of [211], where the weight matrices are unconstrained and parameterized using an approximate projection. For our parameterization approach, Property 2 of Proposition 5.4 is very important as it allows us to back-propagate through $\mathrm{P}_{\mathrm{Lip}}$ during the optimization process. To compute $\mathrm{P}_{\mathrm{Lip}}$ efficiently, we calculate $\mathbf{D}^{\dagger}$ in a matrix-free fashion with a cumulative sum. The computational cost of $\mathrm{P}_{\mathrm{Lip}}$ is negligeable 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 γ. Specifically, we propose the new activation function

$$\tilde{\sigma}(x) = \frac{1}{\gamma}\sigma(\gamma x). \tag{5.42}$$

With this scaling, $\tilde{\sigma}$ is nonlinear on $[k_{\min}T/\gamma, k_{\max}T/\gamma]$ and the Lipschitz constant

$$\mathrm{Lip}(\tilde{\sigma}) = \sup_{x_1,x_2\in\mathbb{R}} \frac{\|\frac{1}{\gamma}\sigma(\gamma x_1) - \frac{1}{\gamma}\sigma(\gamma x_2)\|}{\|x_1 - x_2\|} = \sup_{x_1,x_2\in\mathbb{R}} \frac{\frac{1}{\gamma}\|\sigma(\gamma x_1) - \sigma(\gamma x_2)\|}{\frac{1}{\gamma}\|\gamma x_1 - \gamma x_2\|} = \mathrm{Lip}(\sigma) \tag{5.43}$$

is left unchanged. We can see from Proposition 5.11 that the second-order total variation is preserved as well. Basically, γ allows us to decrease the data-fitting term used for training without breaking the constraints or increasing the complexity of the activation functions. Experimentally, we indeed found that the performance of DSNNs improves if we also optimize over γ. 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 γ is initialized as one and updated via standard stochastic gradient-based methods. Throughout our experiments, every LLS activation function has its own scaling parameter γ.

5.2.5 Experimental Results

We evaluate the performance of 1-Lip neural networks on a variety of tasks. In each case, we compare the performance of LLS and the five activation functions discussed in Section 5.2.3. 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 [194] and the default hyperparameters of its PyTorch implementation. The deep spline NNs have three optimizers with different learning rates: one for the weights (with learning rate η), one for the scaling parameters (with learning rate $\eta/4$) and one for parameters of

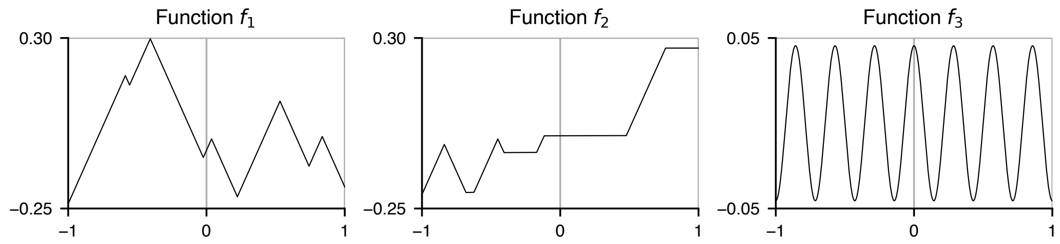


Figure 5.7: Three 1-Lipschitz functions that we attempt to fit with 1-Lip neural networks. All functions have zero mean over the interval $[-1, 1]$.

the activation function (with learning rate $\eta/40$). These ratios remain fixed throughout this section and, hence, only η is going to be stated. Our implementation is available on Github[11].

One-Dimensional Function Fitting

Here, we train 1-Lip neural networks to fit 1-Lipschitz functions $f\colon \mathbb{R} \to \mathbb{R}$ within the model $Y = f(X)$, where X is uniformly distributed on $[-1, 1]$. The task is to fit the three 1-Lipschitz functions depicted in Figure 5.7. The aim of this experiment is twofold. First, we want to probe the impact of the two methods described in Sections 5.2.4 and 5.2.4 on the performance of the DSNNs by comparing the proposed 1-Lip DSNNs (denoted as LLS New) with the ones from our earlier work [157] (denoted as LLS Old) which operate by simple normalization. Second, we want a simple but challenging experiment to compare the various available methods.

Let us briefly comment on the functions in Figure 5.7. For f_1, we have $|\nabla f_1| = 1$ 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 alternates between $|\nabla f_2| = 1$ and $|\nabla f_2| = 0$. It was designed to test the ability of LLS to fit functions with constant regions. Lastly, we benchmark all methods on the highly varying function $f_3(x) = \sin(7\pi x)/7\pi$, which is challenging to fit under Lipschitz constraints.

For each method, we consider two variants: orthonormalization, and spectral normalization of the weights. In both cases, we use the mean squared error (MSE) as 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 without caring about generalization. The hyperparameters were all tuned on the test set. For each activation function, we tuned the width and the depth of the neural network for best performance. ReLU networks have 10 layers and a width of 50; AV, PReLU, and HH networks have 8 layers and a

[11] https://github.com/StanislasDucotterd/Lipschitz_DSNN

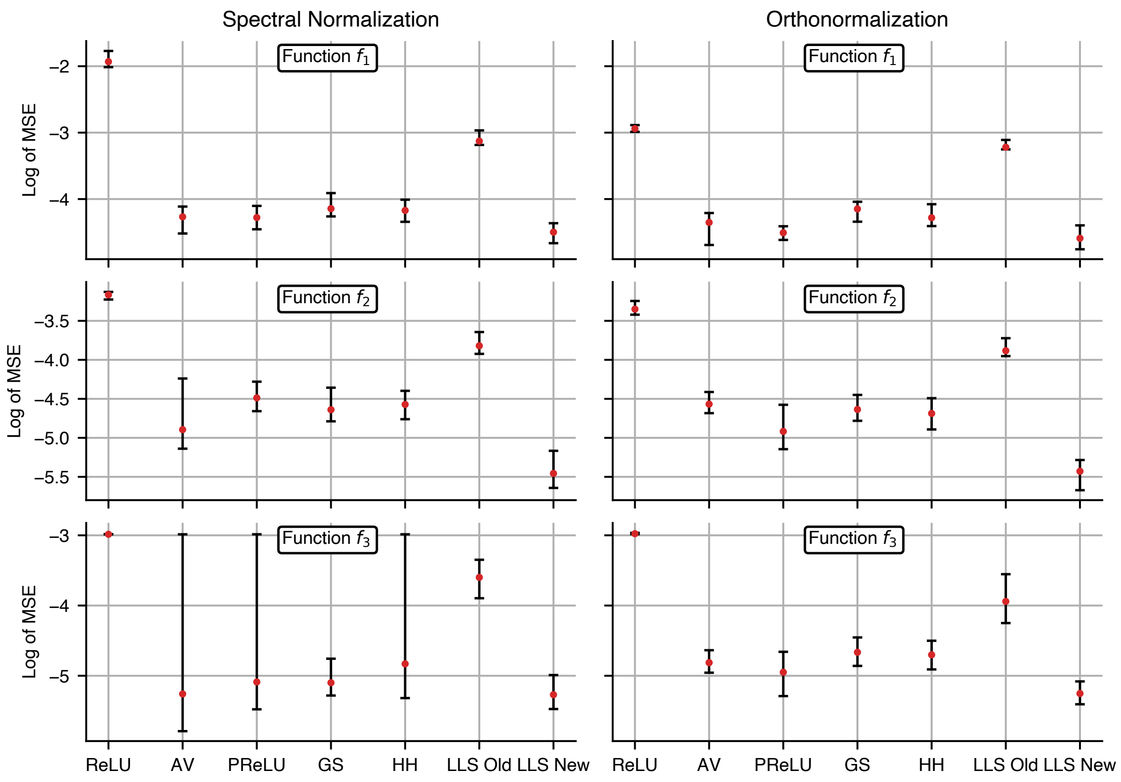


Figure 5.8: Fitting performances for the functions from Figure 5.7. The red markers represent the median performance. The black bars represent the lower and upper quartiles, respectively.

width of 20; GS networks have 7 layers and a width of 20; DSNNs have 4 layers and a width of 10. For 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 and had a range of $[-0.5, 0.5]$, 100 linear regions, and we set $\lambda = 10^{-7}$ for the $\mathrm{TV}^{(2)}$ regularization. The DSNNs used a learning rate of $\eta = 2 \times 10^{-3}$ for every function while the other networks used $\eta = 4 \times 10^{-3}$ for f_1, f_2 and $\eta = 10^{-3}$ for f_3. Every network relied on Kaiming initialization [184] and was trained 25 times with a batch size of 10 for 1000 epochs. We report the median and the two quartiles of the test losses in Figure 5.8.

For the spectral normalization, we observe that AV, PReLU, and HH have a tendency 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 LLS New over LLS Old, which clearly demonstrates the beneficial role of the two modules described in Sections 5.2.4 and 5.2.4. Accordingly, from now on, we drop LLS Old and only retain LLS New.

Table 5.8: Mean and standard deviation of the estimated Wasserstein distance over five trials for several architectures.

Depth	ReLU	AV	PReLU	GS	HH	LLS
3	0.727	**1.190/0.002**	**1.190/0.002**	1.189/0.001	1.165/0.001	**1.190/0.002**
5	0.881/0.001	1.368/0.003	1.371/0.002	1.369/0.002	1.369/0.002	**1.373/0.003**
7	0.960	1.406/0.008	1.437/0.002	1.436/0.001	**1.440/0.003**	1.439/0.001

Estimation of the Wasserstein Distance

The Wasserstein-1 distance is a metric between two probability distributions P_1 and P_2. This metric has been used in [220] to improve the performance of GANs, which were first introduced in [69]. Using the Kantorovich dual formulation [221], we can compute the Wasserstein-1 distance W_1 by solving an optimization problem over the space of 1-Lipschitz functions, leading to

$$W_1(P_1, P_2) = \sup_{\mathrm{Lip}(f)\leq 1} \mathbb{E}_{\mathbf{x}\sim P_1}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim P_2}[f(\mathbf{x})]. \tag{5.44}$$

In our Wasserstein experiment, P_1 is a uniform distribution over a set of real MNIST[12] images and P_2 is the generator distribution of a GAN trained to generate MNIST images. The architecture of this GAN is taken from [222]. It has been shown in [223] that, under reasonable assumptions, any f^* that maximizes (5.44) satisfies $|\nabla f^*| = 1$ almost everywhere. Further, [212] have shown experimentally that, in the context of Wasserstein distance estimation, spectral normalization of the linear layers is outperformed by orthonormalization. Hence, we use the latter parameterization for the Wasserstein experiments. All networks are fully connected with a width of 1024, and various depths. They were trained 5 times each for 2000 epochs with $\eta = 2 \times 10^{-3}$ and orthogonal initialization [224]. For the networks with a depth of 3, GS has group size of 8, and PReLU and LLS were initialized as the absolute value. For a depth of 5 or 7, GS has a group size of 2, and PReLU and LLS were initialized with the identity in half of the activation functions and as the absolute value in the other half. The LLS have a range of $[-0.15, 0.15]$, 20 linear regions, and $\lambda = 10^{-10}$. The spline coefficients only increase the total number of parameters in the neural network by 2%. We train the networks on 54000 images from the MNIST training set and use the 6000 remaining ones as a validation set. The test set contains 10000 MNIST images.

In Table 5.8, we report the estimated Wasserstein distance between the MNIST images of the test set and the ones generated by the GAN. The dimensionality of the problem is such that it is practically unfeasible to compute accurate baseline estimates based on the sampling of both measures and the computation of their true Wasserstein distance.

[12]http://yann.lecun.com/exdb/mnist/

ReLU has an estimate that is significantly lower than the other methods. Most likely, this corresponds to a gross underestimation of the true Wasserstein distance because of its lack of expressivity. We can see that the performances are quite similar between all the other activation functions except for AV and HH with depth 7 and 3, respectively, which are worse than the others.

Image Reconstruction via PnP-FBS

Next, we consider provably convergent PnP-FBS for two ill-posed inverse problems—MRI and CT reconstruction. Recall that a β-averaged denoiser is of the form $\mathrm{D} = \beta\mathrm{N}+(1-\beta)\mathrm{Id}$, where $\beta \in (0, 1)$ and N is nonexpansive. Here, we parametrize N as a 1-Lip neural network and train it as a denoiser. Before we talk about the inverse problem setups, we describe our denoising experiments.

1. Denoising

The state-of-the-art image denoising architectures [154, 225, 226] are not natively 1-Lipschitz. They rely on dedicated modules designed to improve the performance of the denoising network, such as skip connections, downsampling and upsampling layers, batch normalization, and attention modules. These can make it challenging to build provably averaged denoisers, and their effectiveness remains to be demonstrated in a constrained setting. For this reason, we use a simple CNN architecture without batch normalization. This provides excellent performance while relying on a simple architecture that can be directly constrained.

We train N as a 1-Lip denoiser that is composed of 8 orthogonal convolutional layers parameterized with the BCOP framework [217]. For LLS, we take 64 channels; to compensate for the additional spline parameters, we train every other model with 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 [227]. We report the results on the BSD68 test set. All images take values in $[0, 1]$. For our experiment, we add Gaussian noise with $\sigma = 5/255, 10/255, 15/255$. We train all networks 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 50 linear regions, a range of 0.1, and were initialized as the identity. We set $\eta = 4 \times 10^{-5}$ for every noise level and every model. In this experiment, we also investigated the effect of the $\mathrm{TV}^{(2)}$ regularization parameter λ (we used the same parameter for all the layers) on the performance and the number of linear regions in all the activation functions. The performance results are provided in Table 5.9. As expected, ReLU is doing worse than the other activation functions. For each noise level, LLS is outperforming every other activation function.

Table 5.9: PSNR and SSIM values for the Lipschitz denoising experiment in terms of activation functions and noise levels.

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 ($\lambda = 0$)	36.85	0.9540	**32.59**	**0.8978**	30.35	0.8464
LLS ($\lambda = 10^{-6}$)	**36.86**	**0.9546**	32.55	0.8962	**30.38**	**0.8479**
LLS ($\lambda = 10^{-5}$)	**36.86**	0.9543	32.55	0.8960	30.34	0.8455
LLS ($\lambda = 10^{-4}$)	36.82	0.9534	32.57	0.8970	30.36	0.8468
LLS ($\lambda = 10^{-3}$)	36.63	0.9497	32.47	0.8924	30.31	0.8437
LLS ($\lambda = 10^{-2}$)	35.15	0.9142	32.00	0.8782	29.73	0.8156

Table 5.10: Average number of effective linear regions (AELR) for several λ and noise levels. The maximum number of available regions for the LLS is 50.

Noise level	$\sigma = 5/255$	$\sigma = 10/255$	$\sigma = 15/255$
LLS ($\lambda = 0$)	9.24	8.76	8.07
LLS ($\lambda = 10^{-6}$)	1.21	1.24	1.44
LLS ($\lambda = 10^{-5}$)	1.11	1.15	1.24
LLS ($\lambda = 10^{-4}$)	1.07	1.14	1.25
LLS ($\lambda = 10^{-3}$)	1.02	1.06	1.10
LLS ($\lambda = 10^{-2}$)	1.00	1.01	1.02

The number of linear regions for the LLS activation function $\sigma_{\ell,n}$ is equal to $\|\mathbf{L}\mathbf{c}_{\ell,n}\|_0 + 1$. 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 DSNN, we report in Table 5.10 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 become skip connection, 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 $\lambda \in [10^{-6}, 10^{-4}]$, the AELR is between 1.07 and 1.44, which is a large decrease without degradation in the denoising performances. For $\lambda = 10^{-3}$, the LLS are even further sparsified at the cost of a small loss of performance. We observe a significant loss of performance when λ 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 DSNNs with $\lambda \in [10^{-6}, 10^{-3}]$ 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.

2. Biomedical Image Reconstruction

Finally, we deploy the trained 1-Lip denoisers within PnP-FBS to solve MRI and CT reconstruction problems.

(A) MRI The ground-truth images for our MRI experiments are proton-density weighted knee MR images from the fastMRI dataset [228] with fat suppression (PDFS) and without fat suppresion (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. We consider both 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 [229] available in the BART toolbox [230]). For both cases, the subsampling in the Fourier domain is performed with a Cartesian mask that is specified by two parameters: the acceleration $M_{\mathrm{acc}} \in \{2, 4, 8\}$ and the center fraction $M_{\mathrm{cf}} = 0.32/M_{\mathrm{acc}}$. 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 $1/M_{\mathrm{acc}}$. In addition, Gaussian noise with standard deviation $\sigma_{\mathbf{n}} = 2 \times 10^{-3}$ 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.

(B) CT We target the CT experiment proposed in [231]. The data consist of human abdominal CT scans for 10 patients provided by Mayo Clinic for the low-dose CT Grand Challenge [232]. The validation set consists of 6 images taken uniformly from the first patient of the training set from [231]. We use the same test set as [231], more precisely, 128 slices with size (512×512) that correspond to one patient. 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.

For the above-described setups, we run the PnP-FBS algorithm (5.32) with the trained denoising networks (for LLS we use the network corresponding to $\lambda = 10^{-6}$). We tune the noise level σ of each N over $\sigma = 5/255, 10/255, 15/255$. Also, for each σ, we tune $\beta \in (0, 1)$ and the stepsize α given in (5.32) using the coarse-to-fine method given in

Appendix 5.4.7. The hyperparameters for all the methods are tuned on the validation set to maximize the average PSNR.

The reconstruction performances over the test sets are reported in Tables 5.11, 5.12 and 5.20. We observe a significant gap between LLS and the other activation functions for all the setups in terms of PSNR and SSIM. Some example reconstructions are reported in Figures 5.9, 5.10, 5.11, 5.12, 5.13 and 5.14.

5.2.6 Summary

In this work, 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, we have shown that our approach outperforms other Lipschitz-constrained neural networks for a variety of tasks including plug-and-play image reconstruction.

Table 5.11: 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 5.12: 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**

Table 5.13: 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
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**

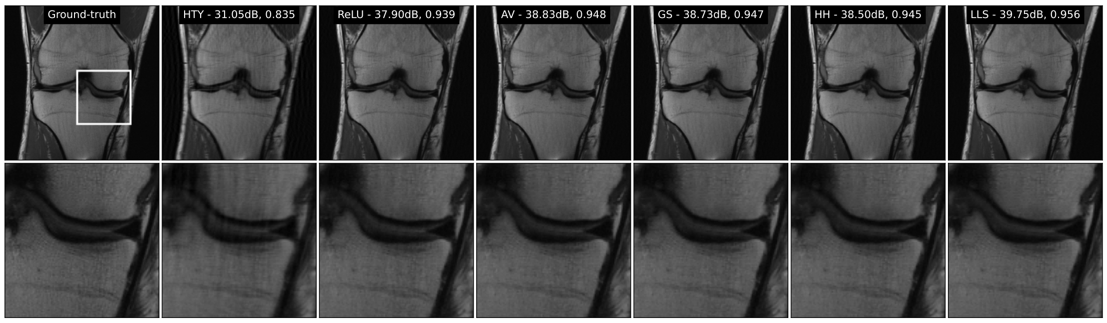


Figure 5.9: Reconstructed images for the 2-fold accelerated single-coil MRI experiment. The reported metrics are PSNR and SSIM.

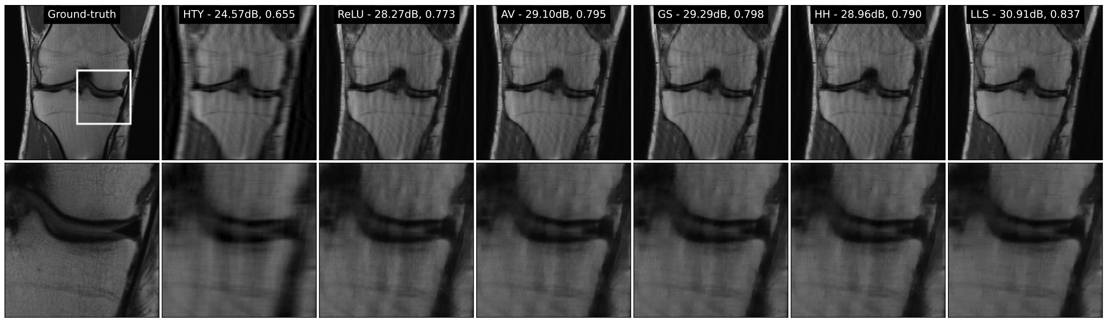


Figure 5.10: Reconstructed images for the 4-fold accelerated single-coil MRI experiment. The reported metrics are PSNR and SSIM.

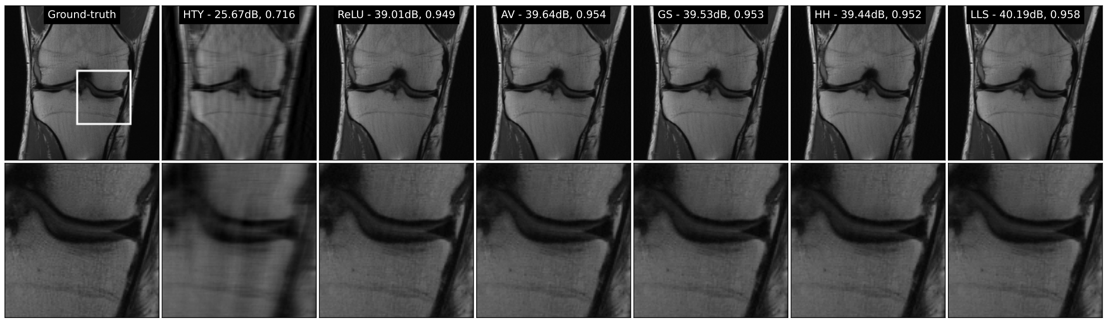


Figure 5.11: Reconstructed images for the 4-fold accelerated multi-coil MRI experiment. The reported metrics are PSNR and SSIM.

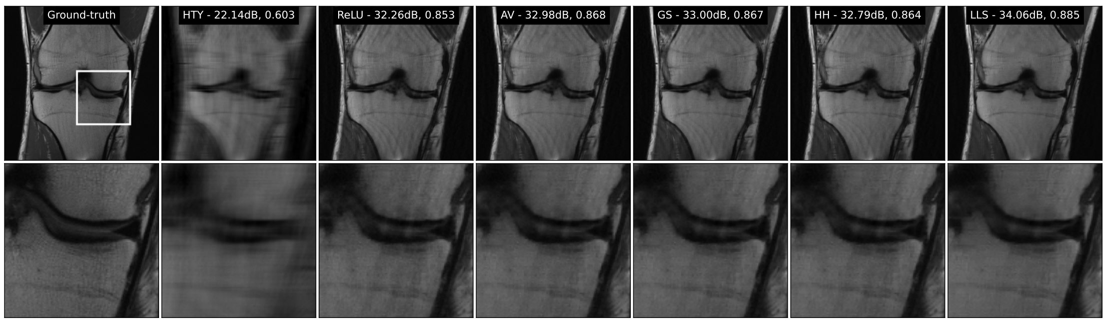


Figure 5.12: Reconstructed images for the 8-fold accelerated multi-coil MRI experiment. The reported metrics are PSNR and SSIM.

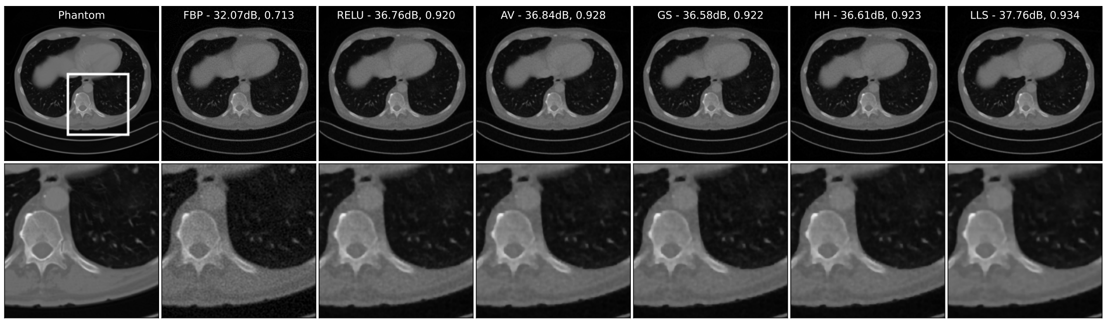


Figure 5.13: Reconstructed images for the CT experiment with $\sigma_{\mathbf{n}} = 0.5$. The reported metrics are PSNR and SSIM.

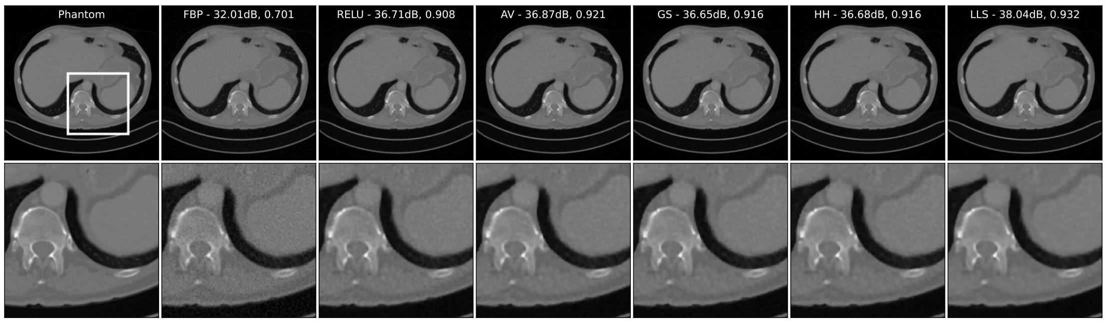


Figure 5.14: Reconstructed images for the CT experiment with $\sigma_{\mathbf{n}} = 0.5$. The reported metrics are PSNR and SSIM.

5.3 A Neural-Network-Based Convex Regularizer

[13]In this section, we show how we can leverage our learnable linear spline module to design explicit neural-network-based convex regularizers.

5.3.1 Introduction

Like in the previous section, we again consider ill-posed linear inverse problems with an AWGN model. Thus, the goal is to reconstruct the image $\mathbf{s} \in \mathbb{R}^d$ from the measurement vector $\mathbf{y} \in \mathbb{R}^m$ given by

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}, \tag{5.45}$$

where $\mathbf{H} \in \mathbb{R}^{m \times d}$ models the physics of the acquisition process and $\mathbf{n} \in \mathbb{R}^m$ accounts for the additive Gaussian noise. The generic MPL estimator for the image can be written as

$$\mathbf{s}^* \in \arg\min_{\mathbf{s} \in \mathbb{R}^d} \frac{1}{2}\|\mathbf{H}\mathbf{s} - \mathbf{y}\|_2^2 + \mathcal{R}(\mathbf{s}), \tag{5.46}$$

where $\mathcal{R} \colon \mathbb{R}^d \to \mathbb{R}$ is a regularizer that incorporates prior information about $\mathbf{s}$ to counteract the ill-posedness of (5.45). Here, we will focus on the learning of the regularization term $\mathcal{R}$ in (5.46). Pioneering work in this direction includes the *fields of experts* [233–235], where $\mathcal{R}$ 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) [236, 237], NETT [238], and the total-deep-variation frameworks [67], or with regularizers for which a proximal operator exists [64–66, 239]. There exists a variety of strategies to learn $\mathcal{R}$, including bilevel optimization [234], unrolling [67, 235], gradient-step denoising [64, 65], and adversarial training [236, 237]. When $\mathcal{R}$ is convex, a global minimizer of (5.46) can be found under mild assumptions. As the relaxation of the convexity constraint usually boosts the performance [234, 240], it is consequently the most popular approach. Unfortunately, one can then expect convergence only to a critical point.

In this work, we prioritize the reliability and interpretability of the method. Thus, we revisit the family of learnable convex-ridge regularizers [181, 233–235, 240]

$$\mathcal{R} \colon \mathbf{s} \mapsto \sum_i \psi_i(\mathbf{w}_i^T \mathbf{s}), \tag{5.47}$$

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 $\mathcal{R}$ is to solve a non-convex bilevel optimization task [241, 242] for a given inverse problem. It was reported in [234] that these learnt regularizers outperform the popular TV regularizer for image reconstruction. As bilevel optimization is computa-

[13]This section is based on our work [159].

tionally quite intensive, it was proposed in [240] to unroll the forward-backward splitting (FBS) algorithm applied to (5.46) with a regularizer of the form (5.47). Accordingly, $\mathcal{R}$ 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[14] to learn some $\mathcal{R}$ of the form (5.47) with free-form convex profiles. We train this $\mathcal{R}$ on a generic denoising task and then plug it into (5.46). This yields a generic reconstruction framework that is applicable to a variety of inverse problems. The main contributions of the present work 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 $\boldsymbol{\nabla}\mathcal{R}$. We prove that this yields the maximal expressivity in the generic setting (5.47).

- **Embedding of the Constraints into the Forward Pass:** The structural constraints on $\boldsymbol{\nabla}\mathcal{R}$ 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 $\boldsymbol{\nabla}\mathcal{R}$, which is required for our training procedure.

- **Ultra-Fast Training:** The regularizer $\mathcal{R}$ is learnt via the training of a multi-gradient-step denoiser. Empirically, we observe that a few gradient steps suffice to learn a best-performing $\mathcal{R}$. 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.

5.3.2 Architecture of the Regularizer

In this section, we introduce the notions required to define the convex-ridge regularizer neural network (CRR-NN).

[14]All experiments can be reproduced with the code published at https://github.com/axgoujon/convex_ridge_regularizers

General Setting

Our goal is to learn a regularizer $\mathcal{R}$ for the variational problem (5.46) that performs well across a variety of ill-posed problems. Similar to the PnP framework, we view the denoising task

$$\mathbf{s}^* = \arg\min_{\mathbf{s}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{s}-\mathbf{y}\|_2^2 + \tau\mathcal{R}(\mathbf{s}) \tag{5.48}$$

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 (5.47) and use its convolutional form. More precisely, the regularity of an image s is measured as

$$\mathcal{R}\colon x \mapsto \sum_{i=1}^{N_C}\sum_{\mathbf{k}\in\mathbb{Z}^2} \psi_i\Big((h_i * s)[\mathbf{k}]\Big), \tag{5.49}$$

where h_i is the impulse response of a 2D convolutional filter, $(h_i * s)[\mathbf{k}]$ is the value of the $\mathbf{k}$-th pixel of the filtered image $h_i * s$, and N_C is the number of channels. In the sequel, we mainly view the (finite-size) image s as the (finite-dimensional) vector $\mathbf{s}\in\mathbb{R}^d$, and since (5.49) is a special case of (5.47), we henceforth use the generic form (5.47) to simplify the notations. We use the notation $\mathcal{R}_{\boldsymbol{\theta}}$ to express the dependence of $\mathcal{R}$ 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. $\psi_i \in C^{1,1}(\mathbb{R})$.

Gradient-Step Neural Network

Given the assumptions on $\mathcal{R}_{\boldsymbol{\theta}}$, the denoised image in (5.48) can be interpreted as the unique fixed point of $\boldsymbol{T}_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}\colon \mathbb{R}^d \to \mathbb{R}^d$ defined by

$$\boldsymbol{T}_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}(\mathbf{s}) = \mathbf{s} - \alpha\Big((\mathbf{s}-\mathbf{y}) + \tau\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}}(\mathbf{s})\Big). \tag{5.50}$$

Iterations of the operator (5.50) implement a gradient descent with stepsize α, which converges if $\alpha \in (0, 2/(1+\tau L_{\boldsymbol{\theta}}))$, where $L_{\boldsymbol{\theta}} = \mathrm{Lip}(\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}})$ is the Lipschitz constant of $\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}}$. In the sequel, we always enforce this constraint on α. The gradient of the generic convex-ridge expression (5.47) is given by

$$\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{W}^T\boldsymbol{\sigma}(\mathbf{W}\mathbf{s}), \tag{5.51}$$

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

$$\boldsymbol{T}_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}(\mathbf{s}) = (1-\alpha)\mathbf{s} + \alpha\Big(\mathbf{y} - \tau\mathbf{W}^T\boldsymbol{\sigma}(\mathbf{W}\mathbf{s})\Big) \tag{5.52}$$

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.

5.3.3 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 5.3.4.

Existence of Minimizers and Stability of the Reconstruction

The convexity of $\mathcal{R}_{\boldsymbol{\theta}}$ is not sufficient to ensure that the solution set in (5.46) 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 5.5). The implications for our implementation are detailed in Section 5.3.4.

Proposition 5.5. *Let $\mathbf{H} \in \mathbb{R}^{m\times d}$ and $\psi_i\colon \mathbb{R} \to \mathbb{R}$, $i = 1,\dots,p$, be convex functions. If $\arg\min_{t\in\mathbb{R}} \psi_i(t) \neq \emptyset$ for all $i = 1,\dots,p$, then*

$$\emptyset \neq \operatorname*{arg\,min}_{\mathbf{s}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{H}\mathbf{s} - \mathbf{y}\|_2^2 + \sum_{i=1}^{p} \psi_i(\mathbf{w}_i^T\mathbf{s}). \tag{5.53}$$

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

- $\Omega_0^i = \{\mathbf{s} \in \mathbb{R}^d : \mathbf{w}_i^T\mathbf{s} \in S_i\}$;
- $\Omega_1^i = \{\mathbf{s} \in \mathbb{R}^d : \mathbf{w}_i^T\mathbf{s} \leq \inf S_i\}$;
- $\Omega_2^i = \{\mathbf{s} \in \mathbb{R}^d : \mathbf{w}_i^T\mathbf{s} \geq \sup S_i\}$.

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 (5.53) 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 $(\mathbf{s}_k)_{k\in\mathbb{N}} \subset P$. 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 $\mathbf{H}\mathbf{s}_k$ 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, $\mathbf{w}_i^T\mathbf{s}_k$ also remains bounded. Therefore, the sequence $(\mathbf{M}\mathbf{s}_k)_{k\in\mathbb{N}}$ is bounded and we can drop to a convergent subsequence with limit $\mathbf{u} \in \mathrm{ran}(\mathbf{M})$. The associated set

$$Q = \{\mathbf{s} \in \mathbb{R}^d \colon \mathbf{M}\mathbf{s} = \mathbf{u}\} = \{\mathbf{M}^\dagger\mathbf{u}\} + \ker(\mathbf{M}) \tag{5.54}$$

is a closed polytope. It holds that

$$\begin{aligned}\mathrm{dist}(\mathbf{s}_k, Q) &= \mathrm{dist}\Big(\mathbf{M}^\dagger\mathbf{M}\mathbf{s}_k + \mathrm{P}_{\ker(\mathbf{M})}(\mathbf{s}_k), Q\Big)\\ &\leq \mathrm{dist}(\mathbf{M}^\dagger\mathbf{M}\mathbf{s}_k, \mathbf{M}^\dagger\mathbf{u}) \to 0 \end{aligned}\tag{5.55}$$

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 $P \cap Q \neq \emptyset$ [243, Theorem 1]. Note that $\psi_i(\mathbf{w}_i^T\cdot)$ is constant on P if $m_i = 0$. Hence, any $\mathbf{s} \in P \cap Q$ is a minimizer of (5.53). □

The proof of Proposition 5.5 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 5.5 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 (5.46) is a key step towards the stability of the reconstruction map in the measurement domain, which is given in Proposition 5.6.

Proposition 5.6. *Let $\mathbf{H} \in \mathbb{R}^{m\times d}$ and $\psi_i \colon \mathbb{R} \to \mathbb{R}$, $i = 1, \dots, p$, 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*

$$\mathbf{s}_q \in \arg\min_{\mathbf{s}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{H}\mathbf{s} - \mathbf{y}_q\|_2^2 + \sum_{i=1}^p \psi_i(\mathbf{w}_i^T\mathbf{s}) \tag{5.56}$$

with $q = 1, 2$ be the corresponding reconstructions. Then,

$$\|\mathbf{H}\mathbf{s}_1 - \mathbf{H}\mathbf{s}_2\|_2 \leq \|\mathbf{y}_1 - \mathbf{y}_2\|_2. \tag{5.57}$$

Proof. Proposition 5.5 guarantees the existence of $\mathbf{s}_q$. Since the objective in (5.53) is smooth, it holds that $\mathbf{H}^T(\mathbf{H}\mathbf{s}_q - \mathbf{y}_q) + \boldsymbol{\nabla}\mathcal{R}(\mathbf{s}_q) = \mathbf{0}$. From this, we infer that

$$\mathbf{H}^T\mathbf{H}(\mathbf{s}_1 - \mathbf{s}_2) + (\boldsymbol{\nabla}\mathcal{R}(\mathbf{s}_1) - \boldsymbol{\nabla}\mathcal{R}(\mathbf{s}_2)) = \mathbf{H}^T(\mathbf{y}_1 - \mathbf{y}_2). \tag{5.58}$$

Taking the inner product with $(\mathbf{s}_1 - \mathbf{s}_2)$ on both sides gives

$$\begin{aligned}&\|\mathbf{H}\mathbf{s}_1 - \mathbf{H}\mathbf{s}_2\|_2^2 + (\mathbf{s}_1 - \mathbf{s}_2)^T(\boldsymbol{\nabla}\mathcal{R}(\mathbf{s}_1) - \boldsymbol{\nabla}\mathcal{R}(\mathbf{s}_2))\\ =&(\mathbf{H}(\mathbf{s}_1 - \mathbf{s}_2))^T(\mathbf{y}_1 - \mathbf{y}_2).\end{aligned}\tag{5.59}$$

To conclude, we use the fact that the gradient of a convex map is monotone, i.e. $(\mathbf{s}_1 -$

$\mathbf{s}_2)^T(\boldsymbol{\nabla}\mathcal{R}(\mathbf{s}_1) - \boldsymbol{\nabla}\mathcal{R}(\mathbf{s}_2)) \geq 0$, and apply the Cauchy-Schwarz inequality to estimate

$$(\mathbf{H}(\mathbf{s}_1 - \mathbf{s}_2))^T(\mathbf{y}_1 - \mathbf{y}_2) \leq \|\mathbf{H}\mathbf{s}_1 - \mathbf{H}\mathbf{s}_2\|\|\mathbf{y}_1 - \mathbf{y}_2\|. \tag{5.60}$$

□

Expressivity of Profile Functions

The gradient-step NN $\boldsymbol{T}_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}$ introduced in (5.52) 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 $\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}}$.

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

$$\mathcal{E}(\mathbb{R}^d) = \left\{\mathbf{W}^T\boldsymbol{\sigma}(\mathbf{W}\cdot) : \mathbf{W} \in \mathbb{R}^{p\times d}, \sigma_i \in C^{0,1}_{\uparrow}(\mathbb{R})\right\} \tag{5.61}$$

and, further, for any $\Omega \subset \mathbb{R}^d$,

$$\mathcal{E}(\Omega) = \left\{\boldsymbol{f}|_{\Omega} : \boldsymbol{f} \in \mathcal{E}(\mathbb{R}^d)\right\}. \tag{5.62}$$

In the following, we set $\|\boldsymbol{f}\|_{C(\Omega)} := \sup_{\mathbf{s}\in\Omega}\|\boldsymbol{f}(\mathbf{s})\|$ and $\|\boldsymbol{f}\|_{C^1(\Omega)} := \sup_{\mathbf{s}\in\Omega}\|\boldsymbol{f}(\mathbf{s})\| + \sup_{\mathbf{s}\in\Omega}\|\boldsymbol{J}_{\boldsymbol{f}}(\mathbf{s})\|$.

The popular ReLU activation function is Lipschitz-continuous and increasing. Unfortunately, it comes with limited expressivity, as shown in Proposition 5.7.

Proposition 5.7. *Let $\Omega \subset \mathbb{R}^d$ be compact with a nonempty interior. Then, the set*

$$\left\{\mathbf{W}^T\mathrm{ReLU}(\mathbf{W}\cdot - \mathbf{b}) \colon \mathbf{W} \in \mathbb{R}^{p\times d}, \mathbf{b} \in \mathbb{R}^p\right\} \tag{5.63}$$

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 (5.63) is dense in $\mathcal{E}(\Omega)$, then the set

$$\begin{aligned}&\left\{(\mathbf{W}\mathbf{v})^T\mathrm{ReLU}(\mathbf{W}\mathbf{v}\cdot - \mathbf{b}) \colon \mathbf{W} \in \mathbb{R}^{p\times d}, \mathbf{b} \in \mathbb{R}^p\right\}\\ =&\left\{\sum_{i=1}^{p} w_i\mathrm{ReLU}(w_i\cdot - b_i) \colon w_i, b_i \in \mathbb{R}\right\}\end{aligned} \tag{5.64}$$

is dense in $\mathcal{E}((a-\delta, a+\delta))$. Note that all functions f in (5.63) can be rewritten in the form

$$f(x) = \sum_{i=1}^{p_1}\mathrm{ReLU}(w_i x - b_i) + \sum_{i=1}^{p_2}(-\mathrm{ReLU}(-\tilde{w}_i x - \tilde{b}_i)), \tag{5.65}$$

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

$$g\colon t \mapsto \mathrm{ReLU}(t - a + \delta/2) - \mathrm{ReLU}(t - a - \delta/2), \tag{5.66}$$

the density implies that there exists f of the form (5.65) 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

$$\begin{aligned}
&f(a+\delta) - f(a+\delta/2) \\
\geq &\sum_{i=1}^{p_1} \mathrm{ReLU}\Big(w_i(a+\delta) - b_i\Big) - \mathrm{ReLU}\Big(w_i(a+\delta/2) - b_i\Big) \\
\geq &\sum_{\{i : b_i \leq w_i(a+\delta/2)\}} w_i(a + \delta - a - \delta/2) \\
= &\sum_{\{i : b_i \leq w_i(a+\delta/2)\}} w_i\delta/2.
\end{aligned} \tag{5.67}$$

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

$$\begin{aligned}
\frac{7}{8}\delta &= g(a+\delta/2) - g(a-\delta/2) - \frac{1}{8}\delta \\
&\leq f(a+\delta/2) - f(a-\delta/2) \\
&\leq \sum_{\{i:b_i \leq w_i(a+\delta/2)\}} \delta w_i + \sum_{\{i:\tilde{b}_i \geq \tilde{w}_i(\delta/2-a)\}} \delta \tilde{w}_i \leq \frac{\delta}{2},
\end{aligned} \tag{5.68}$$

which yields a contradiction. Hence, the set (5.63) cannot be dense in $\mathcal{E}(\Omega)$. □

Remark 5.1. *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 representational 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 5.8, the set $\mathcal{E}(\Omega)$ can be approximated using increasing linear-spline activation functions.

Proposition 5.8. *Let $\Omega \subset \mathbb{R}^d$ be compact and $m \geq 2$. Then, the set*

$$\Big\{\mathbf{W}^T \boldsymbol{\sigma}(\mathbf{W}\cdot)\colon \mathbf{W} \in \mathbb{R}^{p\times d}, \sigma_i \in \mathcal{LS}^m_{\uparrow}(\mathbb{R})\Big\} \tag{5.69}$$

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 $f \in C_{\uparrow}^{0,1}([0,1])$, and φ_n be the linear-spline interpolator of f at locations $0, 1/2^n, \dots, (1 - 1/2^n), 1$. Since f is increasing and φ_n is piecewise linear, φ_n is also increasing. Further, we get that

$$\|f - \varphi_n\|_{C([0,1])} \leq \max_{k \in \{1,\dots,2^n\}} f(k/2^n) - f((k-1)/2^n). \tag{5.70}$$

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

$$\varphi_n(x) = f(0) + \sum_{k=1}^{2^n} a_{k,n} g\Big(2^n \cdot -(k-1)\Big), \tag{5.71}$$

where $a_{k,n} = (f(k/2^n) - f((k-1)/2^n))$ and g is given by

$$g(x) = \begin{cases} 0, & x \leq 0 \\ x, & 0 < x \leq 1 \\ 1, & \text{otherwise.} \end{cases} \tag{5.72}$$

Finally, (5.71) 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 $\mathbf{\Phi} \colon \mathbb{R}^d \to \mathbb{R}^d$ be given by $\mathbf{s} \mapsto \mathbf{W}^T \sigma(\mathbf{W}\mathbf{s})$ with components $\sigma_i \in \mathcal{C}_{\uparrow}^0(\mathbb{R})$. Let $S_i = \{\mathbf{w}_i^T \mathbf{s} \colon \mathbf{s} \in \Omega\}$, 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 $\varphi_{n,i} \in \mathcal{LS}_{\uparrow}^2(\mathbb{R})$ 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

$$\lim_{n \to \infty} \max_{\mathbf{s} \in \{\mathbf{y} \in \mathbb{R}^d : y_i \in S_i\}} \Big\| \mathbf{U}_n^T \boldsymbol{\varphi}_n(\mathbf{U}_n \mathbf{s}) - \boldsymbol{\sigma}(\mathbf{s}) \Big\|_2 = 0. \tag{5.73}$$

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 $\mathbf{\Phi}$ in $C(\Omega)$. This concludes the proof. □

In the end, Propositions 5.7 and 5.8 imply that using linear-spline activation functions instead of the ReLU for the σ_i enables us to approximate more convex regularizers $\mathcal{R}_{\boldsymbol{\theta}}$.

Corollary 5.1. *Let $\Omega \subset \mathbb{R}^d$ be convex and compact with a nonempty interior. Then, the* *regularizers of the form* (5.47) *with Jacobians of the form* (5.69) *are dense in*

$$\left\{\sum_{i=1}^{p} \psi_i(\mathbf{w}_i^T \mathbf{s}) : \psi_i \in C^{1,1}(\mathbb{R}) \ \textit{convex}, \mathbf{w}_i \in \mathbb{R}^d\right\} \tag{5.74}$$

with respect to $\|\cdot\|_{C^1(\Omega)}$. *The density does not hold if we only consider regularizers with Jacobians of the form* (5.63).

Proof. Let $\mathcal{R}$ be in (5.74). Consequently, its Jacobian is in $\mathcal{E}(\Omega)$. Due to Proposition 5.7, the regularizers with Jacobians of the form (5.63) cannot be dense with respect to $\|\cdot\|_{C^1(\Omega)}$. Meanwhile, by Proposition 5.8, we can choose $\mathbf{s}_0 \in \Omega$ and corresponding regularizers $\mathcal{R}_n$ of the form (5.47) with $\boldsymbol{J}_{\mathcal{R}_n} \in$ (5.69), $\|\boldsymbol{J}_{\mathcal{R}_n} - \boldsymbol{J}_{\mathcal{R}}\|_{C(\Omega)} \to 0$ as $n \to \infty$, and $\mathcal{R}_n(\mathbf{s}_0) = \mathcal{R}(\mathbf{s}_0)$. Now, the mean-value theorem readily implies that $\|\mathcal{R}_n - \mathcal{R}\|_{C^1(\Omega)} \to 0$ as $n \to \infty$. $\square$

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.

5.3.4 Implementation

Training a Multi-Gradient-Step Denoiser

Let $\{\mathbf{s}^m\}_{m=1}^M$ be a set of clean images and let $\{\mathbf{y}^m\}_{m=1}^M = \{\mathbf{s}^m + \mathbf{n}^m\}_{m=1}^M$ 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 $\mathcal{R}_{\boldsymbol{\theta}}$ based on (5.48) is to solve

$$\boldsymbol{\theta}_t^*, \tau_t^* \in \arg\min_{\boldsymbol{\theta},\tau} \sum_{m=1}^{M} \mathcal{L}\Big(\boldsymbol{T}^t_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}(\mathbf{y}^m), \mathbf{s}^m\Big) \tag{5.75}$$

for the limiting case $t = \infty$ and an admissible stepsize α. Here, $\boldsymbol{T}^t_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}$ denotes the t-fold composition of the gradient-step NN given in (5.52). In principle, one can optimize the training problem (5.75) with $t = \infty$. This forms a bilevel optimization problem that can be handled with implicit differentiation techniques [234, 244–246]. However, it turns out that it is unnecessary to fully compute the fixed-point $\boldsymbol{T}^\infty_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}(\mathbf{y}^m)$ to learn $\mathcal{R}_{\boldsymbol{\theta}}$ in our constrained setting. Instead, we approximate $\boldsymbol{T}^\infty_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}(\mathbf{y}^m)$ in a finite number of steps. This specifies the *t-step denoiser* NN $\boldsymbol{T}^t_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}$, which is trained such that

$$\boldsymbol{T}^t_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}(\mathbf{y}^m) \simeq \mathbf{s}^m \tag{5.76}$$

for $m = 1, \dots, M$. This corresponds to a partial minimization of (5.48) with initial guess $\mathbf{y}^m$ or, equivalently, as the unfolding of the gradient-descent algorithm for t iterations with shared parameters across iterations [47, 247]. 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 $\mathcal{R}_{\boldsymbol{\theta}}$ into (5.48), 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 τ and the addition of a scaling parameter, as described in Section 5.3.4. Our numerical experiments in Section 5.3.6 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 [240], but there are some fundamental differences. While the model used in [240] 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 [240] observed that an increase in t deters the denoising performances, which is not the case for our architecture. More differences are outlined in Section 5.3.4.

Implementation of the Constraints

Our learning of the t-step denoiser is constrained as follows.

1. The activation functions σ_i must be increasing (convexity constraint on ψ_i).
2. The activation functions σ_i must take the value 0 somewhere (existence constraint).
3. The stepsize in (5.52) should satisfy $\alpha \in (0, 2/(1+\tau L_{\boldsymbol{\theta}}))$ (convergent gradient-descent).

Since the methods to enforce these constraints can have a major impact on the final performance, they must be designed carefully.

1. Monotonic Splines

Here, we address Constraints (i) and (ii) simultaneously. Similar to our previous contributions in Sections 5.1 and 5.2, we use learnable linear splines $\sigma_{\mathbf{c}^i} : \mathbb{R} \to \mathbb{R}$ with $(M+1)$ uniform knots $\nu_m = (m - M/2)\Delta$, $m = 0, \ldots, M$, where Δ is the spacing of the knots. For simplicity, we assume that M is even. The learnable parameter $\mathbf{c}^i = (c^i_m)_{m=0}^M \in \mathbb{R}^{M+1}$ defines the value $\sigma_{\mathbf{c}^i}(\nu_m) = c^i_m$ of $\sigma_{\mathbf{c}^i}$ at the knots. To fully characterize $\sigma_{\mathbf{c}^i}$, we extend it by the constant value $\mathbf{c}^i_0$ on $(-\infty, \nu_0]$ and $\mathbf{c}^i_M$ on $[\nu_M, +\infty)$. This choice results in a linear extension for the corresponding indefinite integrals that appear for the regularizer $\mathcal{R}_{\boldsymbol{\theta}}$ in (5.48). Further details on the implementation of learnable linear splines can be found in Appendix 5.4.2.

Let $\mathbf{D} \in \mathbb{R}^{M\times(M+1)}$ be the one-dimensional finite-difference matrix with $(\mathbf{D}\mathbf{c}^i)_m = c^i_{m+1} - c^i_m$ for $m = 0, \dots, (M-1)$. As $\sigma_{\mathbf{c}^i}$ is piecewise-linear, it holds that

$$\sigma_{\mathbf{c}^i} \text{ is increasing} \Leftrightarrow \mathbf{D}\mathbf{c}^i \geq 0. \tag{5.77}$$

In order to optimize over $\{\sigma_{\mathbf{c}} \colon \mathbf{D}\mathbf{c} \geq 0\}$, we reparameterize the linear splines as $\sigma_{\mathrm{P}_{\mathrm{cvx}}(\mathbf{c}^i)}$, where

$$\mathrm{P}_{\mathrm{cvx}}(\cdot) = \mathbf{C}\mathbf{D}^{\dagger}\mathrm{ReLU}(\mathbf{D}\,\cdot) \tag{5.78}$$

is a nonlinear projection operator onto the feasible set. There, $\mathbf{D}^{\dagger}$ denotes the Moore-Penrose inverse of $\mathbf{D}$ and $\mathbf{C} = (\mathbf{Id}_{M+1} - \mathbf{1}_{M+1}\mathbf{e}^T_{M/2+1})$ 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 5.5 guarantees the existence of a solution for Problem (5.46).

The proposed parameterization $\sigma_{\mathrm{P}_{\mathrm{cvx}}(\mathbf{c}_i)}$ of the splines has the advantage to use unconstrained trainable parameters $\mathbf{c}_i$. The gradient of the objective in (5.75) with respect to $\mathbf{c}_i$ directly takes into account the constraint via $\mathrm{P}_{\mathrm{cvx}}$. This approach differs significantly from the more standard projected gradient descent—as done in [240] to learn convex profiles—where the $\mathbf{c}_i$ 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 (5.75). For an efficient forward and backward pass with auto-differentiation, $\mathrm{P}_{\mathrm{cvx}}$ is implemented with the `cumsum` function instead of an explicit construction of the matrix $\mathbf{D}^{\dagger}$, and the computational overhead is very small.

2. 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 $\|\mathbf{L}\mathrm{P}_{\mathrm{cvx}}(\mathbf{c}_i)\|_1$ of each spline $\sigma_{\mathrm{P}_{\mathrm{cvx}}(\mathbf{c}_i)}$, where $\mathbf{L} \in \mathbb{R}^{(M-1)\times(M+1)}$ is the second-order finite-difference matrix. The final training loss then reads

$$\sum_{m=1}^{M} \mathcal{L}\Big(\boldsymbol{T}^t_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}(\mathbf{y}^m), \mathbf{s}^m\Big) + \lambda \sum_{i=1}^{p} \|\mathbf{L}\mathrm{P}_{\mathrm{cvx}}(\mathbf{c}_i)\|_1, \tag{5.79}$$

where $\lambda \in \mathbb{R}^+$ allows one to tune the strength of the regularization.

3. Convergent Gradient Steps

Constraint (iii) guarantees that the t-fold composition of the gradient-step NN $\boldsymbol{T}^t_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}$ computes the actual minimizer of (5.48) 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 $\mathrm{Lip}(\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}})$. The estimate that we provide in Proposition 5.9 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 5.9. *Let $L_{\boldsymbol{\theta}}$ denote the Lipschitz constant of $\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{W}^T\boldsymbol{\sigma}(\mathbf{W}\mathbf{s})$ with $\mathbf{W} \in \mathbb{R}^{p\times d}$ and $\sigma_i \in \mathcal{C}^{0,1}_{\uparrow}(\mathbb{R})$. With the notation $\boldsymbol{\Sigma}_\infty = \mathbf{diag}(\|\sigma_1'\|_\infty, \dots, \|\sigma_p'\|_\infty)$ it holds that*

$$L_{\boldsymbol{\theta}} \leq \|\mathbf{W}^T\boldsymbol{\Sigma}_\infty\mathbf{W}\| = \|\sqrt{\boldsymbol{\Sigma}_\infty}\mathbf{W}\|^2, \tag{5.80}$$

which is tighter than the naive bound

$$L_{\boldsymbol{\theta}} \leq L_{\boldsymbol{\sigma}}\|\mathbf{W}\|^2. \tag{5.81}$$

Proof. The bound (5.81) is a standard result for compositional models. Next, we note that the Hessian of $\mathcal{R}_{\boldsymbol{\theta}}$ reads

$$\mathbf{H}_{\mathcal{R}_{\boldsymbol{\theta}}}(\mathbf{s}) = \mathbf{W}^T\boldsymbol{\Sigma}(\mathbf{W}\mathbf{s})\mathbf{W}, \tag{5.82}$$

where $\boldsymbol{\Sigma}(\mathbf{z}) = \mathbf{diag}(\sigma_1'(z_1), \dots, \sigma_p'(z_p))$. Further, it holds that $L_{\boldsymbol{\theta}} \leq \sup_{\mathbf{s}\in\mathbb{R}^d}\|\mathbf{H}_{\mathcal{R}_{\boldsymbol{\theta}}}(\mathbf{s})\|$. Since the functions σ_i are increasing, we have for every $\mathbf{s} \in \mathbb{R}^p$ that $\boldsymbol{\Sigma}_\infty - \boldsymbol{\Sigma}(\mathbf{W}\mathbf{s}) \succeq 0$ and, consequently,

$$\mathbf{W}^T\Big(\boldsymbol{\Sigma}_\infty - \boldsymbol{\Sigma}(\mathbf{W}\mathbf{s})\Big)\mathbf{W} \succeq 0. \tag{5.83}$$

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 $\mathbf{W}^T\boldsymbol{\Sigma}(\mathbf{W}\mathbf{s})\mathbf{W}$. □

The bounds (5.80) and (5.81)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 5.3.6, we found that the improved bound (5.80) was on average 3.2 times smaller than (5.81). As (5.80) 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 $\mathbf{W}^T\boldsymbol{\Sigma}_\infty\mathbf{W}$ 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 $L_{\boldsymbol{\theta}} \simeq \|\mathbf{W}^T\boldsymbol{\Sigma}_\infty\mathbf{W}\mathbf{u}\|$ in our model and hence embed it in the forward pass. This approach is inspired by the spectral-normalization technique proposed in [211], 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 [248]. In practice, the estimate $\mathbf{u}$ is stored so that it can be used as a warm start for the next computation of $L_{\boldsymbol{\theta}}$.

From Gradients to Potentials

To recover the regularizer $\mathcal{R}$ from its gradient $\boldsymbol{\nabla}\mathcal{R}$, one has to determine the profiles ψ_i, which satisfy $\psi_i' = \sigma_{\mathrm{P}_{\mathrm{cvx}}(\mathbf{c}^i)}$. 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 [249], more precisely as

$$\psi_i = \sum_{k\in\mathbb{Z}} d_k^i \beta_+^2 \left(\frac{\cdot - k}{\Delta}\right). \tag{5.84}$$

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 [249] for details. Hence, we obtain that $d_k^i - d_{k-1}^i = (\mathrm{P}_{\mathrm{cvx}}(\mathbf{c}^i))_k$, which defines $(d_k^i)_{k\in\mathbb{Z}}$ up to a constant. This constant can be set arbitrarily as it does not affect $\boldsymbol{\nabla}\mathcal{R}$. Due to the finite support of β_+^2, one can efficiently evaluate ψ_i and then $\mathcal{R}$.

Boosting the Universality of the Regularizer

The learnt $\mathcal{R}_{\boldsymbol{\theta}}$ depends on the training task (denoising) and on the noise level. To solve a generic inverse problem, in addition to the regularization strength τ, we propose to incorporate a tunable scaling parameter $\mu \in \mathbb{R}^+$ and to compute

$$\arg\min_{\mathbf{s}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{H}\mathbf{s} - \mathbf{y}\|_2^2 + \tau/\mu\mathcal{R}_{\boldsymbol{\theta}}(\mu\mathbf{s}). \tag{5.85}$$

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 [250]. During the training of t-step denoisers, we also learn a scaling parameter μ by letting the gradient step NN (5.50) become

$$\boldsymbol{T}_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\mu,\alpha}(\mathbf{s}) = \mathbf{s} - \alpha\Big((\mathbf{s} - \mathbf{y}) + \tau\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}}(\mu\mathbf{s})\Big), \tag{5.86}$$

with now $\alpha < 2/(1 + \tau\mu\mathrm{Lip}(\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}}))$.

Reconstruction Algorithm

The objective in (5.85) 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 3) 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 (AdGD) [251] and its proximal extension [252]; both benefit from a stepsize based on an estimate of the local Lipschitz constant of $\boldsymbol{\nabla}\mathcal{R}$ instead of a more conservative global one.

Algorithm 3 FISTA [23] to solve (5.85)

```
**Input:** $\mathbf{s}_0 \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^m$, $\tau \geq 0$, $\mu > 0$
Set $k = 0$, $\mathbf{z}_0 = \mathbf{s}_0$, $\alpha = 1/(\mu\tau\mathrm{Lip}(\boldsymbol{\nabla}\mathcal{R}) + \|\mathbf{H}\|^2)$, $t_0 = 1$
**while** tolerance not reached **do**
    $\mathbf{s}_{k+1} = (\mathbf{z}_k - \alpha(\mathbf{H}^T(\mathbf{H}\mathbf{z}_k - \mathbf{y}) + \tau\boldsymbol{\nabla}\mathcal{R}(\mu\mathbf{z}_k)))_+$
    $t_{k+1} = (1 + \sqrt{4t_k^2 + 1})/2$
    $\mathbf{z}_{k+1} = \mathbf{s}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{s}_{k+1} - \mathbf{s}_k)$
    $k \leftarrow k + 1$
**end while**
**Output:** $\mathbf{s}_k$
```

Table 5.14: Properties of different regularization frameworks.

	Explicit cost	Provably convergent	Universal	Shallow	Smooth reg.
TV	✓	✓	✓	✓	✗
ACR	✓	✓	✗	✗	✗
DnICNN	✓	✓	✓	✗	✓
PnP-βCNN	✗	✓	✓	✗	-
PnP-DnCNN	✗	✗	✓	✗	-
CRR-NN	✓	✓	✓	✓	✓

5.3.5 Connections to Deep-Learning Approaches

Our proposed CRR-NNs have a single nonlinear layer, which is rather unusual in an 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 5.14.

Plug-and-Play and Averaged Denoisers

1. Convergent Plug-and-Play

The training procedure proposed for CRR-NNs leads to a convex regularizer $\mathcal{R}_{\boldsymbol{\theta}}$, whose proximal operator (5.48) is a good denoiser. Conversely, the proximal operator can be replaced by a powerful denoiser $\mathbf{D}$ in proximal algorithms, which is referred to as PnP. In the PnP-FBS algorithm derived from (5.46) [23, 87], the reconstruction is carried out iteratively via

$$\mathbf{s}_{k+1} = \mathbf{D}\Big(\mathbf{s}_k - \alpha\mathbf{H}^T(\mathbf{H}\mathbf{s}_k - \mathbf{y})\Big), \tag{5.87}$$

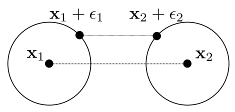


Figure 5.15: The distance between the two noisy images $(\mathbf{x}_1 + \epsilon_1)$ and $(\mathbf{x}_2 + \epsilon_2)$ can be smaller than that between their clean versions $\mathbf{x}_1$ and $\mathbf{x}_2$. This limits the performance of a nonexpansive denoiser $\mathbf{D}$ since $\|\mathbf{D}(\mathbf{x}_1+\epsilon_1)-\mathbf{D}(\mathbf{x}_2+\epsilon_2)\| \leq \|\mathbf{x}_1+\epsilon_1-(\mathbf{x}_2+\epsilon_2)\| < \|\mathbf{x}_1-\mathbf{x}_2\|$ in the scenario depicted.

where α is the stepsize and $\mathbf{D}\colon \mathbb{R}^d \to \mathbb{R}^d$ is a generic denoiser. As mentioned in Section 5.2, a standard set of sufficient conditions[15] to guarantee convergence of the iterations (5.87) is that

1. $\mathbf{D}$ is averaged, namely $\mathbf{D} = \beta\mathbf{N} + (1-\beta)\mathbf{Id}$ where $\beta \in (0,1)$ and $\mathbf{N}\colon \mathbb{R}^n \to \mathbb{R}^n$ is a nonexpansive mapping;
2. $\alpha \in [0, 2/\|\mathbf{H}\|^2)$;
3. the update operator in (5.87) has a fixed point.

In general, Condition (i) is not sufficient to ensure that $\mathbf{D}$ is the proximal operator of some convex regularizer $\mathcal{R}$. Hence, its interpretability is still limited. Further, Condition (ii) implies that $\mathbf{s} \mapsto \big(\mathbf{s} - \alpha\mathbf{H}^T(\mathbf{H}\mathbf{s} - \mathbf{y})\big)$ is averaged. Hence, as averagedness is preserved through composition, the iterates are updated by the application of an averaged operator (see [55] for details). With Condition (iii), the convergence of the iterations (5.87) 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 (see Section 5.2.2), in the same sense as given in Proposition 5.6 for CRR-NNs.

2. Constraint vs Performance

As discussed in [66, 201], the performance of the denoiser $\mathbf{D}$ is in direct competition with its averagedness. A simple illustration of this issue is provided in Figure 5.15. Unsurprisingly, Condition (i) is not met by any learnt state-of-the-art denoiser, and it is usually also relaxed in the PnP literature.

For instance, it is common to use non-1-Lipschitz learning modules, such as batch normalization [57], or to only constrain the residual $(\mathbf{Id} - \mathbf{D})$ to be nonexpansive, which enables one to train a nonexpansive NN in a residual way [55, 57, 253], with the caveat that Lip$(\mathbf{D})$ can be as large as 2. Another recent approach consists of penalizing during training either the norm of the Jacobian of $\mathbf{D}$ at a finite set of locations [66, 254] or of another local estimate of the Lipschitz constant [246, 255]. Interestingly, even slightly relaxed frameworks usually yield significant improvements in the reconstruction quality.

[15]Here, $\mathbf{H}$ can be noninvertible; otherwise, weaker conditions exist [57].

However, they do not provide convergence guarantees for ill-posed inverse problems, which is problematic for sensitive applications such as biomedical imaging.

3. Averaged Deep NNs

To leverage the success of deep learning, $\mathbf{N}$ is typically chosen as a deep CNN of the form[16]

$$\mathbf{N} = \mathbf{C}_K \circ \boldsymbol{\sigma} \circ \cdots \circ \mathbf{C}_2 \circ \boldsymbol{\sigma} \circ \mathbf{C}_1, \tag{5.88}$$

where $\mathbf{C}_k$ are learnable convolutional layers and $\boldsymbol{\sigma}$ is the activation function [57, 211]. To meet Condition (i), $\mathbf{N}$ must be nonexpansive, which one usually achieves by constraining $\mathbf{C}_k$ and $\boldsymbol{\sigma}$ to be nonexpansive. This is predicated on the sub-multiplicativity of the Lipschitz constant with respect to composition; as in $\mathrm{Lip}(\mathbf{f} \circ \mathbf{g}) \leq \mathrm{Lip}(\mathbf{f})\mathrm{Lip}(\mathbf{g})$. Unfortunately, this bound is not sharp and may grossly overestimate $\mathrm{Lip}(\mathbf{f} \circ \mathbf{g})$. 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 [212, 214, 256]. As we saw in Section 5.2, some of these can be resolved by the use of 1-Lipschitz learnable linear spline activation functions instead.

In this work, CRR-NNs are compared against two variants of PnP.

- **PnP-DnCNN** corresponds to the popular implementation given in [57]. 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 (the methods we saw in Section 5.2). 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 [257]. With that in mind, CRR-NNs provide a way to overcome this limitation.

4. Construction of Averaged Denoisers from CRR-NNs

The training of CRR-NNs offers two ways to build averaged denoisers. Since proximal

[16]The benefit of standard skip connections combined with the preservation of the nonexpansiveness of the NN is unclear.

operators are half-averaged, we directly get that the proximal denoiser (5.48) is an averaged operator. For the t-step denoiser, the following holds.

Proposition 5.10. *The t-step denoiser* (5.76) *is averaged for* $\alpha \in [0, 2/(2+\tau L_{\boldsymbol{\theta}})]$ *with* $L_{\boldsymbol{\theta}} = \mathrm{Lip}(\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}})$.

Proof. The t-step denoiser is built from the gradient-step operator $\boldsymbol{T}_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}$. Here, we use the more explicit notation

$$\boldsymbol{T}(\mathbf{s}, \mathbf{y}) = \mathbf{s} - \alpha((\mathbf{s} - \mathbf{y}) + \tau\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}}(\mathbf{s})). \tag{5.89}$$

This makes explicit the dependence on $\mathbf{y}$ and, for simplicity, the dependence on $\mathcal{R}_{\boldsymbol{\theta}}$, τ, and α are omitted. It is known that $\boldsymbol{T}$ is averaged with respect to $\mathbf{s}$ for $\alpha \in (0, 2/(1+\tau L_{\boldsymbol{\theta}}))$. This ensures convergence of gradient descent, but it does not characterize the denoiser itself. The t-step denoiser depends on the initial value $\mathbf{s}_0 = \mathbf{y}$ and is determined by the recurrence relation $\mathbf{s}_{k+1} = \boldsymbol{T}(\mathbf{s}_k, \mathbf{y})$. For the map $\boldsymbol{L}_k \colon \mathbf{y} \mapsto \mathbf{s}_k$, it holds that $\boldsymbol{L}_{k+1} = \boldsymbol{U} \circ \boldsymbol{L}_k + \alpha\mathbf{Id}$, where $\boldsymbol{U} = \mathbf{Id} - \alpha(\mathbf{Id} + \tau\boldsymbol{\nabla}\mathcal{R}_{\boldsymbol{\theta}})$. The Jacobian of $\boldsymbol{U}$ reads $\mathbf{J}_{\boldsymbol{U}} = \mathbf{I} - \alpha(\mathbf{I} + \tau\mathbf{H}_{\mathcal{R}_{\boldsymbol{\theta}}})$ and satisfies that $((1-\alpha) - \alpha\tau L_{\boldsymbol{\theta}})\mathbf{I} \preceq \mathbf{J}_{\boldsymbol{U}} \preceq (1-\alpha)\mathbf{I}$. From this, we infer that

$$\mathrm{Lip}(\mathbf{U}) \leq \max\Big(\alpha\tau L_{\boldsymbol{\theta}} - (1-\alpha), 1-\alpha\Big). \tag{5.90}$$

Since $\alpha \leq 2/(2+\tau L_{\boldsymbol{\theta}})$, 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 $\alpha \in (2/(2+\tau L_{\boldsymbol{\theta}}), 2/(1+\tau L_{\boldsymbol{\theta}}))$, 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 (5.88). 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 5.3.6.

Deep Convex Regularizers

Another approach to leverage deep-learning-based priors with stability and convergence guarantees consists of learning a deep convex regularizer $\mathcal{R}$. These priors are typically parameterized with an ICNN, which is a NN with increasing and convex activation functions along with positive weights for some linear layers [258]. There exist various strategies to train the ICNN.

The adversarial convex regularizer (ACR) framework [231, 259] relies on the adversarial training proposed in [236]. The regularizer is learnt by minimizing its value on clean images and maximizing its value on unregularized reconstructions. This allows for learning non-smooth $\mathcal{R}$ 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 $\mathcal{R}$ being non-smooth, it is challenging to exactly minimize the cost function, but the authors of [231, 259] did not find any practical issues in that matter using gradient-based solvers. To boost the performance of $\mathcal{R}$, they also added a sparsifying filter bank to the ICNN, namely, a convex term of the form $\|\mathbf{Us}\|_1$, where the linear operator $\mathbf{U}$ is made of convolutions learnt conjointly with the ICNN.

In [64], the regularizer is trained so that its gradient step is a good blind Gaussian denoiser. There, the authors use ELU activations in the ICNN[17] to obtain a smooth $\mathcal{R}$.

The aforementioned ICNN-based frameworks [64, 231, 259] have major differences with CRR-NNs: (i) they typically require orders of magnitude more parameters; (ii) the computation of $\boldsymbol{\nabla}\mathcal{R}$, 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 5.3.6). As we shall see, CRR-NNs are much faster to train and tend to perform better (see Section 5.3.6).

5.3.6 Experimental Results

Training of CRR-NNs

The CRR-NNs are trained on a Gaussian-denoising task with noise levels $\sigma \in \{5/255, 25/255\}$. The same procedure as in [57, 154] is used to form 238,400 patches of size (40×40) from 400 images of the BSD500 dataset [227]. For validation, the same 12 images as in [57, 154] are used. The weights $\mathbf{W}$ in $\mathcal{R}_{\boldsymbol{\theta}}$ 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 [234], 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 $\lambda = 2 \times 10^{-3}(255\sigma)$. 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 τ and μ,

[17]The authors also explore non-convex regularization but they offer no guarantees on computing the global minimum.

Table 5.15: Convex models and averaged denoisers tested on BSD68.

	$\sigma = 5/255$	$\sigma = 25/255$
TV[*,‡] [261]	36.41	27.48
Higher-order MRFs[*,‡] [234]	NA	28.04
VN[1,t†][240]	NA	27.69
βCNN$_\sigma$[‡]	36.86	27.93
D$_{\text{ISTA}}$[‡] [260]	36.54	NA
GS-DnICNN[†][64]	36.85	27.76
D$_{\text{ADMM}}$[‡][260]	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 (averaged layers)

to 10^{-3} for $\mathbf{W}$, and to 5×10^{-5} for $\mathbf{c}_i$.

Recall that for a given t, the training yields two denoisers.

- ***t*-Step Denoiser**: This corresponds to $\boldsymbol{T}^t_{\mathcal{R}_{\boldsymbol{\theta}},\tau,\alpha}$ 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 [157, 260], which in general also do not correspond to minimizing an energy.

- **Proximal Denoiser**: The learnt regularizer $\mathcal{R}_{\boldsymbol{\theta}}$ is plugged into (5.85) with $\mathbf{H} = \mathbf{I}$, and the solution is computed using Algorithm 3 with small tolerance (10^{-6} for the relative change of norm between consecutive iterates). The parameters τ and μ are tuned on the validation dataset with the coarse-to-fine method given in Appendix 5.4.7. 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.

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 5.15 on the standard BSD68 test set.

Now, we briefly give the implementation details of the various frameworks. CRR-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 [64], we train

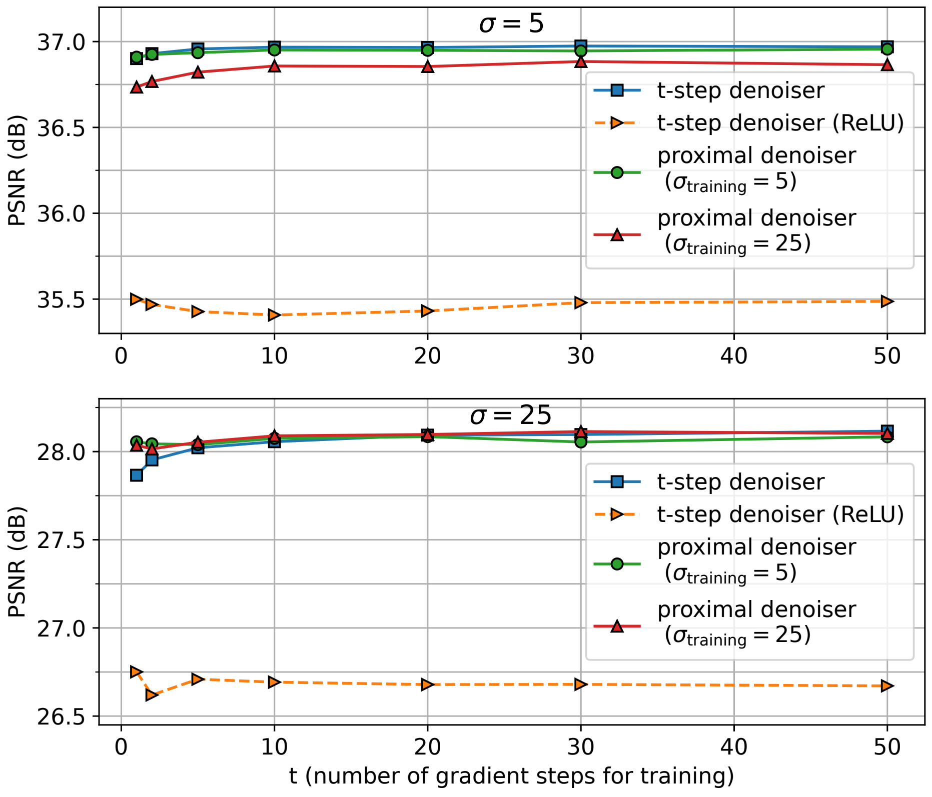


Figure 5.16: Test denoising performance of CRR-NNs for noise level $\sigma = 5/255$ and $\sigma = 25/255$ versus the number of gradient steps used for training, the denoiser type (t-step vs proximal), and the noise level used for training.

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. For the averaged deep CNN denoiser $\beta\text{CNN}_\sigma = \beta\mathbf{N} + (1-\beta)\mathbf{Id}$, we took $\mathbf{N}$ as the LLS network (with $\lambda = 10^{-6}$) from Section 5.2, trained as a denoiser separately for $\sigma \in \{5/255, 25/255\}$, and we set $\beta = 0.99$. 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 [261]. The results for all models are presented in Table 5.15 and Figure 5.16.

- ***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 in our experiments of Section 5.2 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 $\text{VN}^{1,t}$ of variational-network denoisers proposed in [240], which also partially minimizes a convex cost. With a convex model similar to CRR-NNs (see Section 5.3.4 for a discussion), it is shown that an increase in t decreases the performance (reported as $\mathrm{VN}_{24}^{1,t}$ in [240, Figure 5]). The model $\mathrm{VN}^{1,t}$ cannot compete with the proximal denoiser trained with bilevel optimization in [234]. By contrast, for $\sigma = 25/255$ we obtain an improvement over $\mathrm{VN}^{1,t}$ of 0.2dB for $t = 1$, and more than 0.6dB as t increases. Note that, in [240], the layers of the t-step $\mathrm{VN}^{1,t}$ denoiser are not guaranteed to be averaged. Our models also outperform the averaged $\beta\mathrm{CNN}_\sigma$ (+0.1dB for $\sigma = 5$, +0.2dB for $\sigma = 25/255$), and the two averaged denoisers $\mathrm{D}_{\mathrm{ISTA}}$ and $\mathrm{D}_{\mathrm{ADMM}}$ [260] (+0.4/+0.3dB for $\sigma = 5/255$). In 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 [234] (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.

Biomedical Image Reconstruction

The six CRR-NNs trained on denoising with $t \in \{1, 10, 50\}$ and $\sigma \in \{5/255, 25/255\}$ are now used to solve the MRI and CT reconstruction problems described in the experiments of Section 5.2.

1. Reconstruction Frameworks

A reconstruction with isotropic TV regularization is computed with FISTA [23], in which $\mathrm{prox}_{\mathcal{R}}$ is computed as in [26] to enforce positivity. We also consider reconstructions obtained with the PnP method with (i) provably averaged denoisers $\beta\mathrm{CNN}_\sigma$ ($\sigma = 5, 25$); and (ii) the popular pretained DnCNNs [57] ($\sigma = 5, 15, 40$). 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 $\mathbf{D}_\gamma = \gamma\mathbf{D} + (1-\gamma)\mathbf{Id}$ for all denoisers $\mathbf{D}$, where $\gamma \in (0, 1]$ is tuned along with the stepsize α given in (5.87). We only report the performance of the best-performing setting. The ACR framework [231, 259] yields a convex regularizer for (5.46) that is specifically designed to the described CT problem. To be consistent with [231, 259], we apply 400 iterations of gradient descent, even though the objective is nonsmooth, and tune the stepsize and τ. The results are consistent with those reported in [231, 259].

Table 5.16: 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	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 [57]	40.52	**39.02**	0.956	**0.935**	**35.24**	**34.63**	**0.884**	**0.840**

Table 5.17: 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

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 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 5.4.7.

2. Results and Discussion For each modality, a reconstruction example is given for each framework in Figures 5.17 and 5.18. The PSNR and SSIM values for the test set given in Tables 5.16, 5.18, and 5.20 attest that CRR-NNs consistently outperform the other frameworks with comparable guarantees. It can be seen from Tables 5.17, 5.19, and 5.21 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 typically yields artifact-free reconstructions but is more prone to over-smoothing (Figure 5.17) Lastly, observe that the properly constrained PnP-βCNN is not always better than TV. This

Table 5.18: 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	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 [57]	**39.55**	**38.52**	<u>0.947</u>	**0.929**	**35.11**	**35.14**	**0.881**	**0.858**

Table 5.19: 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 5.20: 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	38.19	0.931	35.15	0.897	31.85	0.844
ACR [231, 259]	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 [57]	<u>38.93</u>	0.941	**36.49**	**0.921**	**33.52**	**0.897**

Table 5.21: 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

confirms the difficulty of training provably 1-Lipchitz CNN, which is also reported for MRI image reconstruction in [257].

Under the Hood of the Learnt Regularizers

The filters and activation functions for learnt CRR-NNs with $\sigma \in \{5/255, 25/255\}$ and $t = 5$ are shown in Figures 5.19 and 5.20.

1. 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 learnt kernels typically come in groups that are reminiscent of 2D steerable filters [262, 263]. Interestingly, their support is wider when the denoising task is carried out for $\sigma = 25/255$ than for $\sigma = 5/255$.

2. Activation Functions

The linear splines converge to simple functions throughout the training. The regularization (5.79) 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 $\text{Clip}(x) = \text{sign}(x)\min(|x|, 1)$. The learnt regularizer is closely related to ℓ_1-norm based regularization as many of the learnt convex profiles ψ_i resemble some smoothed version of the absolute-value function.

3. 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

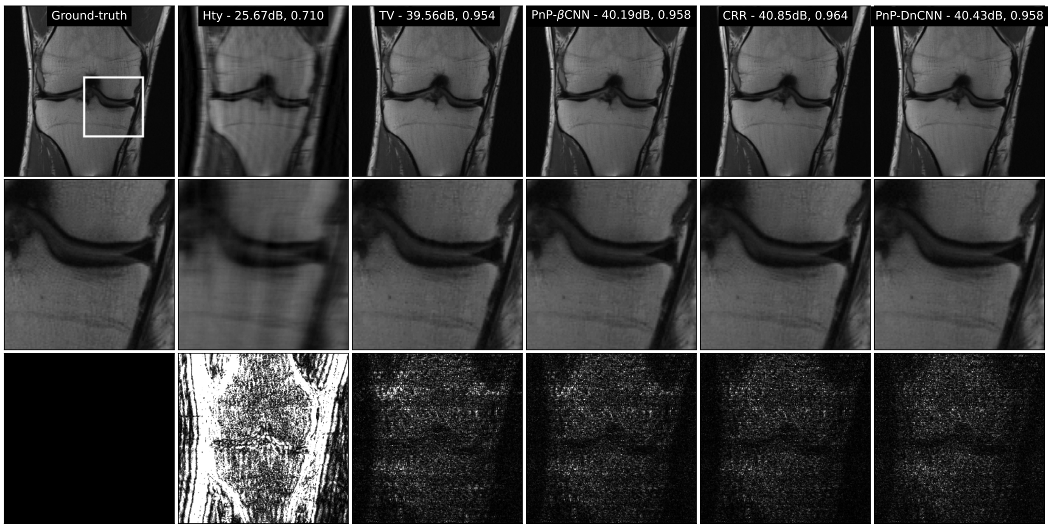


Figure 5.17: Reconstructed images for the 4-fold accelerated multi-coil MRI experiment. The reported metrics are PSNR and SSIM. The last row shows the squared differences between the reconstructions and the ground-truth image.

over deep models, which can usually not be pruned efficiently.

4. A Signal-Processing Interpretation

Given that the gradient-step operator $\mathbf{s} \mapsto (\mathbf{s} - \alpha \mathbf{W}^T \boldsymbol{\sigma}(\mathbf{W}\mathbf{s}))$ of the learnt regularizer is expected to remove some noise from $\mathbf{s}$, the 1-hidden-layer CNN $\mathbf{W}^T \boldsymbol{\sigma}(\mathbf{W}\cdot)$ is expected to extract noise. The response of $\mathbf{s}$ to the learnt filters forms the high-dimensional representation $\mathbf{W}\mathbf{s}$ of $\mathbf{s}$. The clipping function preserves the small responses to the filters, while it cuts the large ones. Hence, the estimated noise $\mathbf{W}^T \boldsymbol{\sigma}(\mathbf{W}\mathbf{s})$ is reconstructed by essentially removing the components of $\mathbf{s}$ 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 $\mathbf{x} \mapsto \|\mathrm{DWT}(\mathbf{s})\|_1$ is given by

$$\begin{aligned} \mathrm{prox}_{\|\mathrm{DWT}(\cdot)\|_1}(\mathbf{s}) &= \mathrm{IDWT}(\mathrm{soft}(\mathrm{DWT}(\mathbf{s}))) \\ &= \mathbf{s} - \mathrm{IDWT}(\mathrm{clip}(\mathrm{DWT}(\mathbf{s}))), \end{aligned} \tag{5.91}$$

where 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 $\mathrm{IDWT}(\mathrm{DWT}(\mathbf{s})) = \mathbf{s}$ and $\mathrm{soft}(\mathbf{s}) = (\mathbf{s} - \mathrm{clip}(\mathbf{s}))$. 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 learnt filters are not orthonormal ($\mathbf{W}^T\mathbf{W} \neq \mathbf{I}$).

5. Role of the Scaling Factor

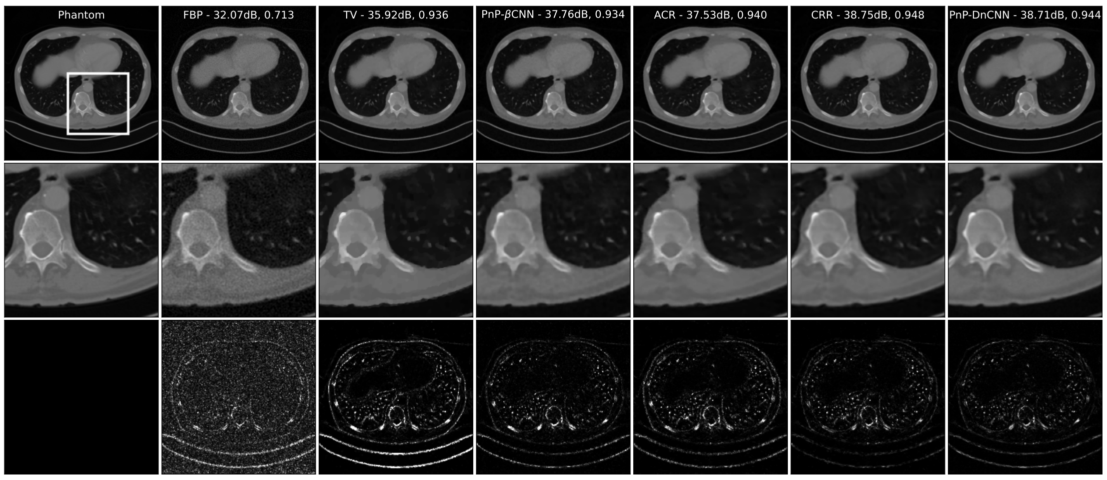


Figure 5.18: Reconstructed images for the CT experiment with $\sigma_{\mathbf{n}} = 0.5$. The reported metrics are PSNR and SSIM. The last row shows the squared differences between the reconstructions and the ground-truth image.

To clarify the role of the scaling factor μ introduced in (5.85), 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 $\mathcal{R}_1$ can be obtained from $\mathcal{R}_{\boldsymbol{\theta}}$ by replacing the 2D convolutional filters with 1D convolutional filters whose kernels are the ones of $\mathcal{R}_{\boldsymbol{\theta}}$ summed along a direction. Next, we seek a compactly supported signal with fixed mass that has minimum regularization cost, as in

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathbb{R}^d} \mathcal{R}_1(\mu\mathbf{c}) \text{ s.t. } \begin{cases} \mathbf{1}^T\mathbf{c} = 1, \\ \mathbf{c}_k = 0, \quad \forall k \notin [k_1, k_2]. \end{cases} \tag{5.92}$$

The solutions for various values of μ are shown in Figure 5.21. 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, allows 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 5.19 and 5.20. 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.

5.3.7 Summary

We have proposed a framework to learn universal convex-ridge regularizers with adaptive profiles (implemented using learnable linear spline activation functions). 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.

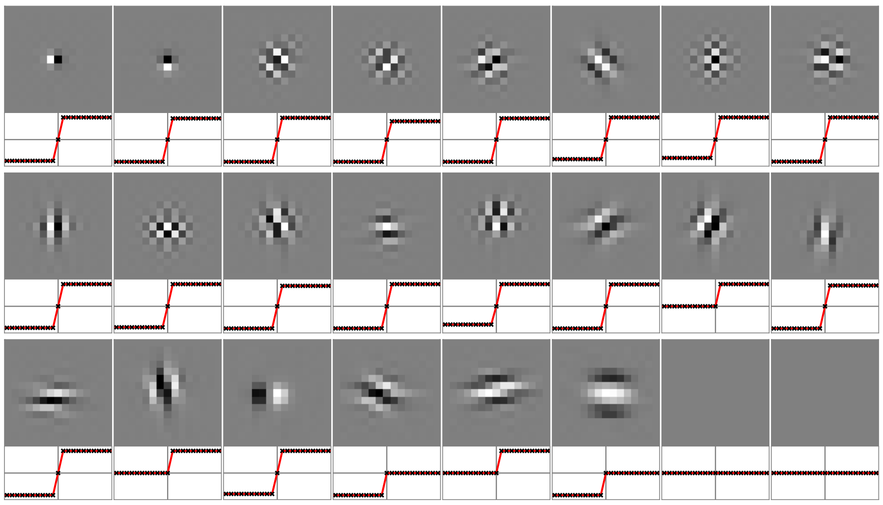


Figure 5.19: Impulse response of the filters and activation functions of the CRR-NN trained with $\sigma = 5$. The crosses indicate the knots of the splines. For the 8 missing filters, the associated activation functions were numerically identically zero.

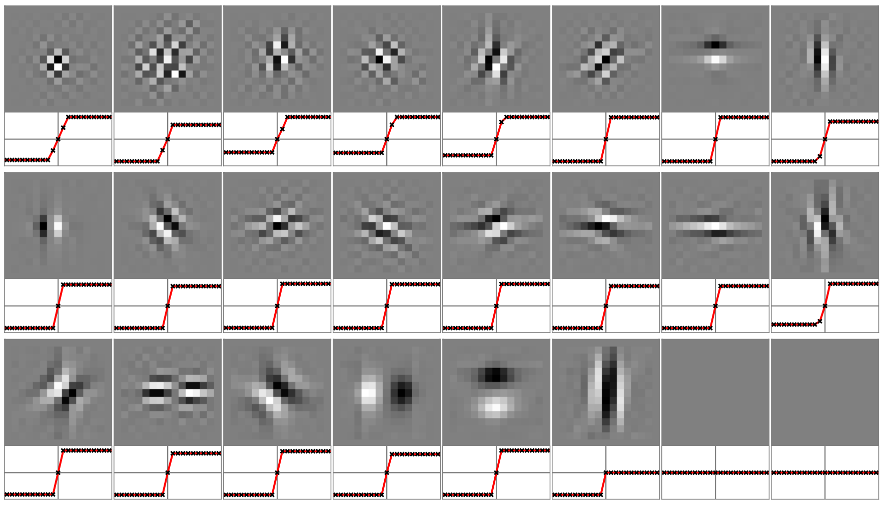


Figure 5.20: Impulse response of the filters and activation functions of the CRR-NN trained with $\sigma = 25/255$.

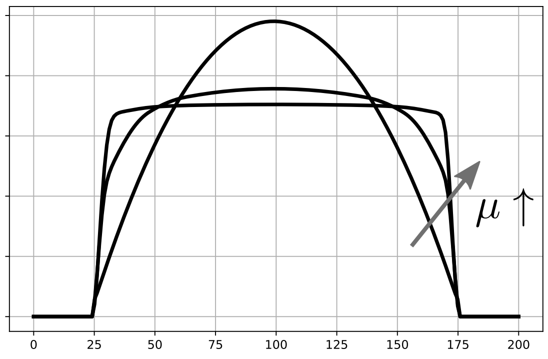


Figure 5.21: Solutions of the one-dimensional problem (5.92) for increasing values of μ. The plotted functions are supported in $[25, 175]$ and minimize the learnt regularizer given a unit sum of their values.

5.4 Appendix

5.4.1 Second-Order Total Variation

In this section, we briefly explain the notion of second-order total variation and provide the definition of the corresponding native space $\mathrm{BV}^{(2)}(\mathbb{R})$. We refer to [164] for more details.

The second-order total-variation seminorm of a function $f : \mathbb{R} \to \mathbb{R}$ is defined as

$$\mathrm{TV}^{(2)}(f) = \|\mathrm{D}^2 f\|_{\mathcal{M}}, \tag{5.93}$$

where D is the (weak) derivative operator and the total-variation norm $\|\cdot\|_{\mathcal{M}}$ is defined over the Banach space $\mathcal{M}(\mathbb{R})$ of bounded Radon measures as

$$\|w\|_{\mathcal{M}} \triangleq \sup_{\varphi\in\mathcal{S}(\mathbb{R}):\ \|\varphi\|_\infty=1} \langle w, \varphi\rangle,$$

where $\mathcal{S}(\mathbb{R})$ is Schwartz' space of smooth and rapidly decaying test functions. The space $\mathcal{M}(\mathbb{R})$ is a generalization of the space $L_1(\mathbb{R})$ of absolutely integrable functions, in the sense that $L_1(\mathbb{R}) \subseteq \mathcal{M}(\mathbb{R})$ and, for any $f \in L_1(\mathbb{R})$, the two norms satisfy $\|f\|_{L_1} = \|f\|_{\mathcal{M}}$. The generalized space $\mathcal{M}(\mathbb{R})$ is, however, larger than $L_1(\mathbb{R})$ as it contains the set of all shifted Dirac impulses $\delta(\cdot - \tau)$ with $\|\delta(\cdot - \tau)\|_{\mathcal{M}} = 1$ for any $\tau \in \mathbb{R}$. In particular, this implies that

$$w_\delta = \sum_{k\in\mathbb{Z}} a[k]\delta(\cdot - \tau_k) \in \mathcal{M}(\mathbb{R}) \quad \text{and} \quad \|w_\delta\|_{\mathcal{M}} = \sum_{k\in\mathbb{Z}} \Big|a[k]\Big|$$

for any absolutely summable sequence $a[\cdot] \in \ell_1(\mathbb{Z})$. Likewise, since $\mathrm{D}^2\{(\cdot - \tau_k)_+\} = \delta(\cdot - \tau_k)$ (Green's function property), one readily deduces that $\mathrm{TV}^{(2)}(\sigma) = \|\mathbf{a}\|_{\ell_1}$ for the generic spline activation function defined by (5.8).

Finally, the native space $\mathrm{BV}^{(2)}(\mathbb{R})$ is the space of functions with second-order bounded variation

$$\mathrm{BV}^{(2)}(\mathbb{R}) = \{f : \mathbb{R} \to \mathbb{R} : \mathrm{TV}^{(2)}(f) < +\infty\}.$$

5.4.2 Learnable Spline Activation Function Module

In this section, we describe our implementation of the B-spline formulation of the learnable linear-spline activation functions. We also detail our sparsification procedure which is a postprocessing step during training; the intent is to control the number of active knots in the network.

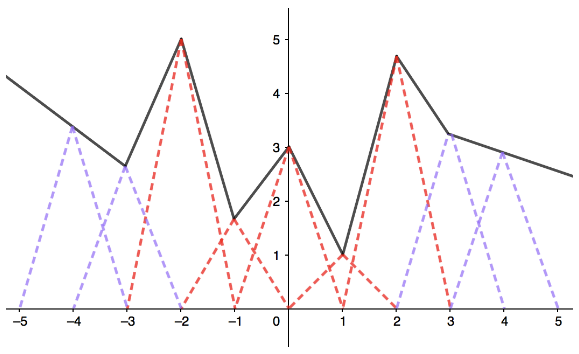


Figure 5.22: The left and right linear extrapolations beyond $[-3, 3]$ of the activation function are computed with the help of two extra B-splines on each side.

B-Spline Formulation

We place a highly redundant set of knots (for the linear spline) on a finite uniform grid of size T. The cardinality of this set of knots is K, with K odd. We define the indices $k_{\min} = -(K-1)/2$ and $k_{\max} = (K-1)/2$. The spline we want to build will extend linearly outside the interval $[k_{\min}T, k_{\max}T]$ and can be represented in the gridded ReLU basis as

$$\sigma(x) = b_0 + b_1 x + \sum_{k=k_{\min}}^{k_{\max}} a_k (x - kT)_+, \tag{5.94}$$

with $\mathrm{TV}^{(2)}(\sigma) = \|\mathbf{a}\|_1$.

Here, we represent σ in a B-spline basis as

$$\sigma(x) = \begin{cases} c_{k_{\min}} + \frac{1}{T}(c_{k_{\min}} - c_{k_{\min}-1})(x - k_{\min}T), & x \in (-\infty, k_{\min}T) \\ \sum\limits_{k=k_{\min-1}}^{k_{\max+1}} c_k \varphi_T(x - kT), & x \in [k_{\min}T, k_{\max}T] \\ c_{k_{\max}} + \frac{1}{T}(c_{k_{\max}+1} - c_{k_{\max}})(x - k_{\max}T), & x \in (k_{\max}T, \infty), \end{cases} \tag{5.95}$$

where φ_T is the triangle-shaped B-spline

$$\varphi_T(x) = \begin{cases} 1 - \left|\frac{x}{T}\right|, & -T \le x \le T, \\ 0, & \text{otherwise.} \end{cases} \tag{5.96}$$

The B-spline representation in (5.95) is equivalent to the one in (5.9). Here, we place $K+2$ triangular basis functions on the grid and, instead of using one-sided boundary basis functions, the linear extrapolations beyond $[k_{\min}T, k_{\max}T]$ are handled with the help of the last two B-spline coefficients on each side: $(k_{\min-1}, k_{\min})$ and $(k_{\max}, k_{\max+1})$. An example of this construction is shown in Figure 5.22.

The relationship between the ReLU coefficients $\mathbf{a} \in \mathbb{R}^K$ and the B-spline coefficients

$\mathbf{c} \in \mathbb{R}^{K+2}$ is given by

$$\begin{bmatrix} a_{k_{\min}} \\ \vdots \\ a_{k_{\max}} \end{bmatrix} = \frac{1}{T} \underbrace{\begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 & -2 & 1 \end{bmatrix}}_{\mathbf{L} \in \mathbb{R}^{(K+2)\times K}} \begin{bmatrix} c_{k_{\min}-1} \\ c_{k_{\min}} \\ \vdots \\ c_{k_{\max}} \\ c_{k_{\max+1}} \end{bmatrix}, \tag{5.97}$$

while the linear-term parameters b_0, b_1 can be determined from $c_{k_{\min-1}}$ and $c_{k_{\min}}$. From (5.97), we see that the $\mathrm{TV}^{(2)}$ regularization of σ can also be computed from the B-spline coefficients as $\mathrm{TV}^{(2)}(\sigma) = \|\mathbf{Lc}\|_1$.

Sparsification

To train networks with learnable spline nonlinearities, we augment the cost function with the $\mathrm{TV}^{(2)}$ regularization of the activation functions. This translates into an ℓ_1-penalty on the ReLU coefficients $\mathbf{a} = (a_k)$ or, equivalently, on the filtered version of the B-spline coefficients $\mathbf{Lc}$. We rely on the sparsifying effect of the ℓ_1-norm to remove some of the redundant knots. In practice, we observe that, while some of the coefficients $a_k = [\mathbf{Lc}]_k$ attain small values, they never vanish entirely. In order to fix this and have a tight control on the number of knots, we have applied a further “sparsification" as a postprocessing step after training.

The first step is to retrieve the ReLU coefficients $\mathbf{a}$ from the trained B-spline coefficients $\mathbf{c}$ using (5.97). Then, every coefficient a_k with absolute value below a certain threshold is set to zero, yielding $\hat{\mathbf{a}} = (\hat{a}_k)$. Finally, we transform these modified ReLU coefficients to the new B-spline coefficients $\hat{\mathbf{c}}$. In this step, the coefficients $\hat{c}_{k_{\min-1}}$ and $\hat{c}_{k_{\min}}$ that determine the linear term are assigned the same values as $c_{k_{\min-1}}$ and $c_{k_{\min}}$, respectively. The other coefficients $\hat{c}_k$ are computed from $\hat{a}_k$ using the relations in (5.97). The sparsification is achieved by selecting the maximum threshold such that the training accuracy does not drop by more than 0.2%.

5.4.3 Hyperparameter Tuning: Training Deep Spline Neural Networks

In this section, we propose a method to tune the hyperparameters of Problems (5.3) and (5.6). Our hyperparameter-tuning method is based on some optimality conditions that we prove for the global minimizers of these problems. It is flexible with respect to the choice of linear layers and architecture and can be applied to any deep spline network.

Optimality Conditions

The main principle of our optimality conditions is based on the scale- and dilation-invariance properties of the second-order total-variation regularization, as we state in Proposition 5.11.

Proposition 5.11. *The second-order total-variation regularization* $\mathrm{TV}^{(2)}:\mathrm{BV}^{(2)}(\mathbb{R})\rightarrow\mathbb{R}$ *is scale- and dilation-invariant. Specifically, for any* $\sigma\in\mathrm{BV}^{(2)}(\mathbb{R})$ *and any* $c\neq 0$, *we have that*

$$\mathrm{TV}^{(2)}\left(c\sigma\right)=|c|\mathrm{TV}^{(2)}\left(\sigma\right), \tag{5.98}$$

and

$$\mathrm{TV}^{(2)}\left(\sigma(c\cdot)\right)=|c|\mathrm{TV}^{(2)}\left(\sigma\right). \tag{5.99}$$

Proof. We first recall that

$$\mathrm{TV}^{(2)}(\sigma)=\|\mathrm{D}^2\sigma\|_{\mathcal{M}}=\sup_{\varphi\in\mathcal{S}(\mathbb{R})\backslash\{0\}}\frac{\langle\mathrm{D}^2\sigma,\varphi\rangle}{\|\varphi\|_\infty}. \tag{5.100}$$

One deduces (5.98) from the linearity of D^2 and the homogeneity of the $\mathcal{M}$-norm. To derive (5.99), we use the relation $\mathrm{D}^2\{\sigma(c\cdot)\}=c^2\mathrm{D}^2\{\sigma\}(c\cdot)$ and the equality $\langle f(c\cdot),g\rangle=c^{-1}\langle f,g(\cdot/c)\rangle$ which, together with (5.100), yields

$$\mathrm{TV}^{(2)}(\sigma(c\cdot))=\sup_{\varphi\in\mathcal{S}(\mathbb{R})\backslash\{0\}}c\frac{\langle\mathrm{D}^2\sigma,\varphi(\cdot/c)\rangle}{\|\varphi\|_\infty}=|c|\sup_{\psi\in\mathcal{S}(\mathbb{R})\backslash\{0\}}\frac{\langle\mathrm{D}^2\sigma,\psi\rangle}{\|\psi\|_\infty}, \tag{5.101}$$

where the latter is obtained via the change of variable $\psi=\mathrm{sgn}(c)\varphi(\cdot/c)$. The last step is to notice that

$$\sup_{\psi\in\mathcal{S}(\mathbb{R})\backslash\{0\}}\frac{\langle\mathrm{D}^2\sigma,\psi\rangle}{\|\psi\|_\infty}=\mathrm{TV}^{(2)}(\sigma)$$

which, when combined with (5.101), yields (5.99). □

In Theorem (5.1), we prove that the energies of all linear and nonlinear layers of any global minimizer of (5.3) are inversely proportional to their corresponding regularization parameters.

Theorem 5.1. *Let* $\mathbf{f}_{\boldsymbol{\theta}^*}$ *be a global minimizer of* (5.3) *with linear parameters* $\boldsymbol{\phi}_\ell^*$ *and learned activation functions* $\mathbf{g}_\ell^*$. *Then, we have that*

$$2\mu_1\|\boldsymbol{\phi}_1^*\|_2^2=\lambda_1\mathrm{TV}^{(2)}(\mathbf{g}_1^*)=\cdots=\lambda_{L-1}\mathrm{TV}^{(2)}(\mathbf{g}_{L-1}^*)=2\mu_L\|\boldsymbol{\phi}_L^*\|_2^2. \tag{5.102}$$

Proof. Let us denote by G^* the geometric mean of the $L+2(L-1)=(3L-2)$ quantities

$$\{\mu_\ell\|\boldsymbol{\phi}_\ell^*\|_2^2\}_{\ell=1}^L\bigcup\left\{\frac{\lambda_\ell}{2}\mathrm{TV}^{(2)}(\mathbf{g}_\ell^*)\right\}_{\ell=1}^{L-1}\bigcup\left\{\frac{\lambda_\ell}{2}\mathrm{TV}^{(2)}(\mathbf{g}_\ell^*)\right\}_{\ell=1}^{L-1}.$$

It turns out that G^* can be computed via the relation

$$G^{*(3L-2)} = \left(\prod_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell^*\|_2^2\right) \left(\prod_{\ell=1}^{L-1} \frac{\lambda_\ell}{2} \mathrm{TV}^{(2)}(\mathbf{g}_\ell^*)\right)^2.$$

Due to the inequality of arithmetic and geometric means (AM and GM, respectively), we have that

$$(3L-2)G^* \leq \sum_{\ell=1}^{L-1} \lambda_\ell \mathrm{TV}^{(2)}(\mathbf{g}_\ell^*) + \sum_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell^*\|_2^2, \tag{5.103}$$

where the inequality is saturated if and only if (5.102) holds.

Inspired from the mentioned AM-GM inequality, we now define a new set of linear parameters $\tilde{\boldsymbol{\phi}}_\ell$, $\ell = 1, \dots, L$ and adjustable activation functions $\tilde{\mathbf{g}}_\ell$, $\ell = 1, \dots, L-1$, as

$$\tilde{\boldsymbol{\phi}}_\ell = c_\ell \boldsymbol{\phi}_\ell, \qquad c_\ell = \left(\frac{G^*}{\mu_\ell \|\boldsymbol{\phi}_\ell\|_2^2}\right)^{\frac{1}{2}},$$
$$\tilde{\mathbf{g}}_\ell = d_\ell \mathbf{g}_\ell\left(\frac{\cdot}{c_\ell d_{\ell-1}}\right), \qquad d_\ell = c_\ell d_{\ell-1} \frac{G^*}{\frac{\lambda_\ell}{2}\mathrm{TV}^{(2)}(\mathbf{g}_\ell)},$$

with the convention that $d_0 = 1$. Let us specify the corresponding linear and nonlinear layers by $\tilde{\mathbf{W}}_\ell$ and $\tilde{\boldsymbol{\sigma}}_\ell$, respectively. One readily observes that

$$\tilde{\mathbf{W}}_\ell = c_\ell \mathbf{W}_\ell, \qquad \tilde{\boldsymbol{\sigma}}_\ell = d_\ell \boldsymbol{\sigma}_\ell\left(\frac{\cdot}{c_\ell d_{\ell-1}}\right)$$

in all layers. Interestingly, the input-output relation of this new neural network is the same as that of $\mathbf{f}_{\boldsymbol{\theta}^*}$. This is due to two simple observations.

- For $\ell = 1, \dots, L-1$, we have that
$$\tilde{\mathbf{W}}_\ell \circ \tilde{\boldsymbol{\sigma}}_\ell(\cdot) = d_\ell \mathbf{W}_\ell \circ \boldsymbol{\sigma}(\cdot / d_{\ell-1}).$$
- For the output-layer, we have that $c_L d_{L-1} = 1$.

Since the input-output relation remains unchanged, the data-fidelity term in the cost functional of the minimization (5.3) does not change either. Now, due to the optimality of $\mathbf{f}_{\boldsymbol{\theta}^*}$, we deduce that

$$\sum_{\ell=1}^{L-1} \lambda_\ell \mathrm{TV}^{(2)}(\mathbf{g}_\ell^*) + \sum_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell^*\|_2^2 \leq \sum_{\ell=1}^{L-1} \lambda_\ell \mathrm{TV}^{(2)}(\tilde{\mathbf{g}}_\ell) + \sum_{\ell=1}^{L} \mu_\ell \|\tilde{\boldsymbol{\phi}}_\ell\|_2^2. \tag{5.104}$$

Using Proposition 5.11, we have that

$$\lambda_\ell \mathrm{TV}^{(2)}(\tilde{\mathbf{g}}_\ell) = \lambda_\ell \frac{d_\ell}{c_\ell d_{\ell-1}} \mathrm{TV}^{(2)}(\mathbf{g}_\ell) = 2G^*,$$

for $\ell = 1, \ldots, L-1$. Similarly, from the scale invariance of the ℓ_2-norm, we deduce that

$$\mu_\ell \|\tilde{\boldsymbol{\phi}}_\ell\|_2^2 = \mu_\ell c_\ell^2 \|\boldsymbol{\phi}_\ell^*\|_2^2 = G^*.$$

Replacing these in (5.104), we obtain that

$$\sum_{\ell=1}^{L-1} \lambda_\ell \mathrm{TV}^{(2)}(\mathbf{g}_\ell^*) + \sum_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell^*\|_2^2 \leq (3L-2)G^*,$$

which is the converse of the AM-GM inequality (5.103). This shows that (5.103) is saturated and, hence, that (5.102) holds. □

For the case where the activation functions are shared across layers, we show in Theorem 5.2 that the optimal configuration is such that there would be a balance between the total energy of linear layers and the second-order total variation of the learned activation functions.

Theorem 5.2. *Let $f_{\boldsymbol{\theta}^*}$ be a global minimizer of* (5.6). *Then, we have that*

$$\lambda \mathrm{TV}^{(2)}(\mathbf{g}^*) = 2 \sum_{\ell=1}^{L-1} \mu_\ell \|\boldsymbol{\phi}_\ell^*\|_2^2. \tag{5.105}$$

Proof. The proof is very similar to the one for Theorem 5.1. We define G^* as

$$G^* = \left(\frac{\lambda}{2} \mathrm{TV}^{(2)}(\mathbf{g}^*)\right)^{\frac{2}{3}} \left(\sum_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell\|_2^2\right)^{\frac{1}{3}}.$$

The AM-GM inequality implies in this case that

$$3G^* \leq \lambda_\ell \mathrm{TV}^{(2)}(\mathbf{g}^*) + \sum_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell^*\|_2^2, \tag{5.106}$$

with equality if and only if (5.105) holds. Now, we define a new set of linear parameters and adjustable activation functions as

$$\begin{aligned} \tilde{\boldsymbol{\phi}}_\ell &= c^{-1} \boldsymbol{\phi}_\ell, & \ell &= 1, \ldots, L, \\ \tilde{\mathbf{g}} &= c\mathbf{g}(c\cdot), \end{aligned}$$

where the constant $c > 0$ is

$$c = \frac{G^*}{\frac{\lambda}{2}\mathrm{TV}^{(2)}(\mathbf{g})}.$$

Again, the data-fidelity term remains unchanged. From the optimality of $\mathbf{f}_{\boldsymbol{\theta}^*}$, we deduce that

$$\lambda \mathrm{TV}^{(2)}(\mathbf{g}^*) + \sum_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell^*\|_2^2 \leq \lambda \mathrm{TV}^{(2)}(\tilde{\mathbf{g}}) + \sum_{\ell=1}^{L} \mu_\ell \|\tilde{\boldsymbol{\phi}}_\ell\|_2^2.$$

By direct calculations, similar to what we did in Theorem 5.1, we simplify the above inequality into

$$\lambda \mathrm{TV}^{(2)}(\mathbf{g}^*) + \sum_{\ell=1}^{L} \mu_\ell \|\boldsymbol{\phi}_\ell^*\|_2^2 \leq 3G^*$$

which, together with (5.106) implies that the AM-GM equality holds, ultimately leading to (5.105). □

Hyperparameter Tuning

Using Theorems 5.1 and 5.2, we now introduce a way to tune the hyperparameters of our optimization problems. The main idea is to enforce the optimality condition in the initial settings (before training) and, consequently, to reduce the dimension of the hyperparameter space so that it is sufficient to perform a grid search over a single parameter.

Our scheme is described as follows:

1. Initialize the linear parameters $\boldsymbol{\phi}_\ell^0$ (*e.g.,* using Xavier's rule) and the activation functions $\mathbf{g}_\ell^0$ (*e.g.,* soft-threshold/absolute value) and compute the quantities $\|\boldsymbol{\phi}_\ell^0\|_2^2$ and $\mathrm{TV}^{(2)}(\mathbf{g}_\ell^0)$ for all layers.

2. Set
$$\mu_\ell = \frac{C}{2\|\boldsymbol{\phi}_\ell^0\|_2^2}, \tag{5.107}$$
where $C > 0$ is the unique hyperparameter that is required to be tuned.

3. If the activation functions are shared across layers, set
$$\lambda = \frac{(L-1)C}{\mathrm{TV}^{(2)}(\mathbf{g}^0)}. \tag{5.108}$$
Otherwise, set
$$\lambda_\ell = \frac{C}{\mathrm{TV}^{(2)}(\mathbf{g}_\ell^0)}. \tag{5.109}$$

4. Perform a grid search to find the optimal value of $C > 0$.

5.4.4 Stability Results for PnP-FBS

Proof of Proposition 5.1

We start by showing that, if D is β-averaged with $\beta \leq 1/2$, then $2\mathrm{D} - \mathrm{Id}$ is 1-Lipschitz since

$$\begin{aligned}\|(2\mathrm{D} - \mathrm{Id})(\mathbf{z}_1 - \mathbf{z}_2)\|_2 &= \|2\beta(\mathrm{N}(\mathbf{z}_1) - \mathrm{N}(\mathbf{z}_2)) + (1-2\beta)(\mathbf{z}_1 - \mathbf{z}_2)\|_2 \\ &\leq 2\beta\|\mathrm{N}(\mathbf{z}_1) - \mathrm{N}(\mathbf{z}_2)\|_2 + (1-2\beta)\|\mathbf{z}_1 - \mathbf{z}_2\|_2 \\ &\leq \|\mathbf{z}_1 - \mathbf{z}_2\|_2, \quad \forall \mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^n. \end{aligned} \tag{5.110}$$

Let $f(\mathbf{Hs}, \mathbf{y}) = \frac{1}{2}\|\mathbf{Hs} - \mathbf{y}\|_2^2$. Using the above property, we get that

$$\begin{aligned}&\|(2\mathrm{D} - \mathrm{Id})(\mathbf{s}_1^* - \alpha\boldsymbol{\nabla} f(\mathbf{Hs}_1^*, \mathbf{y}_1)) - (2\mathrm{D} - \mathrm{Id})(\mathbf{s}_2^* - \alpha\boldsymbol{\nabla} f(\mathbf{Hs}_2^*, \mathbf{y}_2))\|_2 \\ &\qquad \leq \|(\mathbf{s}_1^* - \alpha\boldsymbol{\nabla} f(\mathbf{Hs}_1^*, \mathbf{y}_1)) - (\mathbf{s}_2^* - \alpha\boldsymbol{\nabla} f(\mathbf{Hs}_2^*, \mathbf{y}_2))\|_2\end{aligned} \tag{5.111}$$

and, from the fixed-point property of $\mathbf{s}_1^*$ and $\mathbf{s}_2^*$, we get that

$$\begin{aligned}&\|2(\mathbf{s}_1^* - \mathbf{s}_2^*) - (\mathbf{s}_1^* - \alpha\boldsymbol{\nabla} f(\mathbf{Hs}_1^*, \mathbf{y}_1)) + (\mathbf{s}_2^* - \alpha\boldsymbol{\nabla} f(\mathbf{Hs}_2^*, \mathbf{y}_2))\|_2 \\ &\qquad \leq \|(\mathbf{s}_1^* - \alpha\boldsymbol{\nabla} f(\mathbf{Hs}_1^*, \mathbf{y}_1)) - (\mathbf{s}_2^* - \alpha\boldsymbol{\nabla} f(\mathbf{Hs}_2^*, \mathbf{y}_2))\|_2.\end{aligned} \tag{5.112}$$

Using the fact that $\boldsymbol{\nabla} f(\mathbf{Hs}, \mathbf{y}) = \mathbf{H}^T(\mathbf{Hs} - \mathbf{y})$ and developing on both sides, we get that

$$\langle \mathbf{s}_1^* - \mathbf{s}_2^*, \mathbf{H}^T(\mathbf{Hs}_2^* - \mathbf{y}_2) - \mathbf{H}^T(\mathbf{Hs}_1^* - \mathbf{y}_1)\rangle \geq 0. \tag{5.113}$$

We finally get the result by switching $\mathbf{H}^T$ to the other side and by using the Cauchy-Schwartz inequality, which leads to

$$\|\mathbf{H}(\mathbf{s}_1^* - \mathbf{s}_2^*)\|_2\|\mathbf{y}_1 - \mathbf{y}_2\|_2 \geq \langle \mathbf{H}(\mathbf{s}_1^* - \mathbf{s}_2^*), \mathbf{y}_1 - \mathbf{y}_2\rangle \geq \|\mathbf{H}(\mathbf{s}_1^* - \mathbf{s}_2^*)\|_2^2. \tag{5.114}$$

Proof of Proposition 5.2:

We show the relation between the difference of the kth iterate of PnP-FBS and the difference of its starting points using the fact that the matrix $\mathbf{I} - \alpha\mathbf{H}^T\mathbf{H}$ has a spectral norm of one when α has an appropriate value. The modulus is

$$\begin{aligned}\|\mathbf{s}_1^k - \mathbf{s}_2^k\|_2 &= \|\mathrm{D}(\mathbf{s}_1^{k-1} - \alpha\mathbf{H}^T(\mathbf{H}\mathbf{s}_1^{k-1} - \mathbf{y}_1) - \mathrm{D}(\mathbf{s}_2^{k-1} - \alpha\mathbf{H}^T(\mathbf{H}\mathbf{s}_2^{k-1} - \mathbf{y}_2)\|_2 \\ &\leq K\|(\mathbf{I} - \alpha\mathbf{H}^T\mathbf{H})(\mathbf{s}_1^{k-1} - \mathbf{s}_2^{k-1}) - \alpha\mathbf{H}^T(\mathbf{y}_1 - \mathbf{y}_2)\|_2 \\ &\leq K\|\mathbf{s}_1^{k-1} - \mathbf{s}_2^{k-1}\|_2 + \alpha K\|\mathbf{H}\|\|\mathbf{y}_1 - \mathbf{y}_2\|_2 \\ &\leq K^2\|\mathbf{s}_1^{k-2} - \mathbf{s}_2^{k-2}\|_2 + \alpha\|\mathbf{H}\|(K + K^2)\|\mathbf{y}_1 - \mathbf{y}_2\|_2 \\ &\leq K^k\|\mathbf{s}_1^0 - \mathbf{s}_2^0\|_2 + \alpha\|\mathbf{H}\|\|\mathbf{y}_1 - \mathbf{y}_2\|_2\sum_{n=1}^{k} K^n. \end{aligned} \tag{5.115}$$

Taking the limit $k \to \infty$, we get that

$$\|\mathbf{s}_1^* - \mathbf{s}_2^*\|_2 \leq \frac{\alpha\|\mathbf{H}\|K}{1-K}\|\mathbf{y}_1 - \mathbf{y}_2\|_2. \tag{5.116}$$

5.4.5 Proof of Proposition 5.3

We show that the four activation functions can be expressed in terms of each other on compact sets without violating the 2-norm weight constraints. Choose B such that $x + B > 0$ for all x in the compact set and any pre-activation in the network.

AV as Expressive as PReLU

We can express AV using PReLU with $a = -1$. For the other direction, we have that

$$\begin{aligned}&\mathrm{PReLU}_a(x) \\ &= \left[\sqrt{(1+a)/2} \; -\sqrt{(1-a)/2}\right] \mathrm{AV}\left(\begin{bmatrix}\sqrt{(1+a)/2} \\ \sqrt{(1-a)/2}\end{bmatrix} x + \begin{bmatrix}\sqrt{(1+a)/2}B \\ 0\end{bmatrix}\right) - \frac{1+a}{2B}.\end{aligned} \tag{5.117}$$

AV as Expressive as GS

This was already proven in [212], but we include the expressions for the sake of completeness. It holds that

$$\begin{bmatrix}\max(x_1) \\ \min(x_2)\end{bmatrix} = \mathbf{M}\,\mathrm{AV}\left(\mathbf{M}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} + \begin{bmatrix}B \\ 0\end{bmatrix}\right) - \begin{bmatrix}\sqrt{2}B \\ 0\end{bmatrix}, \tag{5.118}$$

where

$$\mathbf{M} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1 \\ 1 & -1\end{bmatrix}. \tag{5.119}$$

For the reverse direction, we have that

$$\mathrm{AV}(x) = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \mathrm{MaxMin}\left(\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} x \right). \tag{5.120}$$

GS as Expressive as HH

For $\mathbf{v} = \frac{1}{\sqrt{2}}(1, -1)$ we have that $\mathrm{HH}_{\mathbf{v}} = \mathrm{MaxMin}$. Further, we can also express $\mathrm{HH}_{\mathbf{v}}$ using MaxMin as

$$\mathrm{HH}_{\mathbf{v}}(\mathbf{z}) = \mathbf{R}(\mathbf{v})\,\mathrm{MaxMin}\Big(\mathbf{R}(\mathbf{v})^T \mathbf{z}\Big), \tag{5.121}$$

where $\mathbf{R}(\mathbf{v})$ is the rotation matrix

$$\mathbf{R}(\mathbf{v}) = \begin{bmatrix} \cos\gamma(v_1, v_2) & -\sin\gamma(v_1, v_2) \\ \sin\gamma(v_1, v_2) & \cos\gamma(v_1, v_2) \end{bmatrix} \text{ with } \gamma(v_1, v_2) = \frac{\pi}{4} + 2\arctan\frac{v_2}{1+v_1}. \tag{5.122}$$

■

5.4.6 Properties of $\mathrm{P}_{\mathrm{Lip}}$

1. The Least-Square Projection onto $\{\mathbf{c} \in \mathbb{R}^K : \|\mathbf{D}\mathbf{c}\|_\infty \leq T\}$ Preserves the Mean

Let $\mathbf{x} \in \mathbb{R}^K$ and $\mathbf{y} \in \{\mathbf{c} \in \mathbb{R}^K : \|\mathbf{D}\mathbf{c}\|_\infty \leq T\}$ and $\mathbf{x} = \bar{\mathbf{x}} + \mu_x \mathbf{1}$, $\mathbf{y} = \bar{\mathbf{y}} + \mu_y \mathbf{1}$, where $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ have zero mean. It holds that

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = \|\bar{\mathbf{x}} - \bar{\mathbf{y}} + \mathbf{1}(\mu_x - \mu_y)\|_2^2 = \|\bar{\mathbf{x}} - \bar{\mathbf{y}}\|_2^2 + (\mu_x - \mu_y)^2 K. \tag{5.123}$$

Hence, we can add $(\mu_x - \mu_y)\mathbf{1}$ to $\mathbf{y}$ and decrease the distance without violating the constraints.

2. $\mathrm{P}_{\mathrm{Lip}}$ Maps $\mathbb{R}^K$ to $\{\mathbf{x} \in \mathbb{R}^K : \|\mathbf{D}\mathbf{x}\|_\infty \leq T\}$

We have, for any $\mathbf{c} \in \mathbb{R}^K$, that

$$\|\mathbf{D}\mathrm{P}_{\mathrm{Lip}}(\mathbf{c})\|_\infty = \|\mathbf{D}\mathbf{D}^\dagger \,\mathrm{Clip}_{[-T,T]}(\mathbf{D}\mathbf{c}) + \mathbf{D}\mathbf{1}\frac{1}{K}\sum_{k=1}^K c_k\|_\infty = \|\,\mathrm{Clip}_{[-T,T]}(\mathbf{D}\mathbf{c})\|_\infty \leq T. \tag{5.124}$$

Here, we used the fact that $\mathbf{D}\mathbf{D}^\dagger$ is the identity matrix in $\mathbb{R}^{K-1,K-1}$ and that $\mathbf{D}\mathbf{1}$ is equal to the zero vector in $\mathbb{R}^K$.

3. $\mathrm{P}_{\mathrm{Lip}}$ is a Projection

Using the same properties as above, it holds that

$$\begin{aligned}
\mathrm{P_{Lip}}(\mathrm{P_{Lip}}(\mathbf{c})) &= \mathbf{D}^\dagger \operatorname{Clip}_{[-T,T]}(\mathbf{D}\mathbf{D}^\dagger \operatorname{Clip}_{[-T,T]}(\mathbf{Dc}) + \mathbf{D1}\frac{1}{K}\sum_{k=1}^{K} c_k) + \mathbf{1}\frac{1}{K}\sum_{k=1}^{K} c_k \\
&= \mathbf{D}^\dagger \operatorname{Clip}_{[-T,T]}(\operatorname{Clip}_{[-T,T]}(\mathbf{Dc})) + \mathbf{1}\frac{1}{K}\sum_{k=1}^{K} c_k \\
&= \mathbf{D}^\dagger \operatorname{Clip}_{[-T,T]}(\mathbf{Dc}) + \mathbf{1}\frac{1}{K}\sum_{k=1}^{K} c_k = \mathrm{P_{Lip}}(\mathbf{c}). \qquad (5.125)
\end{aligned}$$

4. $\mathrm{P_{Lip}}$ 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

$$\frac{1}{K}\mathbf{1}^T\mathrm{P_{Lip}}(\mathbf{c}) = \frac{1}{K}\mathbf{1}^T\mathbf{D}^\dagger \operatorname{Clip}_{[-T,T]}(\mathbf{Dc}) + \mathbf{1}^T\mathbf{1}\frac{1}{K^2}\sum_{k=1}^{K} c_k = \frac{1}{K}\sum_{k=1}^{K} c_k. \qquad (5.126)$$

5. $\mathrm{P_{Lip}}$ is Differentiable Almost Everywhere with Respect to c

The $\operatorname{Clip}_{[-T,T]}$ function is differentiable everywhere except at T and $-T$. Therefore, the operation $\mathbf{D}^\dagger \operatorname{Clip}_{[-T,T]}(\mathbf{Dc})$ is differentiable everywhere except on

$$S = \bigcup_{k=1}^{K-1} \Big\{\mathbf{x} \in \mathbb{R}^K : |(\mathbf{Dx})_k| = T\Big\}. \qquad (5.127)$$

The set S is a union of $2(K-1)$ hyperplanes with dimension $K-1$. Hence, it has measure zero in $\mathbb{R}^K$.

5.4.7 Hyperparameter Tuning: Solving Inverse Problems

The parameters τ and μ used in (5.85) can be tuned with a coarse-to-fine approach. Given the performance on the 3×3 grid $\{(\gamma_\tau)^{-1}\tau, \tau, \gamma_\tau\tau\} \times \{(\gamma_\mu)^{-1}\mu, \mu, \gamma_\mu\mu\}$, we identify the best values τ^* and μ^* on this subset and move on to the next iteration as follows:

- if $\tau^* = \tau$, we refine the search grid by reducing γ_μ to $(\gamma_\mu)^\zeta$, $\zeta < 1$;
- otherwise, τ is updated to τ^*.

A similar update is performed for the scaling parameter. The search is terminated when both γ_τ and γ_μ are smaller than a threshold, typically, 1.01. In practice, we initialized

$\gamma_\tau = \gamma_\mu = 4$ and set $\zeta = 0.5$. The method usually requires between 50 and 100 evaluations on tuples (τ, μ) on the validation set before it terminates. The proposed approach is predicated on the observation that the optimization landscape in the (τ, μ) domain is typically well-behaved. The same principles apply to tune a single hyperparameter, as found in the TV method. Let us remark that the performances were 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.

6 Deep Generative Priors for Nonlinear Inverse Problems

[1]In this chapter, we show how we can leverage the power of deep generative models as image priors to develop a Bayesian inference pipeline that produces high quality reconstructions together with uncertainty maps. To the best of our knowledge, this is one of the first deployments of such techniques for the resolution of nonlinear inverse problems.

6.1 Contributions

Here, we present a Bayesian framework to solve a broad class of nonlinear inverse problems, where the prior knowledge about the image of interest is specified through a trained deep latent variable generative model such as a GAN or a VAE. Our contributions are listed below.

- We develop a method based on the Metropolis-adjusted Langevin algorithm (MALA) [265, 266] to sample from the posterior distribution for the class of nonlinear inverse problems where the forward model has a neural-network-like structure. This class includes a wide variety of practical imaging modalities. We show that the structure of the forward model and the low-dimensional latent space of the generative prior enable tractable Bayesian inference.
- We introduce the concept of augmented generative models. This is motivated by the observation that the above-mentioned deep generative models are easier to train when the dataset consists of images with the same range of pixel values. Unfortunately, such models are not well-matched to imaging modalities where one is interested in extracting the precise value of objects rather than merely visualizing contrast. Our proposed augmented models provide us with a simple but effective way of dealing with quantitative data.

[1]This chapter is based on our work [264].

- We illustrate the advantages of the proposed reconstruction framework through numerical experiments for two nonlinear imaging modalities: phase retrieval and optical diffraction tomography.

The chapter is organized as follows: In Section 6.2, we discuss the structure of the forward model for our nonlinear inverse problems. We detail the Bayesian reconstruction framework in Section 6.3. There, we introduce augmented generative models and we explain our posterior-sampling scheme. We present our experimental results in Section 6.4.

6.2 Nonlinear Inverse Problems and Forward Models

In this section, we start by describing the class of nonlinear inverse problems that we are interested in. We then focus on two concrete examples—phase retrieval and optical diffraction tomography—and detail the physical models involved.

6.2.1 Nonlinear Inverse Problems

The objective is to recover an image $\mathbf{s}^\dagger \in \mathbb{R}^K$ from its noisy measurements $\mathbf{y}^\dagger \in \mathbb{C}^M$ given by $\mathbf{y}^\dagger = \mathbf{N}(\mathbf{y}_0^\dagger)$ with

$$\mathbf{y}_0^\dagger = \mathbf{H}(\mathbf{s}^\dagger), \tag{6.1}$$

where $\mathbf{H} : \mathbb{R}^K \to \mathbb{C}^M$ is a nonlinear operator that models the physics of the imaging system and $\mathbf{N} : \mathbb{C}^M \to \mathbb{C}^M$ is an operator that models the corruption of the measurements by noise. In this work, we consider the class of nonlinear forward models $\mathbf{H}$ whose computational structure can be encoded by a directed acyclic graph and thus resembles a neural network.

The Jacobian matrix of $\mathbf{H}$ at any point $\mathbf{x} = (x_1, \dots, x_K) \in \mathbb{R}^K$ is defined as

$$\mathbf{J}_{\mathbf{H}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}[\mathbf{H}(\mathbf{x})]_1 & \cdots & \frac{\partial}{\partial x_K}[\mathbf{H}(\mathbf{x})]_1 \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1}[\mathbf{H}(\mathbf{x})]_M & \cdots & \frac{\partial}{\partial x_K}[\mathbf{H}(\mathbf{x})]_M \end{bmatrix}. \tag{6.2}$$

Gradient-based MCMC methods (see Section 6.3 for a specific example) involve the computation of quantities such as $\mathbf{J}_{\mathbf{H}}^H(\mathbf{x})\mathbf{r}$ for some vectors $\mathbf{x} \in \mathbb{R}^K$, $\mathbf{r} \in \mathbb{C}^M$, and this can be a potential bottleneck. The neural-network-like structure of $\mathbf{H}$ allows us to compute these efficiently using the error backpropagation algorithm. This, in turn, makes Bayesian inference computationally feasible.

The class of nonlinear inverse problems that fit this description is very broad and adaptable to most existing imaging modalities. In principle, it covers all possible inverse problems,

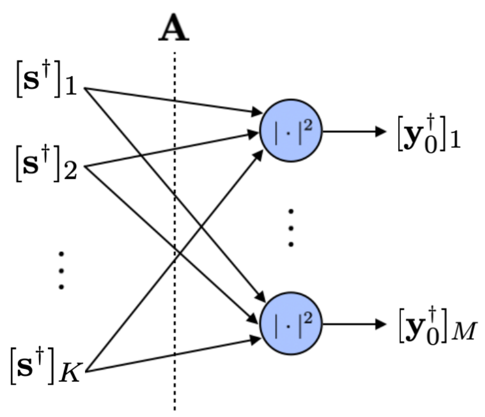


Figure 6.1: The forward model for phase retrieval (6.3) expressed as a one-layer fully-connected neural network with linear weights **A** and quadratic activation functions.

in particular, the linear case is trivially covered. More generally, if sufficient data is available, one can indeed train a neural network to mimic the physics of our forward model. Next, we look at two particular problems that nicely fall within our predefined class.

6.2.2 Phase Retrieval

Phase retrieval [267, 268] is a nonlinear inverse problem that is ubiquitous in computational imaging. It consists in the recovery of a signal from its intensity-only measurements and is a central issue in optics [269, 270], astronomy [271, 272], and computational microscopy [273–276].

In the phase-retrieval problem that we consider in this paper, the noise-free measurements are modeled as

$$\mathbf{y}_0^\dagger = \mathbf{H}_{\mathrm{pr}}(\mathbf{s}^\dagger) = |\mathbf{A}\mathbf{s}^\dagger|^2, \tag{6.3}$$

where $\mathbf{A} : \mathbb{R}^K \to \mathbb{C}^M$ is either the Fourier matrix [269, 276, 277] or some realization of a random matrix with independent and identically distributed (i.i.d.) elements [268, 278, 279], and where $|\cdot|^2$ is a component-wise operator. As shown in Figure 6.1, the forward model in (6.3) can be expressed as a one-layer fully-connected neural network with fixed linear weights **A** and quadratic activation functions.

6.2.3 Optical Diffraction Tomography

In optical diffraction tomography (ODT), the aim is to recover the refractive-index (RI) map of a sample from complex-valued measurements of the scattered fields generated when the sample is probed by a series of tilted incident fields [280]. According to the

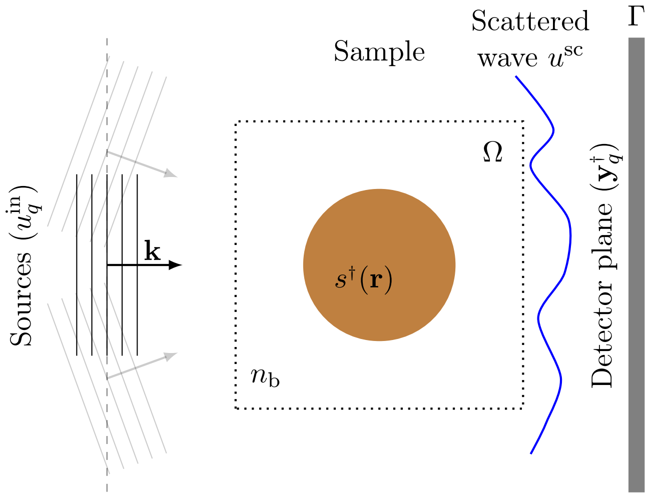

Sample
Scattered wave u^{sc}
Γ
Ω
Sources (u_q^{in})
$\mathbf{k}$
$s^\dagger(\mathbf{r})$
n_b
Detector plane $(\mathbf{y}_q^\dagger)$

Figure 6.2: Optical diffraction tomography. A sample of refractive index $n_\mathrm{b} + s^\dagger(\mathbf{r})$ is immersed in a medium of index n_b and illuminated by an incident plane wave (wave vector $\mathbf{k}$). The interaction of the wave with the object produces scattered waves, which are recorded at the detector plane.

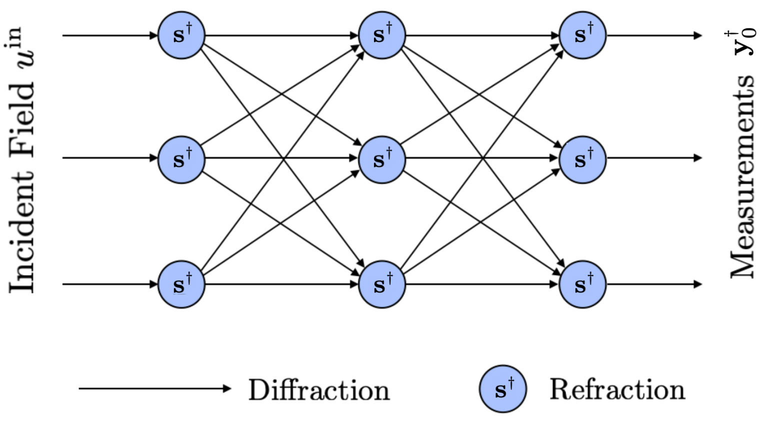
scalar-diffraction theory, the propagation of the incident fields through the sample is governed by the wave equation. While pioneering works relied on linear models to approximate this propagation [280, 281], recent works have significantly improved the quality of RI reconstruction by using more accurate nonlinear models that account for multiple scattering [282]. Here, we look at one such nonlinear model called the beam-propagation method (BPM).

Helmholtz Equation. We consider a sample with a real-valued spatially varying refractive index that is immersed in a medium with constant refractive index n_b, as shown in Figure 6.2. The RI distribution in the region of interest $\Omega = [0, L_\mathrm{x}] \times [0, L_\mathrm{z}]$ is represented as $n(\mathbf{r}) = n_\mathrm{b} + s^\dagger(\mathbf{r})$, where $\mathbf{r} = (x, z)$ and $s^\dagger(\mathbf{r})$ is the RI contrast. The sample is illuminated with an incident plane wave $u^{\mathrm{in}}(\mathbf{r})$ of free-space wavelength λ, whose direction of propagation is specified by the wave vector $\mathbf{k}$. The total field $u(\mathbf{r})$ that results from the interaction between the sample and the incident wave is then recorded at the positions $\{\mathbf{r}_m\}_{m=1}^{M'}$ in the detector plane Γ to yield the complex measurements $\mathbf{y}^\dagger \in \mathbb{C}^{M'}$. The interplay between the total field $u(\mathbf{r})$ at any point in space and the refractive index contrast $\delta n(\mathbf{r})$ is described by the Helmholtz equation

$$\nabla^2 u(\mathbf{r}) + k_0^2 n^2(\mathbf{r}) u(\mathbf{r}) = 0, \tag{6.4}$$

where $k_0 = \frac{2\pi}{\lambda}$.

Beam Propagation Method. For computational purposes, the region of interest Ω



Incident Field u^{in}
$\mathbf{s}^\dagger$
Measurements $\mathbf{y}_0^\dagger$
Diffraction
Refraction

Figure 6.3: The computational structure for BPM resembles a neural network.

is subdivided into an $(N_\text{x} \times N_\text{z})$ array of pixels with sampling steps δ_x and δ_z along the first and second dimension, respectively. The corresponding samples of the RI contrast $s^\dagger(\mathbf{r})$ and total field $u(\mathbf{r})$ are stored in the vectors[2] $\mathbf{s}^\dagger \in \mathbb{R}^K$ and $\mathbf{u} \in \mathbb{C}^K$, respectively, where $K = N_\text{x}N_\text{z}$. Further, let $\mathbf{s}_k^\dagger \in \mathbb{R}^{N_\text{x}}$ and $\mathbf{u}_k \in \mathbb{C}^{N_\text{x}}$ represent the above quantities when restricted to the slice $z = k\delta_\text{z}$.

BPM computes the total field $\mathbf{u}$ in a slice-by-slice manner along the z-axis. For a given incident wave $u^{\text{in}}(\mathbf{r})$ that is propagated over a region larger than Ω, we set the initial conditions as $\mathbf{u}_{-1}(\mathbf{s}_0^\dagger) = \left(u^{\text{in}}(i\delta_\text{x}, -\delta_\text{z})\right)_{i=0}^{N_\text{x}-1} \in \mathbb{C}^{N_\text{x}}$. The total field over Ω is then computed via a series of diffraction and refraction steps

$$\widetilde{\mathbf{u}}_k(\mathbf{s}^\dagger) = \mathbf{u}_{k-1}(\mathbf{s}^\dagger) * \mathbf{h}_{\text{prop}}^{\delta_\text{z}} \qquad \text{(diffraction)} \tag{6.5}$$

$$\mathbf{u}_k(\mathbf{s}^\dagger) = \widetilde{\mathbf{u}}_k(\mathbf{s}_0^\dagger) \odot \mathbf{p}_k(\mathbf{s}^\dagger) \qquad \text{(refraction)}, \tag{6.6}$$

where $k = 0, 1, \dots, (N_\text{z} - 1)$, and the symbols $*$ and $\odot$ stand for convolution and pointwise multiplication, respectively. The convolution kernel $\mathbf{h}_{\text{prop}}^{\delta_\text{z}} \in \mathbb{C}^{N_\text{x}}$ for the diffraction step is characterized in the Fourier domain as

$$\mathcal{F}\left\{\mathbf{h}_{\text{prop}}^{\delta_\text{z}}\right\}(\mathbf{w}_\text{x}) = \mathrm{e}^{\mathrm{j}\delta_\text{z}\left(\sqrt{k_0^2 {n_\text{b}}^2 - \mathbf{w}_\text{x}^2}\right)}, \tag{6.7}$$

where $\mathcal{F}$ denotes the discrete Fourier transform and $\mathbf{w}_\text{x} \in \mathbb{R}^{N_\text{x}}$ is the frequency variable. The subsequent refraction step involves a pointwise multiplication with the phase mask

$$\mathbf{p}_k(\mathbf{s}^\dagger) = \mathrm{e}^{\mathrm{j}k_0\delta_\text{z}\mathbf{s}_k^\dagger}. \tag{6.8}$$

Finally, we define an operator $\mathbf{R} : \mathbb{C}^{N_\text{x}} \mapsto \mathbb{C}^{M'}$ that propagates $\mathbf{u}_{N_\text{z}-1}(\mathbf{s}^\dagger)$ to the detector

[2]Since the total field $u(\mathbf{r})$ depends on the RI contrast $s^\dagger(\mathbf{r})$, we also refer to its discretized version as $\mathbf{u}(\mathbf{s}^\dagger)$.

plane Γ and restricts it to the sensor positions to give us the measurements $\mathbf{y}^\dagger \in \mathbb{C}^{M'}$. Thus, for a given incident wave u^{in}, our noise-free nonlinear BPM forward model is of the form

$$\mathbf{y}_0^\dagger = \mathbf{H}_{\text{bpm}}(\mathbf{s}^\dagger; u^{\text{in}}) = \mathbf{R}\Big(\mathbf{u}_{N_z-1}(\mathbf{s}^\dagger)\Big). \tag{6.9}$$

In Figure 6.3, we show the implementation of $\mathbf{H}_{\text{bpm}}$ as a directed acyclic graph.

Complete Forward Model. We assume that the sample is illuminated with Q incident plane waves $\{u_q^{\text{in}}\}_{q\in\{1,\dots,Q\}}$ and that the corresponding measurements are $\{\mathbf{y}_q^\dagger \in \mathbb{C}^{M'}\}_{q\in\{1,\dots,Q\}}$. These measurements are related to the RI contrast $\mathbf{s}^\dagger$ of the sample through the BPM forward model in (6.9). We define a stacked measurement vector as $\mathbf{y}^\dagger = (\mathbf{y}_1^\dagger, \dots, \mathbf{y}_Q^\dagger) \in \mathbb{R}^M$ $(M = QM')$. This allows us to rewrite the complete forward model in the form of (6.1), where the operator $\mathbf{H}$ consists of the application of $\mathbf{H}_{\text{bpm}}$ with all the illuminations and the concatenation of the outputs into a single vector.

6.3 Bayesian Reconstruction Framework

We now present our reconstruction framework that is based on Bayesian statistics for solving the generic nonlinear inverse problem described in Section 6.2.1. Let Y and S be the random vectors[3] associated with the measurements and the signal, respectively. As in Chapter 2, the statistical model for the measurement noise is specified via the conditional distribution of $\mathsf{Y}|\mathsf{S} = \mathbf{s}$, where $\mathbf{s} \in \mathbb{R}^K$. In this section, we first discuss the prior distribution of S, which, in our framework, is defined through a deep generative model, followed by the posterior distribution of $\mathsf{S}|\mathsf{Y} = \mathbf{y}^\dagger$. Finally, we detail a MCMC scheme to generate samples from the posterior distribution. This allows us to perform inference by computing point estimates and the uncertainties associated with them.

6.3.1 Prior Distribution

The choice of the distribution $\mathbb{P}_{\mathsf{S}}$ reflects our prior knowledge about the image of interest. In classical Bayesian methods, $\mathbb{P}_{\mathsf{S}}$ is generally chosen from a family of distributions with closed-form analytical expressions for their pdfs such that it fits the characteristics of the image and also allows for efficient inference. Popular examples include the Gaussian and Markovian models. In our framework, we instead propose to leverage the power of neural networks to define a data-driven prior distribution.

We assume that we have access to a dataset that contains sample images from the true (but unknown) probability distribution $\mathbb{P}_{\text{image}}$ of our image of interest. The idea then is to approximate $\mathbb{P}_{\text{image}}$ with $\mathbb{P}_{\mathsf{S}}$ as defined by a deep generative model. More specifically,

[3]In this chapter, for a given random vector V, we will denote its probability distribution by $\mathbb{P}_{\mathsf{V}}$ (which is a measure) and its pdf with respect to the Lebesgue measure (if $\mathbb{P}_{\mathsf{V}}$ admits one) by p_{V}.

we consider deep latent variable generative models consisting of a generator network $\mathrm{G} : \mathbb{R}^d \to \mathbb{R}^K$ ($d \ll K$) that maps a low-dimensional latent space to the high-dimensional image space. For such a model, we have $\mathbf{S} = \mathrm{G}(\mathbf{Z})$, where $\mathbf{Z}$ is a random vector that takes values in $\mathbb{R}^d$ with a pdf $p_{\mathbf{Z}}$ (typically a Gaussian or uniform distribution). If this model is properly trained, the resulting $\mathbb{P}_{\mathbf{S}}$ (which is the pushforward of $\mathbb{P}_{\mathbf{Z}}$ through the mapping G) is close to $\mathbb{P}_{\text{image}}$ and the images generated by it are statistically similar to the ones in the dataset.

In our experiments (see Section 6.4), we use the well-known Wasserstein GANs (WGANs) [220] for our data-driven prior. We provide a brief description of WGANs in Appendix 6.6.2

Augmented Deep Generative Priors. The training of deep generative models such as GANs requires large amounts of data and is a challenging task in general. Over the past few years, there have been several proposals for performance improvements that have led to the development of better training schemes and network architectures. Most existing works use normalized datasets, where each image has the same range of pixel values. However, this is not suitable if we wish to use such models as priors in quantitative imaging (*e.g.,* ODT). In these modalities, it is important to recover the actual values of the object (image) as compared to only the contrast. Thus, we require our generative model to be able to output images with different ranges of pixel values.

While performing our experiments, we observed that the training of high-quality WGANs on unnormalized datasets was non-trivial. We propose a simple effective workaround, which simplifies the training and allows us to build models that generate images with different ranges. We define an augmented generative model $\mathrm{G}_h : \mathbb{R}^{d+1} \to \mathbb{R}^K$ ($d \ll K$) that consists of a (standard) generative network $\mathrm{G} : \mathbb{R}^d \to \mathbb{R}^K$ trained on a normalized dataset and a deterministic function $h : \mathbb{R} \to \mathbb{R}$. Here, the latent (random) vector $\mathbf{Z} = (\mathbf{Z}_1, \mathrm{Z}_2)$ takes values in $\mathbb{R}^{d+1}$ and has two independent components $\mathbf{Z}_1$ (that takes values in $\mathbb{R}^d$) and Z_2 (that takes values in $\mathbb{R}$) with pdfs $p_{\mathbf{Z}_1}$ and p_{Z_2}, respectively. The output image of this model is $\mathbf{S} = \mathrm{G}_h(\mathbf{Z}) = h(\mathrm{Z}_2)\mathrm{G}(\mathbf{Z}_1)$, where the term $\mathrm{G}(\mathbf{Z}_1)$ represents it details or contrast, and the term $h(\mathrm{Z}_2)$ represents it scaling factor.

Since G is now required to only produce images with the same range, we can rely on existing GANs to obtain high-quality models. Moreover, the distribution of the scaling factor can be easily controlled by carefully choosing the distribution p_{Z_2} and the function h.

6.3.2 Posterior Distribution

Since our prior distribution $\mathbb{P}_{\mathbf{S}}$ is defined by a pre-trained augmented deep generative model $\mathrm{G}_h : \mathbb{R}^{d+1} \to \mathbb{R}^K, \mathbf{z} \mapsto \mathrm{G}_h(\mathbf{z})$ with $p_{\mathbf{Z}}(\mathbf{z}) = p_{\mathbf{Z}_1}(\mathbf{z}_1)p_{\mathrm{Z}_2}(z_2)$ for any $\mathbf{z} = (\mathbf{z}_1, z_2) \in \mathbb{R}^{d+1}$, our posterior distribution $\mathbb{P}_{\mathbf{S}|\mathbf{Y}=\mathbf{y}^\dagger}$ is given by the push-forward of the posterior distribution

$\mathbb{P}_{\mathbf{Z}|\mathbf{Y}=\mathbf{y}^\dagger}$ of the latent vector through the mapping G_h. The pdf for $\mathbb{P}_{\mathbf{Z}|\mathbf{Y}=\mathbf{y}^\dagger}$ can be written as

$$p_{\mathbf{Z}|\mathbf{Y}}(\mathbf{z}|\mathbf{y}^\dagger) = \frac{p_{\mathbf{Y}|\mathbf{Z}}(\mathbf{y}^\dagger|\mathbf{z})p_{\mathbf{Z}}(\mathbf{z})}{\int_{\mathbb{R}^{d+1}} p_{\mathbf{Y}|\mathbf{Z}}(\mathbf{y}^\dagger|\tilde{\mathbf{z}})p_{\mathbf{Z}}(\tilde{\mathbf{z}})\mathrm{d}\tilde{\mathbf{z}}},$$

where $p_{\mathbf{Y}|\mathbf{Z}}(\cdot|\mathbf{z}) = p_{\mathbf{Y}|\mathbf{S}}(\cdot|\mathbf{s} = \mathrm{G}_h(\mathbf{z}))$.

A Bayesian inverse problem is said to be well-posed in some metric on the space of probability measures if its solution (the posterior distribution) exists, is unique, and is continuous with respect to the measurements for the chosen metric [283]. Depending on the metric, the well-posedness of the Bayesian inverse problem ensures continuity of posterior expectations of appropriate quantities of interest. Based on the work in [283], we can show that for the AWGN model, our Bayesian problem is well-posed in the Prokhorov, total-variation and Hellinger distances. Moreover, our problem is well-posed in the Wasserstein distance if $p_{\mathbf{Z}}$ satisfies a finite-moment-like condition. By using a result from [78], we can also show the existence of the moments of our posterior distribution under mild conditions on $p_{\mathbf{Z}}$ and G_h. We provide the details regarding these properties in Appendix 6.6.1.

6.3.3 Sampling from the Posterior Distribution

The proposed framework allows one to draw samples in the low-dimensional latent space instead of the high-dimensional image space directly. Specifically, if we generate samples $\{\mathbf{z}^{\dagger(t)}\}_{t=1}^T$ from $p_{\mathbf{Z}|\mathbf{Y}}(\cdot|\mathbf{y}^\dagger)$, then the images $\{\mathbf{s}^{\dagger(t)} = \mathrm{G}_h(\mathbf{z}^{\dagger(t)})\}_{t=1}^T$ are samples from $\mathbb{P}_{\mathbf{S}|\mathbf{Y}=\mathbf{y}^\dagger}$.

In this work, we use the MCMC method called Metropolis-adjusted Langevin algorithm (MALA) [265, 266] to sample from $p_{\mathbf{Z}|\mathbf{Y}}(\cdot|\mathbf{y}^\dagger)$. Given a sample $\mathbf{z}^{\dagger(t)}$, it generates $\mathbf{z}^{\dagger(t+1)}$ in two steps. In the first step, we construct a proposal $\tilde{\mathbf{z}}^{\dagger(t+1)}$ for the new sample according to

$$\tilde{\mathbf{z}}^{\dagger(t+1)} = \mathbf{z}^{\dagger(t)} + \eta\nabla_{\mathbf{z}} \log p_{\mathbf{Z}|\mathbf{Y}}(\mathbf{z}^{\dagger(t)}|\mathbf{y}^\dagger) + \sqrt{2\eta}\boldsymbol{\zeta}, \tag{6.10}$$

where $\boldsymbol{\zeta}$ is drawn from the standard multivariate Gaussian distribution and $\eta \in \mathbb{R}_+$ is a fixed step-size. In the second step, the proposal $\tilde{\mathbf{z}}^{\dagger(t+1)}$ is either accepted or rejected, the acceptance probability being

$$\alpha = \min\left\{1, \frac{p_{\mathbf{Z}|\mathbf{Y}}(\tilde{\mathbf{z}}^{\dagger(t+1)}|\mathbf{y}^\dagger)q_{\mathbf{y}^\dagger}(\mathbf{z}^{\dagger(t)}|\tilde{\mathbf{z}}^{\dagger(t+1)})}{p_{\mathbf{Z}|\mathbf{Y}}(\mathbf{z}^{\dagger(t)}|\mathbf{y}^\dagger)q_{\mathbf{y}^\dagger}(\tilde{\mathbf{z}}^{\dagger(t+1)}|\mathbf{z}^{\dagger(t)})}\right\}, \tag{6.11}$$

where $q_{\mathbf{y}}(\bar{\mathbf{z}}|\underline{\mathbf{z}}) = \exp\Big(-\frac{1}{4\eta}\|\bar{\mathbf{z}} - \underline{\mathbf{z}} - \eta\nabla_{\mathbf{z}}\log p_{\mathbf{Z}|\mathbf{Y}}(\underline{\mathbf{z}}|\mathbf{y})\|_2^2\Big)$ for any $\mathbf{y} \in \mathbb{C}^M$ and $\bar{\mathbf{z}}, \underline{\mathbf{z}} \in \mathbb{R}^{d+1}$. If the proposal is accepted, then we set $\mathbf{z}^{\dagger(t+1)} = \tilde{\mathbf{z}}^{\dagger(t+1)}$; otherwise, $\mathbf{z}^{\dagger(t+1)} = \mathbf{z}^{\dagger(t)}$. One advantage of MALA is that it uses the gradient of the (log) target distribution to construct more probable proposals. In doing so, it explores the target distribution faster than some other MCMC methods such as the well-known random walk Metropolis-Hastings algorithm [284].

The major computational bottleneck in MALA is the computation of the gradient term $\nabla_{\mathbf{z}} \log p_{\mathsf{Z}|\mathsf{Y}}(\cdot|\mathbf{y}^\dagger)$ as it involves terms such as $\mathbf{J}_{\mathbf{H}}^H(\mathbf{x}_1)\mathbf{r}_1$ and $\mathbf{J}_{\mathrm{G}_h}^H(\mathbf{x}_2)\mathbf{r}_2$, where $\mathbf{x}_1 \in \mathbb{R}^K$, $\mathbf{r}_1 \in \mathbb{C}^M$, $\mathbf{x}_2 \in \mathbb{R}^{d+1}$, and $\mathbf{r}_2 \in \mathbb{R}^K$. For instance, if we assume an AWGN model with variance σ^2 and that p_{Z} is the standard mutivariate Gaussian distribution, then $p_{\mathsf{Z}|\mathsf{Y}}$ can be written as

$$p_{\mathsf{Z}|\mathsf{Y}}(\mathbf{z}|\mathbf{y}^\dagger) = \frac{1}{C} \exp\left(- \frac{\|\mathbf{y}^\dagger - \mathbf{H}(\mathrm{G}_h(\mathbf{z}))\|_2^2}{2\sigma^2} - \frac{\|\mathbf{z}\|_2^2}{2} \right), \tag{6.12}$$

where C is the normalization factor. In this case, the gradient term is

$$\nabla_{\mathbf{z}} \log p_{\mathsf{Z}|\mathsf{Y}}(\mathbf{z}|\mathbf{y}^\dagger) = -\frac{\mathbf{J}_{\mathrm{G}_h}^H(\mathbf{z})\mathbf{J}_{\mathbf{H}}^H(\mathrm{G}_h(\mathbf{z}))(\mathbf{y}^\dagger - \mathbf{H}(\mathrm{G}_h(\mathbf{z})))}{\sigma^2} - \mathbf{z}. \tag{6.13}$$

Since G_h is a neural network and $\mathbf{H}$ has a neural-network-like structure, we then compute $\nabla_{\mathbf{z}} \log p_{\mathsf{Z}|\mathsf{Y}}$ efficiently using an error backpropagation algorithm.

Once we have obtained the samples $\{\mathbf{z}^{\dagger(t)}\}_{t=1}^T$ from $p_{\mathsf{Z}|\mathsf{Y}}(\cdot|\mathbf{y}^\dagger)$, we transform them to get the samples $\{\mathrm{G}_h(\mathbf{z}^{\dagger(t)})\}_{t=1}^T$ from $\mathbb{P}_{\mathsf{S}|\mathsf{Y}=\mathbf{y}^\dagger}$ and use them to perform inference. Specifically, we approximate any integral of the form $\int_{\mathbb{R}^K} f(\mathbf{s}) p_{\mathsf{S}|\mathsf{Y}}(\mathbf{s}|\mathbf{y}^\dagger)\mathrm{d}\mathbf{s}$, where $f : \mathbb{R}^K \to \mathbb{R}$ is a real-valued function, by its empirical estimate $\frac{1}{T}\sum_{t=1}^T f(\mathrm{G}_h(\mathbf{z}^{\dagger(t)}))$.

In practice, we discard some of the samples generated at the beginning of the chain to correct for their bias. This “burn-in” period can often be shortened by choosing a suitable starting point for the chain. We propose to initialize MALA with

$$\mathbf{z}_{\mathrm{init}}^*(\mathbf{y}^\dagger) = \arg\min_{\mathbf{z} \in \mathbb{R}^{d+1}} \|\mathbf{s}_{\mathrm{init}}^*(\mathbf{y}^\dagger) - \mathrm{G}_h(\mathbf{z})\|_2^2, \tag{6.14}$$

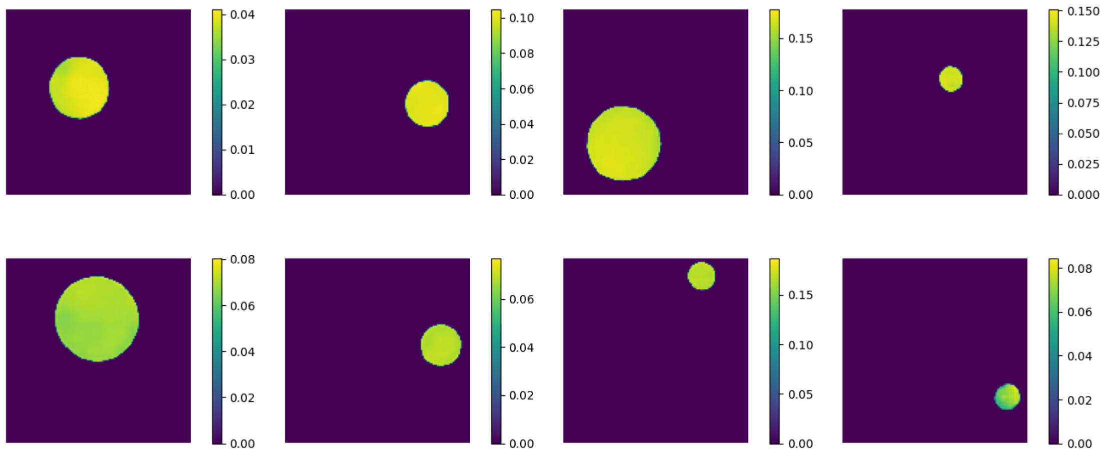
where $\mathbf{s}_{\mathrm{init}}^*(\mathbf{y}^\dagger)$ is a low-quality estimate obtained by using some fast classical reconstruction algorithm.

6.4 Results and Discussion

In this section, we show the benefits of our neural-network-based Bayesian reconstruction framework by applying it to both phase retrieval and optical diffraction tomography.

6.4.1 Augmented Generative Models

In our first experiment, we highlight the importance of the proposed augmented generative models. We consider the task of training WGAN models on synthetic datasets consisting of (128×128) images, where each image contains a constant-valued disc and its background


(a) WGAN

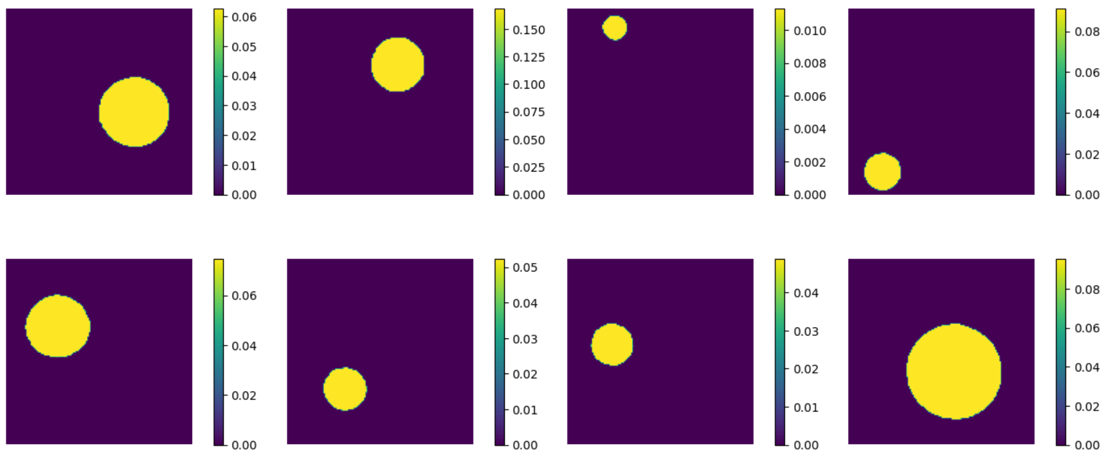

(b) Augmented WGAN

Figure 6.4: Samples generated by trained models.

pixels are zero-valued. The coordinates (x, y) of the center of the disc, its radius r (in pixels), and its constant-intensity value v follow the uniform distributions $U_{(10,115)}$, $U_{(10,115)}$, $U_{[8,35]}$, and $U_{(0,0.2]}$, respectively. The aforementioned parameters implicitly define the probability distribution $\mathbb{P}_{\text{data}}$ that we wish to approximate using WGANs.

We qualitatively compare the performance of two models. The first model is a WGAN trained on 50,000 images sampled from $\mathbb{P}_{\text{data}}$. In this case, the distribution $p_{\mathbf{z}}$ for the latent variable is chosen to be the standard multivariate Gaussian distribution. The second model is an augmented WGAN, where the WGAN component is trained on a normalized dataset with 50,000 images. Thus, we first sample 50,000 images from $\mathbb{P}_{\text{data}}$ and we then normalize each of them such that the value of the disc is one. The distributions $p_{\mathbf{z}_1}$ and $p_{\mathbf{z}_2}$ are chosen to be standard Gaussian distributions as well, and the function h is

$$h(x) = \frac{0.2}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-\frac{t^2}{2}}\,\mathrm{d}t. \tag{6.15}$$

This choice of h and p_{z_2} ensures that the scaling factor of the augmented WGAN follows the uniform distribution $U_{(0,0.2]}$. For both the models, we use the generator and critic network architectures described in Appendix 6.6.3. The WGAN is trained for 2500 epochs while the augmented WGAN is trained for 1250 epochs using RMSProp optimizers with a learning rate of 5×10^{-5} and a batch size of 64. The parameters λ_{gp} and n_{critic} (refer to Appendix 6.6.2) are set as 10 and 5, respectively.

In Figure 6.4, we present typical samples generated by the two models. We observe that the augmented WGAN, unlike the WGAN, is able to produce sharp constant-valued discs.

6.4.2 Phase Retrieval

Next, we look at the phase-retrieval problem. We present two examples where the ground-truth images are taken from the MNIST [285] and Fashion-MNIST [286] testing datasets. In both cases, the measurements $\mathbf{y}^{\dagger} \in \mathbb{N}^M$ are simulated according to (6.3) with a Poisson-noise model, where $\mathbf{A}$ is one realization of a random matrix with i.i.d. entries from a zero-mean Gaussian distribution with variance $\sigma_{\mathbf{A}}^2$.

MNIST

The MNIST dataset contains (28×28) images of handwritten digits. The ground-truth image (Figure 6.5) is first normalized to have values in the range $[0, 1]$ and is then multiplied by a factor α which is picked uniformly at random from $(0, 0.5]$.

In this case, the WGAN component of our augmented model G_h is trained on the normalized MNIST training dataset which contains 50,000 images with values in the range $[0, 1]$. The distributions $p_{\mathbf{z}_1}$ and p_{z_2} are standard Gaussian distributions and the function h is

$$h(x) = \frac{0.5}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-\frac{t^2}{2}}\,\mathrm{d}t. \tag{6.16}$$

The architectures for the generator and critic networks can be found in Appendix 6.6.3. The WGAN is trained for 2000 epochs using ADAM optimizers [194] with a learning rate of 2×10^{-4}, hyperparameters $(\beta_1, \beta_2) = (0.5, 0.999)$, and a batch size of 64. The parameters λ_{gp} and n_{critic} are set as 10 and 5, respectively.

Fashion-MNIST

The Fashion-MNIST dataset consists of (28×28) grayscale images of different fashion products. Our ground-truth image from this dataset is shown in Figure 6.6.

Here, the WGAN for our augmented deep generative prior is trained on the normalized Fashion-MNIST training dataset. It contains 60,000 images whose values lie in the range $[0, 1]$. The distributions $p_{\mathbf{z}_1}$ and $p_{\mathbf{z}_2}$ are taken as standard Gaussian distributions while the function h is

$$h(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-\frac{t^2}{2}} \mathrm{d}t. \tag{6.17}$$

We provide the architectures for the generator and critic networks in Appendix 6.6.3. The WGAN is trained for 2250 epochs using ADAM optimizers with a learning rate of 2×10^{-4}, hyperparameters $(\beta_1, \beta_2) = (0.5, 0.999)$, and a batch size of 64. The parameters λ_{gp} and n_{critic} are set as 10 and 5, respectively.

Methods

As discussed in Section 6.3.3, we draw samples from the posterior distribution using MALA. The estimator $\mathbf{s}^*_{\text{init}}(\mathbf{y}^\dagger)$ that we use for initializing the chain is

$$\mathbf{s}^*_{\text{init}}(\mathbf{y}^\dagger) = \underset{\mathbf{s}\in\mathbb{R}^K}{\arg\min} \left(\sum_{m=1}^{M} \left(-[\mathbf{y}^\dagger]_m \log \left(\left[|\mathbf{A}\mathbf{s}|^2\right]_m \right) + \left[|\mathbf{A}\mathbf{s}|^2\right]_m \right) \right.$$
$$\left. + \ \tau \|\boldsymbol{\nabla}\mathbf{s}\|_{2,2}^2 \ + \ i_+(\mathbf{s}) \right). \tag{6.18}$$

There, $\boldsymbol{\nabla} : \mathbb{R}^K \to \mathbb{R}^{K\times 2}$ is the gradient operator, $\|\cdot\|_{p,q}$ is the (ℓ_p, ℓ_q)-mixed norm defined as

$$\|\mathbf{x}\|_{p,q} \triangleq \left(\sum_{u=1}^{U} \left(\sum_{v=1}^{V} \left([\mathbf{x}]_{u,v} \right)^p \right)^{q/p} \right)^{1/q} \quad \forall \mathbf{x} \in \mathbb{R}^{U\times V}, \tag{6.19}$$

$\tau \in \mathbb{R}_+$ is the regularization parameter and the functional i_+ given by

$$i_+(\mathbf{s}) = \begin{cases} 0, & \mathbf{s} \in \mathbb{R}^K_+ \\ +\infty, & \text{otherwise} \end{cases} \tag{6.20}$$

enforces the non-negativity constraint on the solution. The data-fidelity term in (6.18) corresponds to the negative log-likelihood under the Poisson-noise model. We solve the problem in (6.18) using a projected-gradient-descent algorithm. The regularization parameter τ so that it minimizes the mean-square error (MSE) with respect to the ground-truth is chosen via grid search.

After discarding the first T_{b} samples (burn-in period), we collect the next T samples for performing inference. We compute the posterior mean which corresponds to the minimum mean-square error (MMSE) estimate. Further, to quantify the uncertainty associated with our estimation, we also compute the pixel-wise standard-deviation map.

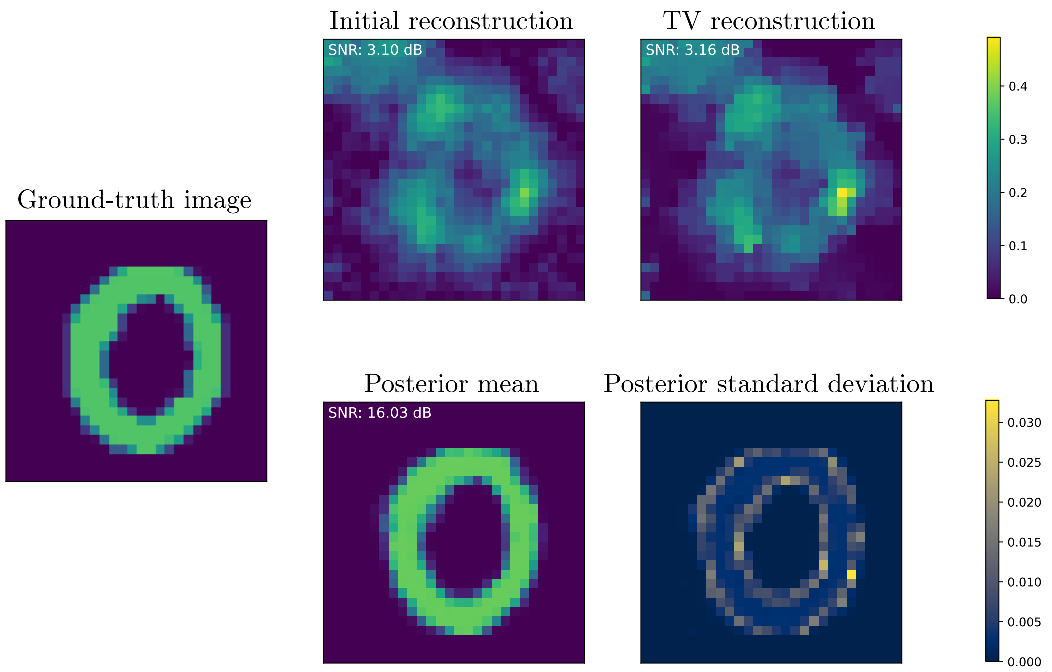


Figure 6.5: Reconstructions for phase retrieval (oversampling ratio $M/K = 0.1$).

We compare the performance of our GAN-based posterior-mean estimator with that of the TV-regularized method [25]

$$\mathbf{s}^*_{\mathrm{TV}}(\mathbf{y}^\dagger) = \arg\min_{\mathbf{s}\in\mathbb{R}^K} \left(\sum_{m=1}^{M} \left(-[\mathbf{y}^\dagger]_m \log\left(\left[|\mathbf{A}\mathbf{s}|^2\right]_m\right) + \left[|\mathbf{A}\mathbf{s}|^2\right]_m \right) + \tau\|\boldsymbol{\nabla}\mathbf{s}\|_{2,1} + i_+(\mathbf{s}) \right). \tag{6.21}$$

TV regularization is known to promote piecewise-constant solutions and is well-matched to our test images. We solve (6.21) using FISTA [23] initialized with $\mathbf{s}^*_{\mathrm{init}}(\mathbf{y}^\dagger)$. The regularization parameter τ is tuned for optimal MSE performance with the help of a grid search.

Results

To illustrate the advantage of our neural-network-based prior, we consider extreme imaging settings where the number of measurements M is very small. For the first case (Figure 6.5), we have that $\alpha = 0.36, M/K = 0.1, \sigma^2_{\mathbf{A}} = 10, \eta = 10^{-5}, T_{\mathrm{b}} = 8 \times 10^5$, and $T = 12 \times 10^5$. The parameters for the second case (Figure 6.6) are $M/K = 0.15, \sigma^2_{\mathbf{A}} = 0.5, \eta = 1.75 \times 10^{-6}, T_{\mathrm{b}} = 17.5 \times 10^5$, and $T = 5 \times 10^5$.

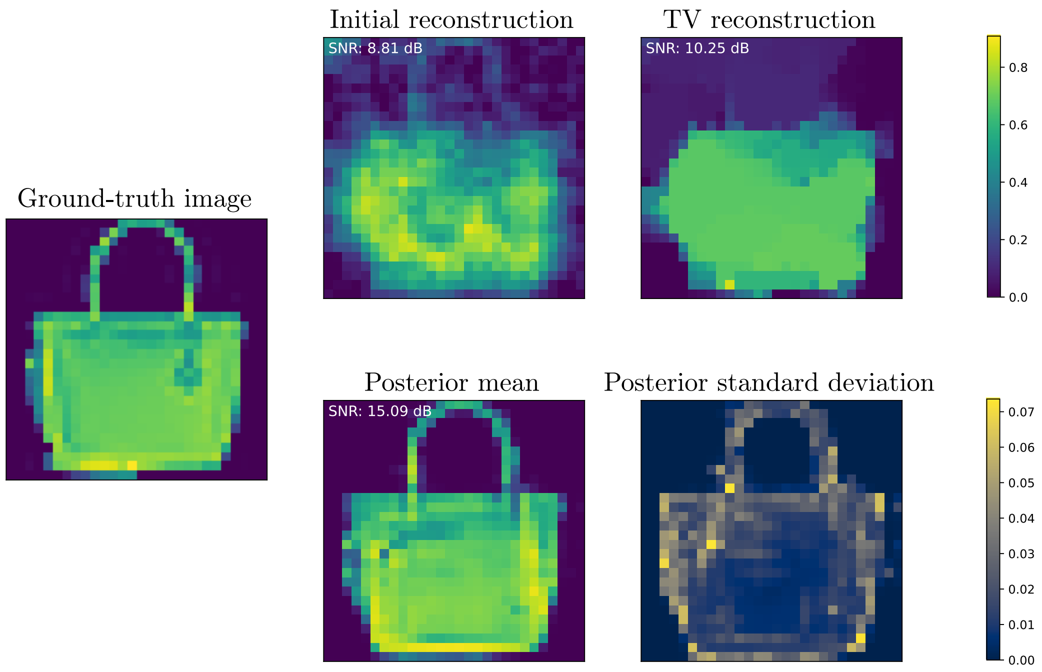


Figure 6.6: Reconstructions for phase retrieval (oversampling ratio $M/K = 0.15$).

In Figures 6.5 and 6.6, we see that the GAN-based posterior-mean estimator outperforms the TV-regularized method considerably. Here, the very low oversampling ratios severely affect the performance of TV regularization, even though it is a good fit for the underlying images. By contrast, despite the scarcity of measurements, our estimator remarkably yields excellent results. This highlights the potential of learning-based priors for highly ill-posed problems. Finally, we observe that, as one would expect, the standard-deviation maps indicate higher uncertainty at the edges for the posterior-mean estimator.

6.4.3 Optical Diffraction Tomography

We consider both simulated and real data for our ODT experiments.

Simulated data

In our simulated setup, the test image (Figure 6.7) that represents the RI contrast is a random sample from the dataset described in Section 6.4.1: a disc with constant intensity v.

The measurements are simulated using the BPM of Section 6.2.3 with an AWGN model of variance $\sigma_{\mathrm{n}}^2 = 0.05$. We set the sampling steps to $\delta_{\mathrm{x}} = \delta_{\mathrm{y}} = 0.1\,\mu\mathrm{m}$, the medium RI to $n_{\mathrm{b}} = 1.52$, and the wavelength to $\lambda = 0.406\,\mu\mathrm{m}$. We use $Q = 20$ incident tilted plane waves with angles that are uniformly spaced in the range $[-\pi/12, \pi/12]$.

For this setting, we use the augmented WGAN prior of Section 6.4.1 in our reconstruction framework.

Real data

In our experiment with real data, the sample is a 2D cross-section of two non-overlapping fibres immersed in oil ($n_{\mathrm{b}} = 1.525$) [287]. The RI contrast of the sample is negative. A standard Mach-Zehnder interferometer relying on off-axis digital holography ($\lambda = 0.450\,\mu\mathrm{m}$) is used to collect measurements from $Q = 59$ views in the range $[-\pi/6, \pi/6]$.

We crop the acquired data such that the measurement vector for each view is of length $M' = 256$. We take the discretized region of interest to be of the size (256×256) and we set the sampling steps for BPM (used for reconstruction) to $\delta_{\mathrm{x}} = \delta_{\mathrm{y}} = 0.1257\,\mu\mathrm{m}$. We assume an AWGN model of variance $\sigma_{\mathrm{n}}^2 = 0.15$ for the measurements.

Here, the WGAN for our prior is trained on a synthetic dataset containing 100,000 images of size (256×256), where each image consists of two non-overlapping discs with a constant intensity of one and a zero-valued background. The coordinates of the centers of the two discs are sampled from $U_{(20,235)}$ and their radii are sampled from $U_{[10,50]}$ subject to the constraint that they do not overlap. The distributions $p_{\mathbf{z}_1}$ and p_{z_2} are standard Gaussian distributions and the function h is taken to be

$$h(x) = -\frac{0.1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-\frac{t^2}{2}}\,\mathrm{d}t. \tag{6.22}$$

The architectures for the generator and critic networks are detailed in Appendix 6.6.3. The WGAN is trained for 500 epochs using RMSProp optimizers with a learning rate of 5×10^{-5} and a batch size of 128. The parameters λ_{gp} and n_{critic} are set as 10 and 5, respectively.

Methods

For both settings, the estimate $\mathbf{s}^*_{\mathrm{init}}(\mathbf{y}^\dagger)$ for MALA is obtained by the application of a filtered backpropagation algorithm that uses the Rytov approximation [281] to model the scattering. We collect T samples from the posterior distribution using MALA with a step-size τ and burn-in period T_{b}, and use them to compute the posterior mean and pixel-wise standard-deviation map.

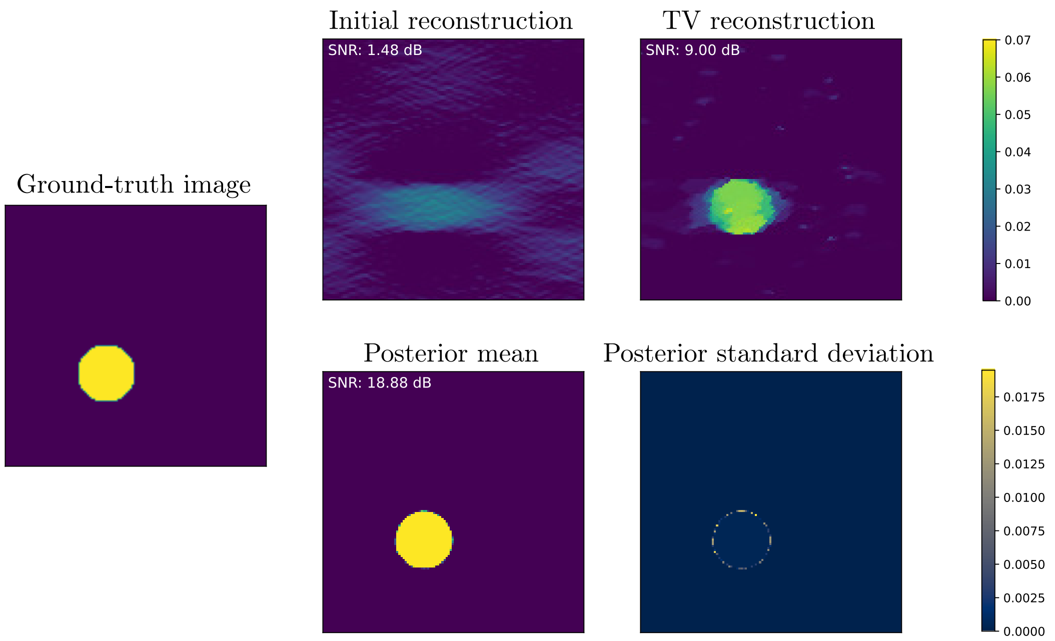


Figure 6.7: Reconstructions for ODT ($v = 0.07$).

We compare our estimator with the TV-based method

$$\mathbf{s}^*_{\mathrm{TV}}(\mathbf{y}^\dagger) = \arg\min_{\mathbf{s}\in\mathbb{R}^K} \left(\sum_{q=1}^{Q} \|\mathbf{y}^\dagger_q - \mathbf{H}_{\mathrm{bpm}}(\mathbf{s}; u^{\mathrm{in}}_q)\|_2^2 + \tau\|\boldsymbol{\nabla}\mathbf{s}\|_{2,1} + \mathcal{I}(\mathbf{s}) \right), \tag{6.23}$$

where $\mathcal{I}(\mathbf{s}) = i_+(\mathbf{s})$ for the simulated data and $\mathcal{I}(\mathbf{s}) = i_-(\mathbf{s})$ for the real data. This is a state-of-the-art method for ODT and is commonly used in practice [288, 289]. Moreover, it is well-suited for the constant-valued discs in our samples. The problem in (6.23) is solved using FISTA initialized with $\mathbf{s}^*_{\mathrm{init}}(\mathbf{y}^\dagger)$. The regularization parameter τ is tuned for optimal MSE performance in the simulated-data setting via a grid search, while it is tuned manually in the real-data setting.

Results

The settings that we consider for our ODT experiments are highly ill-posed as the incident waves only explore a limited range. As a result, the measurements lack information along the horizontal axis, which leads to the so-called missing-cone problem. For the first case (Figure 6.7), we have that $v = 0.07, \eta = 2 \times 10^{-7}, T_{\mathrm{b}} = 2 \times 10^4$, and $T = 8 \times 10^4$. For the second case (Figure 6.8), we have that $\eta = 5 \times 10^{-8}, T_{\mathrm{b}} = 15 \times 10^4$, and $T = 5 \times 10^4$.

In Figures 6.7 and 6.8, we observe that the TV reconstructions (and the initial ones) are elongated in the horizontal direction due to the lack of information along this axis. However, the GAN-based estimator is able to overcome the missing-cone problem. It

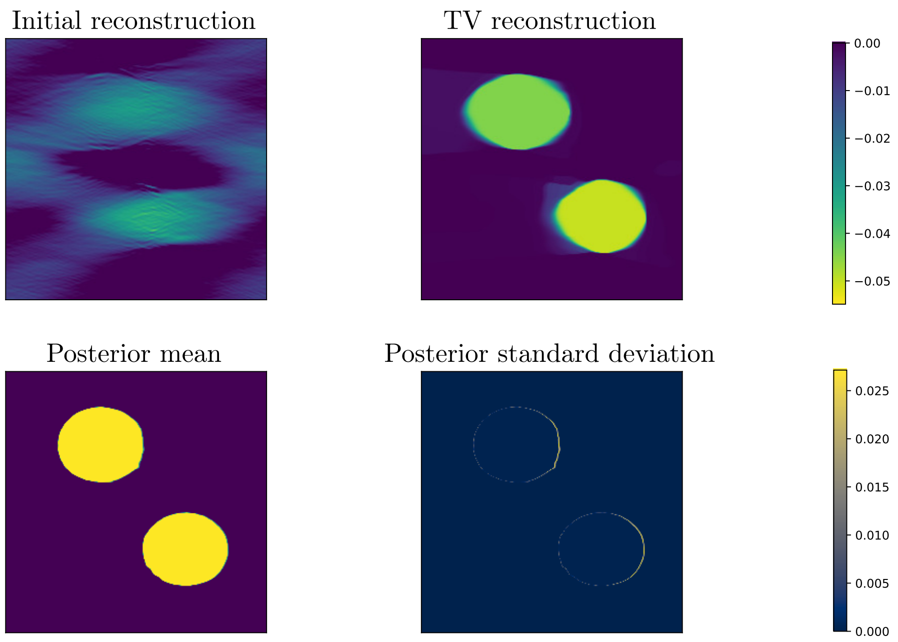


Figure 6.8: Reconstructions for ODT (real data).

yields reconstructions whose quality is remarkable.

6.4.4 Discussion

With the help of the above-described experiments, we have demonstrated the potential of our deep-generative-prior-based Bayesian reconstruction framework for challenging nonlinear inverse problems. We now mention some directions for future work which can further improve this framework.

In the present form, our scheme lacks theoretical guarantees for MALA to be geometrically ergodic (convergence to the equilibrium distribution at a geometric rate). A topic of future work could be to investigate the imposition of appropriate constraints on the generative model such that the resulting posterior distribution satisfies certain smoothness and tail conditions [290] that ensure geometric ergodicity of MALA.

The performance of our scheme heavily relies on how well the prior models the object of interest. Thus, any progress on the side of designing and training high-quality large-scale deep generative models could be translated to our framework.

While the neural-network-like structure of our forward models make our approach tractable, like MCMC methods in general, it requires a lot of computation. It could be interesting to consider alternatives to MALA that might help in speeding up this approach.

6.5 Summary

We have presented a Bayesian reconstruction framework for nonlinear inverse problems where the prior information on the image of interest is encoded by a deep latent variable generative model. Specifically, we have designed a tractable posterior-sampling scheme based on the Metropolis-adjusted Langevin algorithm for the class of nonlinear inverse problems where the forward model has a neural-network-like computational structure. This class includes most practical imaging modalities. We have proposed the concept of augmented generative models. They allow us to tackle the problem of the quantitative recovery of images. Finally, we have illustrated the benefits of our framework by applying it to two nonlinear imaging modalities—phase retrieval and optical diffraction tomography.

6.6 Appendix

6.6.1 Properties of the Posterior Distribution

Well-posedness

A Bayesian inverse problem is said to be well-posed in some metric on the space of probability measures if the posterior distribution exists, is unique, and is continuous with respect to the measurements for the chosen metric [283]. Here, we present sufficient conditions from [283, Assumptions 3.5, 3.10 and Theorems 3.6, 3.12] that guarantee the well-posedness of our problem in the latent space, that is, with respect to $\mathbb{P}_{\mathbf{Z}|\mathbf{Y}=\mathbf{y}^\dagger}$ as described in Section 6.3.2.

The following conditions are stated for $p_{\mathbf{Z}}$-almost every (a.e.) $\mathbf{z}' \in \mathbb{R}^{d+1}$ and every $\mathbf{y}^\dagger \in \mathbb{R}^M$.

Conditions.

1. $p_{\mathbf{Y}|\mathbf{Z}}(\cdot|\mathbf{z}')$ is a strictly positive pdf.
2. $\int_{\mathbb{R}^{d+1}} |p_{\mathbf{Y}|\mathbf{Z}}(\mathbf{y}^\dagger|\tilde{\mathbf{z}})| p_{\mathbf{Z}}(\tilde{\mathbf{z}}) \mathrm{d}\tilde{\mathbf{z}} < \infty$
3. There exists g with $\int_{\mathbb{R}^{d+1}} |g(\tilde{\mathbf{z}})| p_{\mathbf{Z}}(\tilde{\mathbf{z}}) \mathrm{d}\tilde{\mathbf{z}} < \infty$ such that $p_{\mathbf{Y}|\mathbf{Z}}(\mathbf{y}^\ddagger|\cdot) \leq g$ for all $\mathbf{y}^\ddagger \in \mathbb{R}^M$.
4. $p_{\mathbf{Y}|\mathbf{Z}}(\cdot|\mathbf{z}')$ is continuous.
5. There exists g' with $\int_{\mathbb{R}^{d+1}} |g'(\tilde{\mathbf{z}})| p_{\mathbf{Z}}(\tilde{\mathbf{z}}) \mathrm{d}\tilde{\mathbf{z}} < \infty$ such that $\|\mathbf{z}'\|_2^p \, p_{\mathbf{Y}|\mathbf{Z}}(\mathbf{y}^\dagger|\mathbf{z}') \leq g'(\mathbf{z}')$, where $p \in [1, \infty)$.

If the conditions (1) – (4) hold, our Bayesian inverse problem in the latent space is well-posed in the Prokhorov, Hellinger and total-variation distances. In addition, if condition (5) holds, then the problem is also well-posed in the Wasserstein p-distance.

For additive white-Gaussian-noise (AWGN) models, the conditions (1) − (4) are satisfied for any physical forward model $\mathbf{H}$ and prior distribution $p_{\mathbf{Z}}$. Further, if $p_{\mathbf{Z}}$ is such that $\int \|\tilde{\mathbf{z}}\|_2^p \, p_{\mathbf{Z}}(\tilde{\mathbf{z}}) \mathrm{d}\tilde{\mathbf{z}} < \infty$ (*e.g.,* Gaussian distribution), condition (5) is also satisfied [283, Corollary 5.1]. As for the Poisson-noise models used in some of our experiments, they do not fall within this framework of well-posedness developed in [283].

Existence of Moments

Based on Proposition 3.6 in [78], we also present some conditions under which the moments of our posterior distribution $\mathbb{P}_{\mathbf{S}|\mathbf{Y}=\mathbf{y}^\dagger}$ exist. If the augmented deep generative prior G_h is Lipschitz-continuous and the prior distribution $p_{\mathbf{Z}}$ has finite moments $\mathbb{E}_{\mathbb{P}_{\mathbf{Z}}}[|\mathbf{Z}|^k]$ for $k = 1, 2, \ldots, K$, then the Kth posterior moment $\mathbb{E}_{\mathbb{P}_{\mathbf{S}|\mathbf{Y}=\mathbf{y}^\dagger}}[|(\mathbf{S}|\mathbf{Y} = \mathbf{y}^\dagger)|^K]$ exists for almost all measurements $\mathbf{y}^\dagger$.

The typical choice for $p_{\mathbf{Z}}$ is the standard Gaussian distribution, which has finite moments. The Lipschitz-continuity of G_h is guaranteed if the generative network G and the function h are both Lipschitz-continuous and bounded. The Lipschitz condition on the network G holds when its weights and biases are finite-valued and it consists of Lipschitz-continuous activation functions (*e.g.,* ReLU, sigmoid). The boundedness of G is ensured when the activation function in the output layer is bounded (such as the sigmoid function). These are conditions that are satisfied by the networks used in Section 6.4. Further, in our experiments, we choose the function h to be a scaled version of the cumulative density function of the standard normal distribution, which is Lipschitz-continuous and bounded.

6.6.2 Wasserstein Generative Adversarial Networks (WGANs)

Classical generative adversarial networks (GANs) [69] are known to suffer from issues such as the instability of the training process [291, 292], vanishing gradients, and mode collapse. The framework of Wasserstein GANs (WGANs) [220] is an alternative that alleviates these problems.

Let $\mathcal{D}$ be a dataset consisting of samples drawn from a probability distribution $\mathbb{P}_{\mathrm{r}}$. The goal is to build a model using $\mathcal{D}$ that can generate samples that follow a distribution that closely approximates $\mathbb{P}_{\mathrm{r}}$. A WGAN consists of a generator network $\mathrm{G}_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^K$ ($d \ll K$), where $\boldsymbol{\theta} \in \mathbb{R}^{d_1}$ denotes its trainable parameters. It takes an input vector $\mathbf{z} \in \mathbb{R}^d$, sampled from a fixed pdf $p_{\mathbf{Z}}$, and outputs $\mathrm{G}_{\boldsymbol{\theta}}(\mathbf{z}) \in \mathbb{R}^K$. The samples generated by this model follow some distribution $\mathbb{P}_{\boldsymbol{\theta}}$ that is characterized by $\mathrm{G}_{\boldsymbol{\theta}}$ and $p_{\mathbf{Z}}$. Thus, the parameters $\boldsymbol{\theta}$ need to be chosen such that $\mathbb{P}_{\boldsymbol{\theta}}$ approximates $\mathbb{P}_{\mathrm{r}}$ well.

In the WGAN framework, the generator is trained to minimize the Wasserstein-1 (or

Earth-Mover) distance between $\mathbb{P}_\mathrm{r}$ and $\mathbb{P}_{\boldsymbol{\theta}}$, which is given by

$$W(\mathbb{P}_\mathrm{r}, \mathbb{P}_{\boldsymbol{\theta}}) = \inf_{\gamma \in \pi(\mathbb{P}_\mathrm{r}, \mathbb{P}_{\boldsymbol{\theta}})} \mathbb{E}_{(\mathsf{U},\mathsf{V})\sim\gamma}\Big[\|\mathsf{U} - \mathsf{V}\|\Big]. \tag{6.24}$$

Here, $\pi(\mathbb{P}_\mathrm{r}, \mathbb{P}_{\boldsymbol{\theta}})$ is the collection of all joint distributions with marginals $\mathbb{P}_\mathrm{r}$ and $\mathbb{P}_{\boldsymbol{\theta}}$. The Kantorovich-Rubinstein duality theorem [293] states that (6.24) can be written as

$$W(\mathbb{P}_\mathrm{r}, \mathbb{P}_{\boldsymbol{\theta}}) = \sup_{f \in \mathcal{X}} \Bigg(\mathbb{E}_{\mathsf{U}\sim\mathbb{P}_\mathrm{r}}[f(\mathsf{U})] - \mathbb{E}_{\mathsf{V}\sim\mathbb{P}_{\boldsymbol{\theta}}}[f(\mathsf{V})]\Bigg), \tag{6.25}$$

where $\mathcal{X} = \{f : \mathbb{R}^K \to \mathbb{R} \mid f \text{ is 1-Lipschitz}\}$. The space $\mathcal{X}$ is then replaced by a family of 1-Lipschitz functions represented by a critic neural network $\mathrm{D}_{\boldsymbol{\phi}} : \mathbb{R}^K \to \mathbb{R}, \mathbf{w} \mapsto \mathrm{D}_{\boldsymbol{\phi}}(\mathbf{w})$ with appropriately constrained parameters $\boldsymbol{\phi} \in \mathbb{R}^{d_2}$. This leads to the minimax problem

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{d_1}} \max_{\boldsymbol{\phi} \in \mathcal{Y}} \Bigg(\mathbb{E}_{\mathsf{U}\sim\mathbb{P}_\mathrm{r}}[\mathrm{D}_{\boldsymbol{\phi}}(\mathsf{U})] - \mathbb{E}_{\mathsf{V}\sim\mathbb{P}_{\boldsymbol{\theta}}}[\mathrm{D}_{\boldsymbol{\phi}}(\mathsf{V})]\Bigg), \tag{6.26}$$

where $\mathcal{Y} = \{\boldsymbol{\phi} \in \mathbb{R}^{d_2} \mid \mathrm{D}_{\boldsymbol{\phi}} \text{ is 1-Lipschitz}\}$. In [220], the authors enforce the 1-Lipschitz condition on $\mathrm{D}_{\boldsymbol{\phi}}$ by clipping its weights during training. Instead, the 1-Lipschitz constraint can also be enforced by adding a gradient penalty to the cost function in (6.26) [294]. The regularized minimax problem becomes

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{d_1}} \max_{\boldsymbol{\phi} \in \mathbb{R}^{d_2}} \Bigg(\mathbb{E}_{\mathsf{U}\sim\mathbb{P}_\mathrm{r}}[\mathrm{D}_{\boldsymbol{\phi}}(\mathsf{U})] - \mathbb{E}_{\mathsf{V}\sim\mathbb{P}_{\boldsymbol{\theta}}}[\mathrm{D}_{\boldsymbol{\phi}}(\mathsf{V})\Big] + \lambda_\mathrm{gp}\mathbb{E}_{\mathsf{W}\sim\mathbb{P}_\mathrm{int}}\Big[(\|\boldsymbol{\nabla}_{\mathbf{w}}\mathrm{D}_{\boldsymbol{\phi}}(\mathsf{W})\| - 1)^2\Big]\Bigg), \tag{6.27}$$

where a point $\mathsf{W} \sim \mathbb{P}_\mathrm{int}$ is obtained by sampling uniformly along straight lines between points drawn from $\mathbb{P}_\mathrm{r}$ and $\mathbb{P}_{\boldsymbol{\theta}}$, and $\lambda_\mathrm{gp} > 0$ is a hyperparameter.

In practice, Problem (6.27) is solved using mini-batch stochastic-gradient algorithms in an alternating manner. During each iteration for the critic, we collect a batch of samples $\{\mathbf{x}^{(n)}\}_{n=1}^{N_c}$ from the dataset $\mathcal{D}$. We sample vectors $\{\mathbf{z}^{(n)}\}_{n=1}^{N_c}$ from p_{Z} and a sequence of numbers $\{\alpha^{(n)}\}_{n=1}^{N_c}$ from the uniform distribution $U_{[0,1]}$, and we construct $\mathbf{w}^{(n)} = \alpha^{(n)}\mathbf{x}^{(n)} + (1-\alpha^{(n)})\mathrm{G}_{\boldsymbol{\theta}}(\mathbf{z}^{(n)})$. The critic parameters are then updated by ascending along the gradient given by

$$\frac{1}{N_c}\boldsymbol{\nabla}_{\boldsymbol{\phi}}\left(\sum_{n=1}^{N_c} \mathrm{D}_{\boldsymbol{\phi}}(\mathbf{x}^{(n)}) - \mathrm{D}_{\boldsymbol{\phi}}(\mathrm{G}_{\boldsymbol{\theta}}(\mathbf{z}^{(n)})) + \lambda_\mathrm{gp}(\|\boldsymbol{\nabla}_{\mathbf{w}}\mathrm{D}_{\boldsymbol{\phi}}(\mathbf{w}^{(n)})\| - 1)^2\right). \tag{6.28}$$

During each iteration for the generator, we sample latent vectors $\{\mathbf{z}^{(n)}\}_{n=1}^{N_g}$ from p_{Z}. The generator parameters are then updated by descending along the gradient given by

$$\frac{1}{N_g}\nabla_{\boldsymbol{\theta}}\left(\sum_{n=1}^{N_g} -\mathrm{D}_{\boldsymbol{\phi}}(\mathrm{G}_{\boldsymbol{\theta}}(\mathbf{z}^{(n)}))\right). \tag{6.29}$$

Typically, for every generator iteration, the critic is trained for n_{critic} iterations.

6.6.3 WGAN Architectures

The generator and critic architectures used for datasets consisting of constant-valued discs are shown in Table 6.1 and 6.4. The architectures used for the MNIST and Fashion MNIST datasets are shown in Table 6.2 and 6.3, respectively.

Layers	Output shape
Conv 4×4 + LReLU	$512 \times 4 \times 4$
Conv 3×3 + LReLU	$512 \times 4 \times 4$
Upsample	$512 \times 8 \times 8$
Conv 3×3 + LReLU	$256 \times 8 \times 8$
Upsample	$256 \times 16 \times 16$
Conv 3×3 + LReLU	$128 \times 16 \times 16$
Upsample	$128 \times 32 \times 32$
Conv 3×3 + LReLU	$64 \times 32 \times 32$
Upsample	$64 \times 64 \times 64$
Conv 3×3 + LReLU	$32 \times 64 \times 64$
Upsample	$32 \times 128 \times 128$
Conv 3×3 + LReLU	$16 \times 128 \times 128$
Conv 1×1 + Sigmoid	$1 \times 128 \times 128$

(a) Generator network with ($128 \times 1 \times 1$) input shape.

Layers	Output shape
Conv 1×1 + LReLU	$16 \times 128 \times 128$
Conv 3×3 + LReLU	$16 \times 128 \times 128$
Conv 3×3 + LReLU	$32 \times 128 \times 128$
Downsample	$32 \times 64 \times 64$
Conv 3×3 + LReLU	$64 \times 64 \times 64$
Downsample	$64 \times 32 \times 32$
Conv 3×3 + LReLU	$128 \times 32 \times 32$
Downsample	$128 \times 16 \times 16$
Conv 3×3 + LReLU	$256 \times 16 \times 16$
Downsample	$256 \times 8 \times 8$
Conv 3×3 + LReLU	$512 \times 8 \times 8$
Downsample	$512 \times 4 \times 4$
Conv 3×3 + LReLU	$512 \times 4 \times 4$
Conv 4×4 + LReLU	$512 \times 1 \times 1$
Reshape	1×512
Fully-connected	1×1

(b) Critic network with ($1 \times 128 \times 128$) input shape.

Table 6.1: Generator and critic architectures (single disc). The negative slope for LReLU is set as 0.2. The upsampling layer uses nearest-neighbor interpolation while the downsampling layer involves max pooling.

Layers	Output shape
Fully-connected + LReLU	1×128
Fully-connected + Batch-norm + LReLU	1×256
Fully-connected + Batch-norm + LReLU	1×512
Fully-connected + Batch-norm + LReLU	1×1024
Fully-connected + Sigmoid	1×784

(a) Generator network with (1×100) input shape.

Layers	Output shape
Fully-connected + LReLU	1×512
Fully-connected + LReLU	1×256
Fully-connected	1×1

(b) Critic network with (1×784) input shape.

Table 6.2: Generator and critic architectures (MNIST). The negative slope for LReLU is set as 0.2.

Layers	Output shape
Fully-connected + Batch-norm + ReLU	1×1024
Fully-connected + Batch-norm + ReLU	1×6272
Reshape	$128 \times 7 \times 7$
ConvTranspose 4×4 + Batch-norm + ReLU	$64 \times 14 \times 14$
ConvTranspose 4×4 + Sigmoid	$1 \times 28 \times 28$

(a) Generator network with (1×100) input shape.

Layers	Output shape
Conv 4×4 + LReLU	$64 \times 14 \times 14$
Conv 4×4 + Batch-norm + LReLU	$128 \times 7 \times 7$
Reshape	1×6272
Fully-connected + Batch-norm + LReLU	1×1024
Fully-connected	1×1

(b) Critic network with ($1 \times 28 \times 28$) input shape.

Table 6.3: Generator and critic architectures (Fashion-MNIST). The negative slope for LReLU is set as 0.2.

Layers	Output shape
Conv 4×4 + LReLU	$256 \times 4 \times 4$
Conv 3×3 + LReLU	$256 \times 4 \times 4$
Upsample	$256 \times 8 \times 8$
Conv 3×3 + LReLU	$128 \times 8 \times 8$
Upsample	$128 \times 16 \times 16$
Conv 3×3 + LReLU	$64 \times 16 \times 16$
Upsample	$64 \times 32 \times 32$
Conv 3×3 + LReLU	$32 \times 32 \times 32$
Upsample	$32 \times 64 \times 64$
Conv 3×3 + LReLU	$16 \times 64 \times 64$
Upsample	$16 \times 128 \times 128$
Conv 3×3 + LReLU	$8 \times 128 \times 128$
Upsample	$8 \times 256 \times 256$
Conv 3×3 + LReLU	$4 \times 256 \times 256$
Conv 1×1 + Sigmoid	$1 \times 256 \times 256$

(a) Generator network with $(128 \times 1 \times 1)$ input shape.

Layers	Output shape
Conv 1×1 + LReLU	$4 \times 256 \times 256$
Conv 3×3 + LReLU	$4 \times 256 \times 256$
Conv 3×3 + LReLU	$8 \times 256 \times 256$
Downsample	$8 \times 128 \times 128$
Conv 3×3 + LReLU	$16 \times 128 \times 128$
Downsample	$16 \times 64 \times 64$
Conv 3×3 + LReLU	$32 \times 64 \times 64$
Downsample	$32 \times 32 \times 32$
Conv 3×3 + LReLU	$64 \times 32 \times 32$
Downsample	$64 \times 16 \times 16$
Conv 3×3 + LReLU	$128 \times 16 \times 16$
Downsample	$128 \times 8 \times 8$
Conv 3×3 + LReLU	$256 \times 8 \times 8$
Downsample	$256 \times 4 \times 4$
Conv 3×3 + LReLU	$256 \times 4 \times 4$
Conv 4×4 + LReLU	$256 \times 1 \times 1$
Reshape	1×256
Fully-connected	1×1

(b) Critic network with $(1 \times 256 \times 256)$ input shape.

Table 6.4: Generator and critic architectures (two non-overlapping discs). The negative slope for LReLU is set as 0.2. The upsampling layer uses nearest-neighbor interpolation while the downsampling layer involves max pooling.

7 Deep Spatiotemporal Regularization for Dynamic Fourier Ptychography

[1]This chapter explores the use of an untrained neural network as an implicit regularizer in the context of Fourier ptychography (FP). This modality involves the acquisition of several low-resolution intensity images of a sample under varying illumination angles. They are then combined into a high-resolution complex-valued image by solving a phase-retrieval problem. The objective in dynamic FP is to obtain a sequence of high-resolution images of a moving sample. There, the application of standard frame-by-frame reconstruction methods limits the temporal resolution due to the large number of measurements that must be acquired for each frame. We instead propose a neural-network-based reconstruction framework for dynamic FP, which achieves high temporal resolution without compromising the spatial resolution. It does not require training data and also recovers the pupil function of the microscope.

7.1 Introduction

In Fourier ptychography (FP) [276], hundreds of low-resolution intensity images are acquired by illuminating the object of interest with a coherent light source with varying incidence angles. This task is typically performed using a LED array and a microscope with a low numerical aperture (NA) objective lens, which makes FP a low-cost and label-free imaging modality. The collection of measurements is then algorithmically combined into a high-resolution complex-valued image of the sample over a large field of view. Thus, FP has a high space-bandwidth product.

Building upon the pioneering work of Zheng *et al.* [276], the capabilities of FP have been extended in a variety of ways by improving the optical acquisition setup. For instance, in [296] and [297], the sequence of illuminations is optimized via an importance metric and neural networks, respectively. Multiplexed FP is introduced in [298], where one illuminates the sample with multiple LEDs and is able to reduce the number of

[1]This chapter is based on our work [295].

measurements. Further, optimal combinations of LEDs are studied in [297, 299–301].

There have also been several improvements on the computational side for FP. At its core, the reconstruction process involves the solution of a phase-retrieval (PR) problem—the recovery of phase information from intensity measurements. In [276], this task is performed by using the iterative Gerchberg-Saxton (GS) algorithm [302]. As PR is a non-convex problem, the solution obtained by GS depends on the starting point. This problem of initialization is tackled in [303]. In [304–306], PR is formulated as a convex optimization problem with the help of a lifting scheme. However, this elegant approach comes at the cost of a large computational burden. As the acquired measurements are typically corrupted by noise, maximum-likelihood estimation offers an adequate framework for one to incorporate the noise statistics [307]. The resulting optimization problems are solved efficiently by gradient-based or higher-order methods [308, 309]. A thorough comparative study of different methods for PR can be found in [310]. In addition to solving the PR problem, algorithms that include the estimation of the pupil function of the microscope [311] and correction of the LED positions [312, 313] have also been proposed.

While FP has matured into a versatile modality with numerous applications [314], high-quality high-speed imaging remains a challenge. The temporal resolution in FP is inherently limited by the large number of measurements that need to be acquired in order to reconstruct the high-resolution image of the sample. To alleviate this problem, *ad hoc* acquisition setups [299, 300, 315] have been devised. They allow one to obtain a higher temporal resolution without a significant deterioration of the spatial resolution. Alternatively, there has been a lot of interest in the development of sophisticated computational methods to solve the PR problem with only a few measurements. In such ill-posed scenarios, regularization techniques can be used to incorporate some prior knowledge about the sample of interest. These are typically applied by formulating PR as an optimization problem where the cost functional consists of a data-fidelity term and a regularization term. The data-fidelity term ensures that the solution is consistent with the observed data while the regularization promotes solutions with the desired properties. For example, the popular total-variation (TV) regularization [25] favors piecewise-constant images and has been adapted for FP in several works [316–319]. Group-sparsity-based priors have been successfully deployed in FP as well [320]. An online plug-and-play approach for FP has also been proposed in [321], where sophisticated denoisers such as BM3D [204] are used for (implicit) regularization.

Over the past few years, deep-learning-based methods have yielded impressive results, outperforming the model-based regularized methods in a variety of imaging modalities, especially in ill-posed settings [38, 322]. In the context of FP, deep neural networks have been trained in a supervised manner as nonlinear mappings that take the low-resolution measurements and output the high-resolution image of interest [323–325]. Further, in [326, 327], pre-trained deep generative priors are used to solve the PR problem. For more details regarding FP, we refer the reader to recent comprehensive reviews [314, 328].

In dynamic FP, when it is desired to image a moving sample, the computational methods described above must be applied in a frame-by-frame manner to obtain the sequence of high-resolution images, without accounting for the temporal dependencies in the measurements. Yet, one can decrease the number of measurements required per frame (thus increasing the effective imaging speed) by exploiting the temporal correlations in the sequence of images to be recovered. Based on this idea, the concept of low-rank FP is introduced in [329], where a low-rank constraint is enforced on the matrix formed by stacking the (vectorized) images.

7.1.1 Contributions

In this chapter, we propose a novel computational framework for dynamic FP. Inspired by the method developed in [330] for dynamic magnetic resonance imaging, we use a deep neural network to impose a spatiotemporal regularization on the sequence of complex-valued images to be recovered. More specifically, we parameterize each image in the sequence as the output of a single convolutional network corresponding to some fixed latent input vector. These input vectors are chosen to lie on a one-dimensional manifold. The parameters of the network are then estimated by optimizing a likelihood-based criterion. The architecture of the generative network imposes an implicit spatial regularization on the images while the constraints on the input latent vectors allow the network to associate their proximity with temporal variations in the sequence. Our method does not require any training data. It also estimates the pupil function together with the complex-valued images, which means it can be readily applied for different settings. We assess the performance of our framework on simulated data with a single measured low-resolution image per reconstructed frame and show that it paves the way for high-quality ultrafast FP.

The chapter is organized as follows. In 7.2, we describe a continuous-domain physical model for FP along with its computationally efficient discretization. We present the proposed reconstruction framework in 7.3 and the experimental results in 7.4.

7.2 Physical Model

In this section, we first formulate the physical model that relates the acquired measurements and the sample of interest in the continuous domain. Then, we present a discretized version of the forward model that can be implemented in a computationally efficient manner.

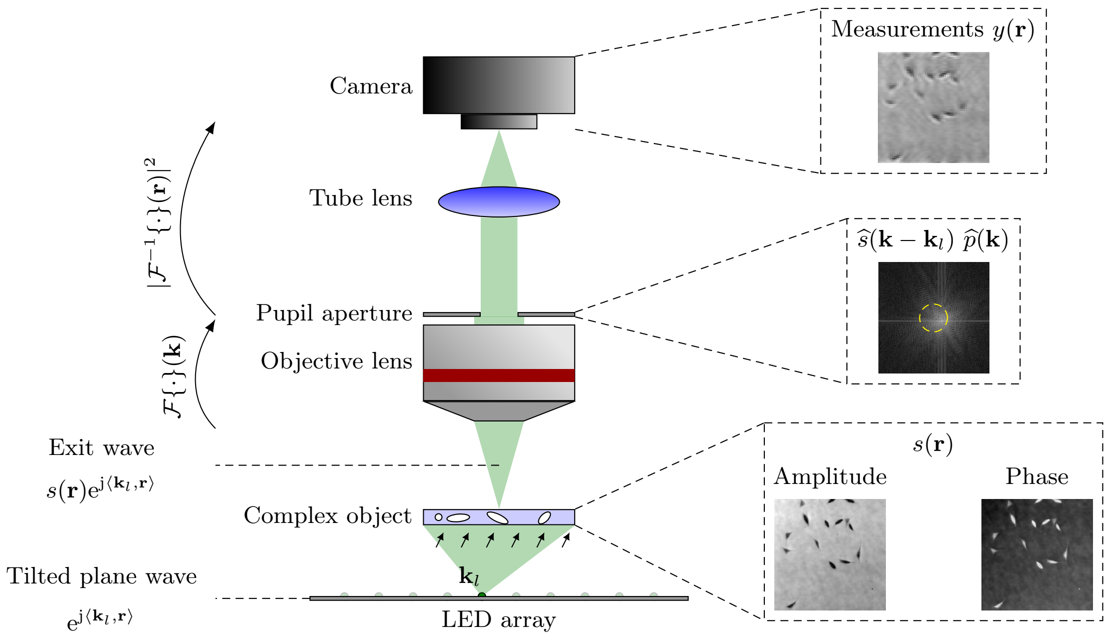

Measurements $y(\mathbf{r})$
Camera
$|\mathcal{F}^{-1}\{\cdot\}(\mathbf{r})|^2$
Tube lens
$\widehat{s}(\mathbf{k}-\mathbf{k}_l)\ \widehat{p}(\mathbf{k})$
Pupil aperture
Objective lens
$\mathcal{F}\{\cdot\}(\mathbf{k})$
Exit wave
$s(\mathbf{r})\mathrm{e}^{\mathrm{j}\langle\mathbf{k}_l,\mathbf{r}\rangle}$
$s(\mathbf{r})$
Amplitude
Phase
Complex object
Tilted plane wave
$\mathrm{e}^{\mathrm{j}\langle\mathbf{k}_l,\mathbf{r}\rangle}$
$\mathbf{k}_l$
LED array

Figure 7.1: Acquisition setup of Fourier ptychography.

7.2.1 Continuous-Domain Formulation

The optical system in FP usually involves an array of L LEDs (see Figure 7.1), where the lth LED illuminates the specimen with a tilted plane wave with wave vector $\mathbf{k}_l \in \mathbb{R}^2$ ($l \in \mathcal{L} = \{1, 2, \dots, L\}$) and wavelength $\lambda > 0$. In this work, we consider the case where only one LED is turned on for each measured image. However, our framework is also compatible with more sophisticated acquisition settings [298].

We model the sample of interest as a 2D complex object, which is a valid assumption for thin samples. Therefore, we can represent the moving sample as a complex-valued function $s : \Omega_{\mathrm{S}} \times \mathbb{R}_{\geq 0} \to \mathbb{C}$, where $\Omega_{\mathrm{S}} \subset \mathbb{R}^2$ includes the region of interest of the sample. Let $\{t_q\}_{q=1}^{Q}$ be the uniformly-spaced timestamps, with spacing Δ_{t}, at which we are interested in observing the sample. We assume that the sample moves very slowly in the intervals $\{T_q = [t_q - \Delta_{\mathrm{t}}/2, t_q + \Delta_{\mathrm{t}}/2]\}_{q=1}^{Q}$. Thus, during T_q, we can acquire multiple measurements $\{y_{q,w} : \Omega_{\mathrm{Y}} \to \mathbb{R}\}_{w=1}^{W}$, where $W \leq L$ and where $\Omega_{\mathrm{Y}} \subset \mathbb{R}^2$ includes the support of the measurement, of the object $s(\cdot, t_q)$. Here, the tradeoff between the temporal resolution and the spatial resolution can be understood in terms of Δ_{t} and W: a small value of Δ_{t} (high temporal resolution) implies a small value of W, which yields a low spatial resolution.

Let $\mathcal{I}_q \subset \mathcal{L}$, where $q \in \{1, 2, \dots, Q\}$, be the set of LEDs that are switched on during T_q; the cardinality of this set is $|\mathcal{I}_q| = W$. Further, for $w \in \{1, 2, \dots, W\}$, we introduce $l_{q,w} = \mathcal{I}_q(w) \in \mathcal{L}$ to denote the wth entry of $\mathcal{I}_q$. The measurement image $y_{q,w}$ is obtained when $s(\cdot, t_q)$ is illuminated by the $l_{q,w}th$ LED with the tilted plane wave $\mathbf{r} \mapsto \mathrm{e}^{\mathrm{j}\langle \mathbf{k}_{l_{q,w}}, \mathbf{r}\rangle}$.

As mentioned in [298, 310], it is given by

$$\begin{aligned} y_{q,w}(\mathbf{r}) &= \left| \mathcal{F}^{-1} \left\{ \widehat{p}(\mathbf{k}) \mathcal{F} \left\{ s(\cdot, t_q) \mathrm{e}^{\mathrm{j}\langle \mathbf{k}_{l_{q,w}}, \cdot \rangle} \right\} (\mathbf{k}) \right\} (\mathbf{r}) \right|^2 \\ &= \left| \mathcal{F}^{-1} \left\{ \widehat{p}(\mathbf{k}) \widehat{s}(\mathbf{k} - \mathbf{k}_{l_{q,w}}, t_q) \right\} (\mathbf{r}) \right|^2 . \end{aligned} \tag{7.1}$$

Here, the operators $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, respectively, $\mathbf{k} \in \mathbb{R}^2$ is the 2D spatial frequency variable, and the quantity $\widehat{s}(\mathbf{k}, t_q)$ denotes the Fourier transform of $s(\mathbf{r}, t_q)$. The pupil function[2] $\widehat{p} : \mathbb{R}^2 \to \mathbb{C}$ models the pupil aperture and is compactly supported on a disk of radius $2\pi \frac{\mathrm{NA}}{\lambda}$, where NA is the numerical aperture of the system, thus cutting off high frequencies.

7.2.2 Camera Sampling

The camera in the acquisition setup samples $y_{q,w}$ on a uniform grid with stepsize Δ and records a discrete image $\widetilde{\mathbf{y}}^{\mathrm{im}}_{q,w}$ of size[3] $(M \times M)$ such that

$$\widetilde{\mathbf{y}}^{\mathrm{im}}_{q,w} = \mathrm{Noise}(\mathbf{y}^{\mathrm{im}}_{q,w}), \tag{7.2}$$

where the $(M \times M)$ image $\mathbf{y}^{\mathrm{im}}_{q,w}$ is the sampled version of $y_{q,w}$ given by

$$\mathbf{y}^{\mathrm{im}}_{q,w}[m_1, m_2] = y_{q,w}\Big((m_1 - M/2)\Delta, (m_2 - M/2)\Delta \Big) \tag{7.3}$$

for $m_1 = 0, \ldots, (M-1)$ and $m_2 = 0, \ldots, (M-1)$, and the operator $\mathrm{Noise}(\cdot)$ models the corruption of $\mathbf{y}^{\mathrm{im}}_{q,w}$ by noise. Consider the quantity

$$u_{q,w}(\mathbf{r}) = \mathcal{F}^{-1} \left\{ \widehat{p}(\mathbf{k}) \mathcal{F} \left\{ s(\cdot, t_q) \mathrm{e}^{\mathrm{j}\langle \mathbf{k}_{l_{q,w}}, \cdot \rangle} \right\} (\mathbf{k}) \right\} (\mathbf{r}). \tag{7.4}$$

Due to the compact support of the pupil function $\widehat{p}$, the maximum angular frequency of $u_{q,w}$ is $2\pi \frac{\mathrm{NA}}{\lambda}$. Note that the Fourier transform of $y_{q,w}$ can be written as

$$\mathcal{F}\left\{ y_{q,w} \right\}(\mathbf{k}) = \mathcal{F}\left\{ |u_{q,w}|^2 \right\}(\mathbf{k}) = \left(\overline{\widehat{u}^{\vee}_{q,w}} * \widehat{u}_{q,w} \right)(\mathbf{k}), \tag{7.5}$$

where $\overline{\widehat{u}^{\vee}_{q,w}}$ denotes the complex conjugate of $\widehat{u}^{\vee}_{q,w}$ which is given by $\widehat{u}^{\vee}_{q,w}(\mathbf{k}) = \widehat{u}_{q,w}(-\mathbf{k})$, and $*$ denotes the convolution operation. Thus, the maximum angular frequency of $y_{q,w}$ is $\frac{4\pi \mathrm{NA}}{\lambda}$. Consequently, the Nyquist criterion dictates that the sampling step Δ of the camera should satisfy

$$\Delta \leq \frac{\lambda}{4\mathrm{NA}}. \tag{7.6}$$

[2]The pupil function $\widehat{p}$ is described directly in the Fourier domain.

[3]We consider square even-sized images for the sake of simplicity.

7.2.3 Discretized Forward Model

In this work, we obtain a discrete version $\mathbf{s}_q^{\mathrm{im}}$ of $s(\mathbf{r}, t_q)$ by sampling it on a uniform $(N \times N)$ grid with pixel-size Δ_{r}, as

$$\mathbf{s}_q^{\mathrm{im}}[n_1, n_2] = s\Big((n_1 - N/2)\Delta_{\mathrm{r}}, (n_2 - N/2)\Delta_{\mathrm{r}}, t_q\Big) \tag{7.7}$$

for $n_1 = 0, \ldots, (N-1)$ and $n_2 = 0, \ldots, (N-1)$. The image size is given by $N = r_{\mathrm{p}} M$, where $r_{\mathrm{p}} = \Delta/\Delta_{\mathrm{r}} \in \mathbb{N}$ is the upsampling factor. Now, consider the 2D discrete Fourier transform (DFT) of $\mathbf{s}_q^{\mathrm{im}}$. The corresponding pixel size in the Fourier domain (or angular frequency resolution) is $\Delta_{\mathrm{k}} = 2\pi/N\Delta_{\mathrm{r}}$. Thus, we discretize the pupil function such that

$$\widehat{\mathbf{p}}^{\mathrm{im}}[k_1, k_2] = \widehat{p}\Big((k_1 - M/2)\Delta_{\mathrm{k}}, (k_2 - M/2)\Delta_{\mathrm{k}}\Big) \tag{7.8}$$

for $k_1 = 0, \ldots, (M-1)$ and $k_2 = 0, \ldots, (M-1)$. Note that the choice of Δ and Δ_{k} ensures that the support of the pupil function lies within the $(M \times M)$ sampling grid for $\widehat{p}$. Moreover, in our discretization scheme, we assume that the wave vector $\mathbf{k}_{l_{q,w}}$ can be written as $\mathbf{k}_{l_{q,w}} = (b_{l_{q,w},1}\Delta_{\mathrm{k}}, b_{l_{q,w},2}\Delta_{\mathrm{k}})$, where $b_{l_{q,w},1}, b_{l_{q,w},2} \in \mathbb{Z}$.

We now introduce some additional notations to specify the discrete forward model. Let $\mathbf{y}_{q,w} \in \mathbb{R}^{M^2}$, $\mathbf{s}_q \in \mathbb{C}^{N^2}$, and $\widehat{\mathbf{p}} \in \mathbb{C}^{M^2}$ be the vectorized versions of $\mathbf{y}_{q,w}^{\mathrm{im}}$, $\mathbf{s}_q^{\mathrm{im}}$, and $\widehat{\mathbf{p}}^{\mathrm{im}}$, respectively. Then, let $\mathbf{F}_Q, \mathbf{F}_Q^{-1} \in \mathbb{C}^{Q^2 \times Q^2}$ be matrices that represent the 2D DFT and its inverse of a $(Q \times Q)$ image, respectively. Next, we define $\mathbf{diag}(\widehat{\mathbf{p}}) \in \mathbb{C}^{M^2 \times M^2}$ to be a diagonal matrix whose entries are the values in $\widehat{\mathbf{p}}$. Finally, $\mathbf{C}_{\mathbf{k}_{l_{q,w}}}$ is a boolean matrix that restricts an N^2-dimensional vector to an M^2-dimensional vector depending on the illumination wave vector $\mathbf{k}_{l_{q,w}}$.

Proposition 7.1. *The discrete counterpart of* (7.1) *can be computed as*

$$\mathbf{y}_{q,w} = |\mathbf{H}_{l_{q,w}}\mathbf{s}_q|^2 = \left|\frac{4\pi^2}{r_{\mathrm{p}}^2}\mathbf{F}_M^{-1}\mathbf{diag}(\widehat{\mathbf{p}})\mathbf{C}_{\mathbf{k}_{l_{q,w}}}\mathbf{F}_N\mathbf{s}_q\right|^2. \tag{7.9}$$

Proof. Consider the quantity

$$\begin{aligned} u_{q,w}(\mathbf{r}) &= \mathcal{F}^{-1}\left\{\widehat{p}(\mathbf{k})\mathcal{F}\left\{s(\cdot, t_q)\mathrm{e}^{\mathrm{j}\langle \mathbf{k}_{l_{q,w}}, \cdot\rangle}\right\}(\mathbf{k})\right\}(\mathbf{r}) \\ &= \int_{\mathbb{R}^2} \widehat{p}(\mathbf{k})\mathrm{e}^{\mathrm{j}\langle \mathbf{k}, \mathbf{r}\rangle}\left(\int_{\mathbb{R}^2} s(\mathbf{x}, t_q)\mathrm{e}^{-\mathrm{j}\langle \mathbf{k} - \mathbf{k}_{l_{q,w}}, \mathbf{x}\rangle}\mathrm{d}\mathbf{x}\right)\mathrm{d}\mathbf{k}. \end{aligned} \tag{7.10}$$

We discretize the integrals in (7.10) using Riemann sums. A step-size Δ_{k} is used for the integral with respect to $\mathbf{k}$ and a step-size Δ_{r} is used for the integral with respect to *versus.* The samples $\mathbf{u}_{q,w}^{\mathrm{im}}[m_1, m_2] = u_{q,w}\Big((m_1 - M/2)\Delta, (m_2 - M/2)\Delta\Big)$ for $m_1 = 0, \ldots, (M-1)$

and $m_2 = 0, \ldots, (M-1)$, are then given by

$$\mathbf{u}^{\mathrm{im}}_{q,w}[m_1, m_2] = (\Delta_{\mathrm{k}}\Delta_{\mathrm{r}})^2 \sum_{k_1=0}^{M-1}\sum_{k_2=0}^{M-1} \Bigg(\underbrace{\widehat{p}\Big((k_1 - M/2)\Delta_{\mathrm{k}}, (k_2 - M/2)\Delta_{\mathrm{k}}\Big)}_{\widehat{\mathbf{p}}^{\mathrm{im}}[k_1,k_2]} \times \mathrm{e}^{\mathrm{j}(k_1 - M/2)(m_1 - M/2)\Delta_{\mathrm{k}}\Delta}\ \mathrm{e}^{\mathrm{j}(k_2 - M/2)(m_2 - M/2)\Delta_{\mathrm{k}}\Delta}\ \mathbf{a}_{q,w}[k_1, k_2] \Bigg), \quad (7.11)$$

where

$$\mathbf{a}_{q,w}[k_1, k_2] = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} \Bigg(\underbrace{s\Big((n_1 - N/2)\Delta_{\mathrm{r}}, (n_2 - N/2)\Delta_{\mathrm{r}}, t_q\Big)}_{\mathbf{s}^{\mathrm{im}}_q[n_1,n_2]} \times \mathrm{e}^{-\mathrm{j}(k_1 - b_{l_{q,w},1} - M/2)(n_1 - N/2)\Delta_{\mathrm{k}}\Delta_{\mathrm{r}}}\ \mathrm{e}^{-\mathrm{j}(k_2 - b_{l_{q,w},2} - M/2)(n_2 - N/2)\Delta_{\mathrm{k}}\Delta_{\mathrm{r}}} \Bigg). \quad (7.12)$$

The limits in the sums in (7.11) and (7.12) are dictated by the supports of $\widehat{p}$ and $s(\mathbf{r}, t_q)$, respectively. By rearranging some terms and using the fact that $\Delta_{\mathrm{k}}\Delta_{\mathrm{r}} = 2\pi/N$, we rewrite (7.12) as

$$\mathbf{a}_{q,w}[k_1, k_2] = \underbrace{\left(\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} \mathbf{s}^{\mathrm{im}}_q[n_1, n_2]\ \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}(k_1 - b_{l_{q,w},1} - M/2)n_1}\ \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}(k_2 - b_{l_{q,w},2} - M/2)n_2} \right)}_{\widehat{\mathbf{s}}^{\mathrm{im}}_q[k_1 - b_{l_{q,w},1} - M/2,\, k_2 - b_{l_{q,w},2} - M/2]} \times \mathrm{e}^{\mathrm{j}\pi(k_1 - b_{l_{q,w},1} - M/2)}\ \mathrm{e}^{\mathrm{j}\pi(k_2 - b_{l_{q,w},2} - M/2)}, \quad (7.13)$$

where $\widehat{\mathbf{s}}^{\mathrm{im}}_q$ is the (N, N)-point DFT of $\mathbf{s}^{\mathrm{im}}_q$ and the shifts in the DFT are applied in a circular manner. On plugging (7.13) into (7.11), we get that

$$\mathbf{u}^{\mathrm{im}}_{q,w}[m_1, m_2] = (2\pi/N)^2 \sum_{k_1=0}^{M-1}\sum_{k_2=0}^{M-1} \Bigg(\widehat{\mathbf{p}}^{\mathrm{im}}[k_1, k_2]\ \widehat{\mathbf{s}}^{\mathrm{im}}_q[k_1 - b_{l_{q,w},1} - M/2, k_2 - b_{l_{q,w},2} - M/2] \times \mathrm{e}^{\mathrm{j}(k_1 - M/2)(m_1 - M/2)\Delta_{\mathrm{k}}\Delta}\ \mathrm{e}^{\mathrm{j}(k_2 - M/2)(m_2 - M/2)\Delta_{\mathrm{k}}\Delta} \times \mathrm{e}^{\mathrm{j}\pi(k_1 - b_{l_{q,w},1} - M/2)}\ \mathrm{e}^{\mathrm{j}\pi(k_2 - b_{l_{q,w},2} - M/2)} \Bigg). \quad (7.14)$$

Next, we group all the exponential terms involving k_1 and k_2 and use $\Delta_{\mathrm{k}}\Delta = 2\pi/M$ to obtain that

$$\mathbf{u}^{\mathrm{im}}_{q,w}[m_1,m_2] = (2\pi/N)^2 \left(\sum_{k_1=0}^{M-1}\sum_{k_2=0}^{M-1} \widehat{\mathbf{p}}^{\mathrm{im}}[k_1,k_2]\ \widehat{\mathbf{s}}^{\mathrm{im}}_q[k_1 - b_{l_{q,w},1} - M/2, k_2 - b_{l_{q,w},2} - M/2] \right.$$
$$\left. \times\, \mathrm{e}^{\mathrm{j}\frac{2\pi}{M}k_1 m_1}\ \mathrm{e}^{\mathrm{j}\frac{2\pi}{M}k_2 m_2} \right) \times \mathrm{e}^{-\mathrm{j}\pi(m_1+b_{l_{q,w},1})}\ \mathrm{e}^{-\mathrm{j}\pi(m_2+b_{l_{q,w},2})}. \tag{7.15}$$

Let $\mathbf{g}^{\mathrm{im}}_{q,w}$ be the (M,M)-point IDFT of $\widehat{\mathbf{g}}^{\mathrm{im}}_{q,w}[k_1,k_2] = \widehat{\mathbf{p}}^{\mathrm{im}}[k_1,k_2]\ \widehat{\mathbf{s}}^{\mathrm{im}}_q[k_1 - b_{l_{q,w},1} - M/2, k_2 - b_{l_{q,w},2} - M/2]$. Then, the discrete measurements can be expressed as

$$\mathbf{y}^{\mathrm{im}}_{q,w}[m_1,m_2] = \left|\mathbf{u}^{\mathrm{im}}_{q,w}[m_1,m_2]\right|^2 = \left|(4\pi^2/r_{\mathrm{p}}^2)\ \mathbf{g}^{\mathrm{im}}_{q,w}[m_1,m_2]\right|^2. \tag{7.16}$$

Note that the computation of $\mathbf{g}^{\mathrm{im}}_{q,w}$ involves taking the (N,N)-point DFT of $\mathbf{s}^{\mathrm{im}}_q$, (circularly) shifting it according to the wave vector $\mathbf{k}_{l_{q,w}}$, restricting the shifted DFT to an $(M \times M)$ image, performing pointwise multiplication with $\widehat{\mathbf{p}}^{\mathrm{im}}$, and then taking the (M,M)-point IDFT. This allows us to write (7.16) in vectorized form as in the right-hand side of (7.9). $\square$

While the discrete forward model (7.9) has previously been used in works such as [310], to the best of our knowledge, a systematic derivation of (7.9) from the continuous model (7.1) has not been presented in the literature.

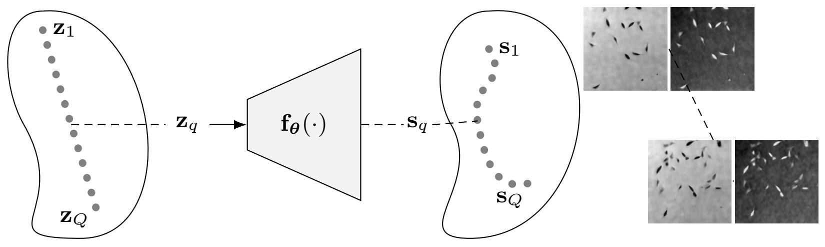
7.3 Reconstruction Framework

The goal in dynamic FP is to reconstruct the images $\{\mathbf{s}_q \in \mathbb{C}^{N^2}\}_{q=1}^{Q}$ from the recorded measurements $\{\{\widetilde{\mathbf{y}}_{q,w} \in \mathbb{R}^{M^2}\}_{w=1}^{W}\}_{q=1}^{Q}$. We first present our neural-network-based framework for the case of a well-characterized pupil function. Then, we describe a way to incorporate the recovery of the pupil function into our reconstruction algorithm.

7.3.1 Deep Spatiotemporal Regularization

In our framework, we propose to use an extended version of the untrained-neural-network-based method presented in Section 2.2.3 to impose spatiotemporal regularization on the sequence of images. We parameterize each of the Q images as the output of a single CNN $\mathbf{f}_{\boldsymbol{\theta}} : \mathbb{R}^{N_z^2} \to \mathbb{C}^{N^2}$, with parameters $\boldsymbol{\theta} \in \mathbb{R}^P$, applied to some fixed input latent vector $\mathbf{z}_q \in \mathbb{R}^{N_z^2}$, $q = 1,\dots,Q$. We choose these latent vectors such that they lie on a straight line, in accordance with

$$\mathbf{z}_q = \mathbf{z}_1 + \frac{q-1}{Q-1}\left(\mathbf{z}_Q - \mathbf{z}_1\right), \qquad q = 1,\dots,Q, \tag{7.17}$$



Optimization problem: $\boldsymbol{\theta}^* \in \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^P} \sum_{q=1}^{Q} \sum_{w=1}^{W} \mathcal{D}\Big(\widetilde{\mathbf{y}}_{q,w}, |\mathbf{H}_{l_{q,w}} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_q)|^2\Big)$

Figure 7.2: Spatiotemporal regularization using the generative neural network $\mathbf{f}_{\boldsymbol{\theta}}$.

where the end-points $\mathbf{z}_1, \mathbf{z}_Q$ are fixed beforehand (for example, by drawing two samples from some multivariate probability distribution). We then estimate the parameters of the network according to

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^P} \sum_{q=1}^{Q} \sum_{w=1}^{W} \mathcal{D}\Big(\widetilde{\mathbf{y}}_{q,w}, |\mathbf{H}_{l_{q,w}} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_q)|^2\Big), \tag{7.18}$$

where $\mathcal{D} : \mathbb{R}^{M^2} \times \mathbb{R}^{M^2} \to \mathbb{R}_+$ is a data-fidelity term derived from a suitable statistical model for the Noise$(\cdot)$ operator (like in Equation 2.8) and the reconstructed sequence is $\{\mathbf{s}_q^*\}_{q=1}^Q = \{\mathbf{f}_{\boldsymbol{\theta}^*}(\mathbf{z}_q)\}_{q=1}^Q$. The rationale behind our choice of the latent vectors is to allow the CNN to associate the spatial proximity between them with the temporal proximity of the images. In this manner, the architecture of the network imposes spatial regularization while the use of a shared network for all images and the design of the latent space impose temporal regularization. A schematic illustration of our framework is given in Figure 7.2.

7.3.2 Optimization Strategy

The relation between the measurements and the underlying images is nonlinear, which makes the inverse problem very challenging. The fact that only one LED is switched on for each measurement further adds to the difficulty. Thus, in order to avoid bad local minima while solving the optimization problem in (7.18), we initialize the parameters of the network according to

$$\widetilde{\boldsymbol{\theta}} \in \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^P} \sum_{q=1}^{Q} \left(\Big\| |\widetilde{\mathbf{s}}_q| - |\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_q)| \Big\|_1 + \Big\| \arg\Big(\widetilde{\mathbf{s}}_q\Big) - \arg\Big(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_q)\Big) \Big\|_1 \right), \tag{7.19}$$

where $\{\widetilde{\mathbf{s}}_q\}_{q=1}^Q$ are low-quality reconstructions obtained via a standard frame-by-frame method. The magnitude $|\cdot|$ and phase $\arg(\cdot)$ operations in (7.19) are applied component-wise. We can solve (7.19) using off-the-shelf minibatch stochastic gradient-descent algorithms. However, it is not desirable to run these algorithms till convergence as the

Algorithm 4 Initialization of network parameters.

```
Input: Low-quality reconstructions $\{\widetilde{\mathbf{s}}_q\}_{q=1}^Q$, latent vectors $\{\mathbf{z}_q\}_{q=1}^Q$, batch size $B_Q$, tolerance $\epsilon_{\text{tol}}$, maximum number of iterations $n_{\max}$.
Randomly initialize $\boldsymbol{\theta}$
$\mathcal{L}_{\text{batch}} \leftarrow +\infty$, $i \leftarrow 0$
while $\mathcal{L}_{\text{batch}} > \epsilon_{\text{tol}}$ do
    Randomly sample a batch $\mathcal{Q}$ of size $B_Q$ from $\{1, 2, \dots, Q\}$
    Compute $\mathcal{L}_{\text{batch}}(\boldsymbol{\theta}) = \sum_{q\in\mathcal{Q}} \left( \left\| |\widetilde{\mathbf{s}}_q| - |\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_q)| \right\|_1 + \left\| \arg\left(\widetilde{\mathbf{s}}_q\right) - \arg\left(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_q)\right) \right\|_1 \right)$
    Update $\boldsymbol{\theta}$ with gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}}\mathcal{L}_{\text{batch}}(\boldsymbol{\theta})$
    $i \leftarrow i + 1$
    if $i > n_{\max}$ then
        Exit the while loop
    end if
end while
Output: Network parameters $\boldsymbol{\theta}$
```

network then overfits the artifacts present in the low-quality reconstructions. Thus, in our initialization routine, which is described in Algorithm 4, we deploy early stopping by choosing suitable values for the tolerance ϵ_{tol} and the maximum number of iterations $n_{\max}$ (see Section 7.4.1 for details).

After the initialization, we can solve (7.18) using again some minibatch stochastic gradient-descent algorithm. In some cases (for example, when the measurements are corrupted by a non-negligible amount of noise), running the optimization process beyond a certain number of iterations leads to deterioration of the reconstruction quality as the network begins to overfit the measurements. Thus, we also adopt early stopping when necessary.

For both the initialization and reconstruction tasks, we use (minibatch) stochastic gradient-descent algorithms instead of deterministic ones. This introduces additional hyperparameters (batch sizes) that must be set appropriately. However, stochastic methods with small batch sizes require much less memory than the deterministic ones. In fact, if the number of frames Q is large, applying a deteministic gradient-descent method is infeasible. Further, such stochastic methods are also more likely to escape bad local minima and thus reach better solutions. Indeed, in our experiments, we observed that using reasonably small batch sizes ($B_Q = 10$) led to better reconstructions than using large batch sizes ($B_Q = 40$).

7.3.3 Recovery of the Pupil Function

So far, we have assumed complete knowledge of the pupil function in our reconstruction framework. However, the pupil function is typically not well-characterized in FP. Thus, similar to the work in [311, 319], we estimate it along with the sequence of images.

Algorithm 5 Joint recovery of dynamic sample and pupil function.

```
Input: Measurements $\{\{\widetilde{\mathbf{y}}_{q,w}\}_{w=1}^{W}\}_{q=1}^{Q}$, LED indices $\{\{l_{q,w}\}_{w=1}^{W}\}_{q=1}^{Q}$, latent vectors $\{\mathbf{z}_q\}_{q=1}^{Q}$, initial network parameters $\widetilde{\boldsymbol{\theta}}$, initial Zernike coefficients $\widetilde{\mathbf{c}}$, regularization parameter $\tau$, batch sizes $\{B_W, B_Q\}$, number of epochs $n_{\mathrm{ep}}$.
$\boldsymbol{\theta} \leftarrow \widetilde{\boldsymbol{\theta}}$, $\mathbf{c} \leftarrow \widetilde{\mathbf{c}}$
$n_W \leftarrow \lfloor \frac{W}{B_W} \rfloor$, $n_Q \leftarrow \lfloor \frac{Q}{B_Q} \rfloor$
for $n_{\mathrm{ep}}$ epochs do
    for $n_W$ iterations do
        Randomly sample a batch $\mathcal{W}$ of size $B_W$ from $\{1, 2, \dots, W\}$
        for $n_Q$ iterations do
            Randomly sample a batch $\mathcal{Q}$ of size $B_Q$ from $\{1, 2, \dots, Q\}$
            Compute the loss $\mathcal{L}_{\mathrm{batch}}(\boldsymbol{\theta}, \mathbf{c}) = \sum_{q\in\mathcal{Q}} \sum_{w\in\mathcal{W}} \mathcal{D}\Big(\widetilde{\mathbf{y}}_{q,w}, |\mathbf{H}_{l_{q,w}}(\mathbf{c})\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_q)|^2\Big)$
            Update $\boldsymbol{\theta}$ with gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{batch}}(\boldsymbol{\theta}, \mathbf{c})$
            Update $\mathbf{c}$ with gradient $\boldsymbol{\nabla}_{\mathbf{c}}\mathcal{L}_{\mathrm{batch}}(\boldsymbol{\theta}, \mathbf{c})$
        end for
    end for
end for
Output: Reconstructed images $\{\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_q)\}_{q=1}^{Q}$, Zernike coefficients $\mathbf{c}$
```

Following [319], we use Zernike polynomials to represent the pupil function with only a few parameters ($\ll M^2$). These functions are orthogonal on the unit circle and are often used in optics for modeling aberrations. We express the pupil function in polar coordinates (ρ, ϕ) as

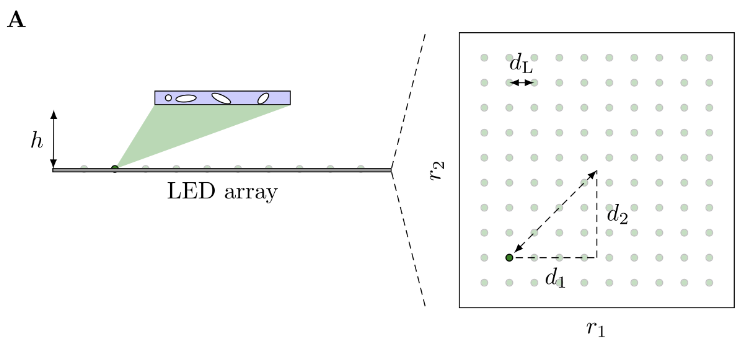
$$\widehat{p}(\rho, \phi) = \begin{cases} \exp\left(\mathrm{j} \sum\limits_{a=1}^{A} c_a Z_a\left(\frac{\rho\lambda}{2\pi\mathrm{NA}}, \phi\right)\right), \rho \leq \frac{2\pi\mathrm{NA}}{\lambda} \\ 0, \qquad\qquad\qquad\qquad\qquad \text{otherwise,} \end{cases} \tag{7.20}$$

where Z_a is the ath Zernike polynomial according to Noll's sequential indices (refer to 7.6.1 for details) and $\mathbf{c} = (c_a)_{a=1}^{A} \in \mathbb{R}^A$ ($A \ll M^2$) contains the Zernike coefficients. The pupil function is discretized as in (7.8) by evaluating (7.20) on the required Cartesian grid. We denote the vectorized discrete pupil function by $\widehat{\mathbf{p}}(\mathbf{c}) \in \mathbb{R}^{M^2}$ to explicitly indicate the dependence on the Zernike coefficients. Similarly, our forward model (7.9) is then written as

$$\mathbf{y}_{q,w} = |\mathbf{H}_{l_{q,w}}(\mathbf{c})\mathbf{s}_q|^2 = \left|\frac{4\pi^2}{r_{\mathrm{p}}^2}\mathbf{F}_M^{-1}\mathbf{diag}(\widehat{\mathbf{p}}(\mathbf{c}))\mathbf{C}_{\mathbf{k}_{l_{q,w}}}\mathbf{F}_N\mathbf{s}_q\right|^2. \tag{7.21}$$

Finally, the optimization problem for the joint recovery of the pupil function and the sequence of images is

$$(\boldsymbol{\theta}^*, \mathbf{c}^*) \in \underset{\boldsymbol{\theta}\in\mathbb{R}^P, \mathbf{c}\in\mathbb{R}^A}{\arg\min} \sum_{q=1}^{Q}\sum_{w=1}^{W} \mathcal{D}\Big(\widetilde{\mathbf{y}}_{q,w}, |\mathbf{H}_{l_{q,w}}(\mathbf{c})\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}_q)|^2\Big), \tag{7.22}$$



$$\boxed{\mathbf{k}_l = \frac{2\pi}{\lambda}\left(\sin\left(\arctan\left(\frac{d_1}{h}\right)\right), \sin\left(\arctan\left(\frac{d_2}{h}\right)\right)\right)}$$

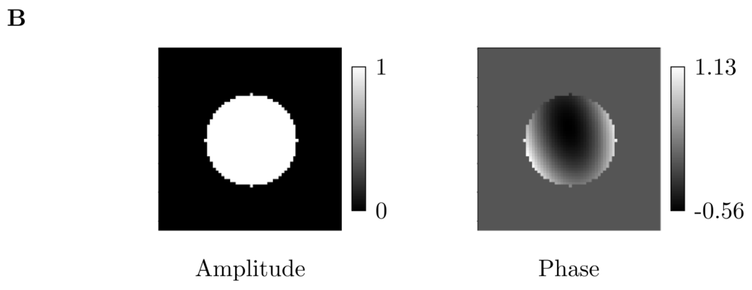


Figure 7.3: Simulated FP setup. Panel A: LED array. Panel B: Pupil function.

where $\mathcal{D} : \mathbb{R}^{M^2} \times \mathbb{R}^{M^2} \to \mathbb{R}_+$ is the data-fidelity term. We can solve (7.22) using a minibatch stochastic gradient-descent algorithm coupled with early stopping if required. Our complete reconstruction algorithm is summarized in Algorithm 5.

7.4 Numerical Results

7.4.1 Simulated Setup

We demonstrate the advantages of our reconstruction method on simulated data. We consider an FP setup consisting of $L = 100$ LEDs arranged in a (10×10) uniform grid with a spacing of $d_{\mathrm{L}} = 4\,\mathrm{mm}$. The maximum illumination NA of the LED array, which is placed at distance $h = 90.88\,\mathrm{mm}$ from the sample, is 0.27. The LEDs emit light with wavelength $\lambda = 532\,\mathrm{nm}$. The numerical aperture of the objective is $\mathrm{NA} = 0.1$. We have chosen these values of $d_{\mathrm{L}}, h, \lambda$ and NA based on the experimental setup in [314]. The pupil function is defined according to (7.20) using the first nine Zernike polynomials with coefficients $\mathbf{c} = (0, 0.15, 0.3, -0.1, 0.2, 0, 0, 0, 0) \in \mathbb{R}^9$. We take the low-resolution measurements acquired by the camera to be of size (64×64) with pixel-size $\Delta = \frac{\lambda}{4\mathrm{NA}} = 1.33\,\mu\mathrm{m}$ and we set the oversampling ratio as $r_{\mathrm{p}} = 4$. Consequently, the pixel

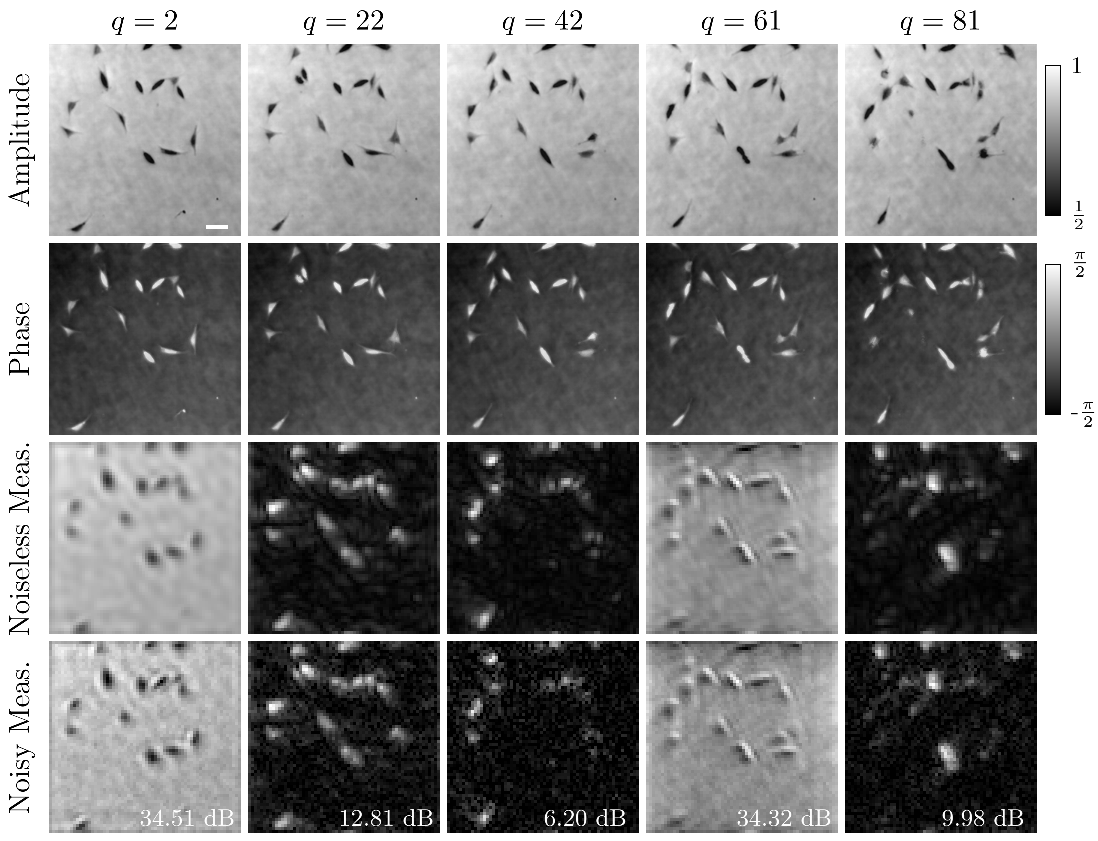


Figure 7.4: First and second row: frames of the ground-truth sequence (amplitude and phase). Third and fourth row: corresponding low-resolution measurements (noiseless and noisy, normalized for visualization). The signal-to-noise ratios for the noisy measurements, computed as $20\log_{10}\frac{\|\mathbf{y}_q\|_2}{\|\mathbf{y}_q-\widetilde{\mathbf{y}}_q\|_2}$, are indicated at the bottom right corners of the measurement images. Scale bar: 10 µm.

size for the highresolution image is $\Delta_{\mathrm{r}} = 332.5\,\mathrm{nm}$ and the step-size for discretizing the pupil function is $\Delta_{\mathrm{k}} = 0.074\,\mu\mathrm{m}^{-1}$. The LED array and the pupil function are shown in Figure 7.3.

Our ground truth is a sequence of complex-valued images $\{\mathbf{s}_q \in \mathbb{C}^{256^2}\}_{q=1}^{100}$ of size (256×256) which we created from experimental phase images[4]. We place ourselves in the extremely challenging ultrafast regime where only one measurement is acquired for each image in the sequence. For each measurement, a single LED of index[5] l_q is randomly activated and a low-resolution image[5] $\mathbf{y}_q \in \mathbb{R}^{64^2}$ is simulated according to (7.21). The recorded measurement image $\widetilde{\mathbf{y}}_q \in \mathbb{R}^{64^2}$ is then generated according to

$$\widetilde{\mathbf{y}}_q = \mathbf{y}_q + \mathbf{n}_q, \tag{7.23}$$

[4]The experimental phase images are from [331] and are available at http://celltrackingchallenge.net/2d-datasets/.

[5]We have dropped the index w as $W = 1$.

where $\mathbf{n}_q \in \mathbb{R}^{64^2}$ is a realization of a zero-mean Gaussian random vector with covariance matrix $\mathbf{\Sigma}_q \in \mathbb{R}^{64^2 \times 64^2}$. Specifically, we consider two settings for our simulations. In the first case, $\mathbf{\Sigma}_q$ is the zero matrix, which means that the recorded measurements are noiseless. In the second case, $\mathbf{\Sigma}_q$ is a diagonal matrix with entries $\Big(([\mathbf{y}_q]_m)/1000\Big)_{m=1}^{64^2}$. There, (7.23) corresponds to a Gaussian approximation of the Poisson noise model with a photon budget of 1000.

We show some of the frames in the ground-truth sequence and the corresponding measurements for both the settings (noiseless and noisy) in 7.4. The full sequences are provided in the supplementary material.

7.4.2 Implementation of the Deep Spatiotemporal Regularizer

In this subsection, we describe the implementation of our reconstruction method—the deep spatiotemporal regularizer (DSTR).

Network Architecture

It has been observed that the choice of the network architecture can greatly affect the performance of untrained-neural-network-based methods [82]. Therefore, the common practice when deploying such schemes is to select the architecture in an empirical trial-and-error manner for the specific task at hand. For our experiments, inspired by [330], we adopt a convolutional decoder-like architecture for $\mathbf{f}_{\boldsymbol{\theta}}$, which, as we demonstrate in Sections 7.4.5 and 7.4.6, yields high-quality reconstructions. It takes a low-dimensional input vector $\mathbf{z} \in \mathbb{R}^{8^2}$ and outputs a complex-valued (vectorized) image $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}) \in \mathbb{C}^{256^2}$. The architectural details are described in Table 7.1. In particular, the complex-valued image is generated from a pair of magnitude and phase images. The initial part of the network creates feature maps of size $(128 \times 256 \times 256)$. These are then fed into both the magnitude and phase branches of the network. The magnitude branch consists of a convolutional layer followed by the pointwise differentiable rectified linear unit (DReLU) activation function, which we define as

$$\mathrm{DReLU}(x) = \begin{cases} \gamma \exp(\frac{x}{\gamma} - 1), & x < \gamma \\ x, & \text{otherwise}, \end{cases} \tag{7.24}$$

where $\gamma > 0$ is set a priori. We use DReLU (with $\gamma = 0.1$) instead of ReLU to avoid the "dead-neuron" issue during the first few iterations of the optimization, while ensuring that the magnitude is positive. Meanwhile, the phase branch consists of a convolutional layer followed by the $\pi \tanh$ nonlinearity to constrain the phase to lie within the range $[-\pi, \pi]$.

Layers	Output shape
Reshape	$1 \times 8 \times 8$
$2 \times$ (Conv + BN + ReLU)	$128 \times 8 \times 8$
Upsampling + $2 \times$ (Conv + BN + ReLU)	$128 \times 16 \times 16$
Upsampling + $2 \times$ (Conv + BN + ReLU)	$128 \times 32 \times 32$
Upsampling + $2 \times$ (Conv + BN + ReLU)	$128 \times 64 \times 64$
Upsampling + $2 \times$ (Conv + BN + ReLU)	$128 \times 128 \times 128$
Upsampling + $2 \times$ (Conv + BN + ReLU)	$128 \times 256 \times 256$
Magnitude: Conv + DReLU	$1 \times 256 \times 256$
Phase: Conv + $\pi \tanh$	$1 \times 256 \times 256$
Combination: Magnitude $\odot$ $\mathrm{e}^{\mathrm{j}\mathrm{Phase}}$	$1 \times 256 \times 256$
Reshape	1×256^2

Table 7.1: Architecture of the network $\mathbf{f}_{\boldsymbol{\theta}}$. Size of input: (1×8^2). Conv: convolutional layer with (3×3) kernels and reflective boundary conditions. BN: batch normalization layer. Upsampling: nearest neighbor interpolation. The amplitude and phase branches take the same input of size $(128 \times 256 \times 256)$ and output the magnitude and phase images of size $(1 \times 256 \times 256)$, respectively. DReLU is described in (7.24). The combination layer generates a complex-valued image from the magnitude and phase images. This network consists of 1,628,546 learnable parameters.

Latent Vectors

As mentioned in Section 7.3.1, the latent vectors $\{\mathbf{z}_q \in \mathbb{R}^{8^2}\}_{q=1}^{100}$ are chosen such that they lie on the straight line defined in (7.17). We fix the end-points $\mathbf{z}_1, \mathbf{z}_{100}$ of this line by drawing two samples from the standard multivariate normal distribution in 8^2 dimensions.

Initialization

In all our experiments, we initialize the parameters of the network using reconstructions obtained from the Gerchberg-Saxton algorithm (briefly described in Section 7.4.3). We run Algorithm 4 using the AMSGrad solver [332] with a learning rate of 10^{-3}, batch size $B_Q = 10$, tolerance $\epsilon_{\mathrm{tol}} = 0.1 \times (B_Q \times 256^2)$ and maximum number of iterations $n_{\max} = 1000$. We then freeze the tunable parameters of the batch-normalization layers. For experiments involving the estimation of the pupil function, we initialize the Zernike coefficients as $\widetilde{\mathbf{c}} = \mathbf{0}$.

We have observed that the initialization of the network parameters has an impact on the reconstruction quality. For example, randomly initializing the parameters does not lead to satisfactory results. However, initializing the network by simply fitting it to low-quality solutions of the Gerchberg-Saxton algorithm (along with early stopping to avoid overfitting the artifacts) allows us to obtain excellent reconstructions (see Sections 7.4.5 and 7.4.6).

Choice of the Data-Fidelity Term

In our simulations, we consider two kinds of measurements—noiseless and those corrupted by (an approximation of) Poisson noise. For the latter, the data-fidelity term based on the negative log-likelihood function corresponding to the Poisson distribution is given by

$$\mathcal{D}_{\text{Poisson}}(\mathbf{a},\mathbf{b}) = \sum_{m=1}^{M} \Big(- [\mathbf{a}]_m \log([\mathbf{b}]_m) + [\mathbf{b}]_m \Big), \tag{7.25}$$

where $\mathbf{a},\mathbf{b} \in \mathbb{R}^{M^2}$. However, for optimization-based FP reconstruction, it has been shown (experimentally) in [310] that a cost function of the form

$$\mathcal{D}_{\text{sqrt}}(\mathbf{a},\mathbf{b}) = \frac{1}{2}\left\|\sqrt{\mathbf{a}} - \sqrt{\mathbf{b}}\right\|_2^2 \tag{7.26}$$

yields better reconstructions than the one in (7.25). Thus, for our reconstruction experiments, we use the slightly modified version of (7.26) given by

$$\mathcal{D}(\mathbf{a},\mathbf{b}) = \frac{1}{2}\left\|\sqrt{\mathbf{a}+\epsilon\mathbf{1}} - \sqrt{\mathbf{b}+\epsilon\mathbf{1}}\right\|_2^2, \tag{7.27}$$

where $\mathbf{1} \in \mathbb{R}^{M^2}$ is a vector with all entries equal to 1 and $\epsilon = 10^{-10}$ helps us avoid numerical instabilities in the computation of the gradient. We would like to mention that $\mathcal{D}_{\text{sqrt}}$ can be interpreted from a statistical point of view as being based on the negative log-likelihood function derived from the distribution of the transformed measurements $\sqrt{\tilde{\mathbf{y}}_q}$. This is due to the well-known property that if X is a Poisson random variable with mean μ_{X}, then $\mathsf{Y} = \sqrt{\mathsf{X}}$ approximately follows a Gaussian distribution with mean $\mu_{\mathsf{Y}} = \sqrt{\mu_{\mathsf{X}}}$ and variance $\sigma_{\mathsf{Y}}^2 = 1/4$ [333, 334].

For the noiseless setting, ideally we should optimize the network parameters such that the generated sequence fits the measurements exactly. However, it is difficult to solve this constrained optimization problem and thus we use the data-fidelity term from (7.27) in this case as well.

Note: The details regarding the optimization process for (7.18) and (7.22) are provided in Sections 7.4.5 and 7.4.6.

7.4.3 Comparisons

We compare our proposed framework to the following methods.

Gerchberg-Saxton Algorithm The GS algorithm [302] is a classical method for phase retrieval. Assuming that the Zernike coefficient vector $\mathbf{c}$ is known, it aims at solving the feasibility problem

$$\mathbf{s}^*_{\mathrm{GS},q} \in \left\{\mathbf{s} : \widetilde{\mathbf{y}}_q = |\mathbf{H}_{l_q}(\mathbf{c})\mathbf{s}|^2\right\} \tag{7.28}$$

for $q = 1, 2, \dots, 100$, by alternately updating the image plane and the object plane. We refer the reader to [302] for more details. When the pupil function is not well-characterized, we do not incorporate its recovery within the GS algorithm. Instead, we solve (7.28) assuming an idealized pupil function with no phase aberrations that corresponds to $\mathbf{c} = \mathbf{0}$.

Data-Consistency Estimator Based on the work in [310], we consider a data-consistency (DC) estimator that minimizes the (slightly modified) "amplitude-based" cost function (7.27). For the joint recovery of the images and pupil function, it is given by

$$\left(\mathbf{s}^*_{\mathrm{DC},1}, \dots, \mathbf{s}^*_{\mathrm{DC},100}, \mathbf{c}^*_{\mathrm{DC}}\right) \in \arg \min_{\mathbf{s}_1,\dots,\mathbf{s}_{100},\mathbf{c}} \sum_{q=1}^{100} \mathcal{D}\Big(\widetilde{\mathbf{y}}_q, |\mathbf{H}_{l_q}(\mathbf{c})\mathbf{s}_q|^2\Big), \tag{7.29}$$

where $\mathcal{D}(\cdot,\cdot)$ is defined in (7.27).

Spatially Total-Variation-Regularized Estimator In our numerical simulations, we also consider a regularized estimator where the cost function in (7.29) is augmented with spatial anisotropic TV regularization for each frame. It is given by

$$\begin{aligned}\left(\mathbf{s}^*_{\mathrm{STV},1}, \dots, \mathbf{s}^*_{\mathrm{STV},100}, \mathbf{c}^*_{\mathrm{STV}}\right) \in \arg \min_{\mathbf{s}_1,\dots,\mathbf{s}_{100},\mathbf{c}} \sum_{q=1}^{100} \Bigg(\mathcal{D}\Big(\widetilde{\mathbf{y}}_q, |\mathbf{H}_{l_q}(\mathbf{c})\mathbf{s}_q|^2\Big) \\ + \tau_{\mathrm{amp},q}\Big\|\mathbf{L}\{|\mathbf{s}_q|\}\Big\|_1 + \tau_{\mathrm{phase},q}\Big\|\mathbf{L}\{\arg(\mathbf{s}_q)\}\Big\|_1\Bigg),\end{aligned} \tag{7.30}$$

where the operator $\mathbf{L} : \mathbb{R}^N \to \mathbb{R}^{2N}$ computes finite differences in both the directions for the underlying image, and $\{\tau_{\mathrm{amp},q}, \tau_{\mathrm{phase},q}\}_{q=1}^{100} \subset \mathbb{R}_+$ are hyperparameters that control the strength of the regularization.

Spatiotemporally Total-Variation-Regularized Estimator Finally, we also implement a spatiotemporally-regularized estimator where the cost function in (7.29) is augmented with both spatial and temporal TV regularization. It is given by

$$
\begin{aligned}
\left(\mathbf{s}^*_{\mathrm{STTV},1}, \ldots, \mathbf{s}^*_{\mathrm{STTV},100}, \mathbf{c}^*_{\mathrm{STTV}}\right) \in \arg \min_{\mathbf{s}_1,\ldots,\mathbf{s}_{100},\mathbf{c}} & \sum_{q=1}^{100} \left(\mathcal{D}\left(\widetilde{\mathbf{y}}_q, |\mathbf{H}_{l_q}(\mathbf{c})\mathbf{s}_q|^2\right) \right.\\
& \left. + \tau_{\mathrm{amp},s} \left\| \mathbf{L}\{|\mathbf{s}_q|\} \right\|_1 + \tau_{\mathrm{phase},s} \left\| \mathbf{L}\{\arg(\mathbf{s}_q)\} \right\|_1 \right)\\
& + \sum_{q'=1}^{99} \left(\tau_{\mathrm{amp},t} \left\| |\mathbf{s}_{q'+1}| - |\mathbf{s}_{q'}| \right\|_1 \right.\\
& \left. + \tau_{\mathrm{phase},t} \left\| \arg(\mathbf{s}_{q'+1}) - \arg(\mathbf{s}_{q'}) \right\|_1 \right),
\end{aligned} \tag{7.31}
$$

where $\mathbf{L} : \mathbb{R}^N \to \mathbb{R}^{2N}$ is the finite-difference operator and $\{\tau_{\mathrm{amp},s}, \tau_{\mathrm{phase},s}, \tau_{\mathrm{amp},t}, \tau_{\mathrm{phase},t}\} \subset \mathbb{R}_+$ are the regularization hyperparameters.

7.4.4 Evaluation Metric

We quantify the performance of a method by computing the regressed signal-to-noise ratio (RSNR) for the entire reconstructed sequence of images. Let $\bar{\mathbf{s}}$ and $\bar{\mathbf{s}}^*$ denote vectorized versions of the ground truth and reconstruction, respectively. These are created by concatenating the vectorized representations of each frame in the sequence. The RSNR is computed as

$$
\mathrm{RSNR}(\bar{\mathbf{s}}^*, \bar{\mathbf{s}}) = \max_{a \in \mathbb{C}} 20 \log_{10} \frac{\|\bar{\mathbf{s}}\|_2}{\|\bar{\mathbf{s}} - a\bar{\mathbf{s}}^*\|_2}. \tag{7.32}
$$

We also report the SNR for the pupil function whenever it is jointly estimated with the sequence of images. This metric is computed as

$$
\mathrm{SNR}\left(\widehat{\mathbf{p}}(\mathbf{c}), \widehat{\mathbf{p}}(\mathbf{c}^*)\right) = 20 \log_{10} \frac{\|\widehat{\mathbf{p}}(\mathbf{c})\|_2}{\|\widehat{\mathbf{p}}(\mathbf{c}) - \widehat{\mathbf{p}}(\mathbf{c}^*)\|_2}, \tag{7.33}
$$

where $\mathbf{c}$ and $\mathbf{c}^*$ are the ground-truth and estimated Zernike coefficients, respectively.

7.4.5 Reconstruction from Noiseless Measurements

We now present two experiments involving noiseless measurements. In both of them, we run the iterative algorithm for each method for a sufficient number of iterations (details are provided below), beyond which the reconstruction does not change significantly. In other words, we do not deploy early stopping for any method as the measurements are noiseless.

Table 7.2: Reconstruction from noiseless measurements with a perfectly characterized pupil function.

Method	GS	DC	STV	STTV	DSTR
RSNR [dB]	17.24	9.66	17.85	18.58	**28.61**

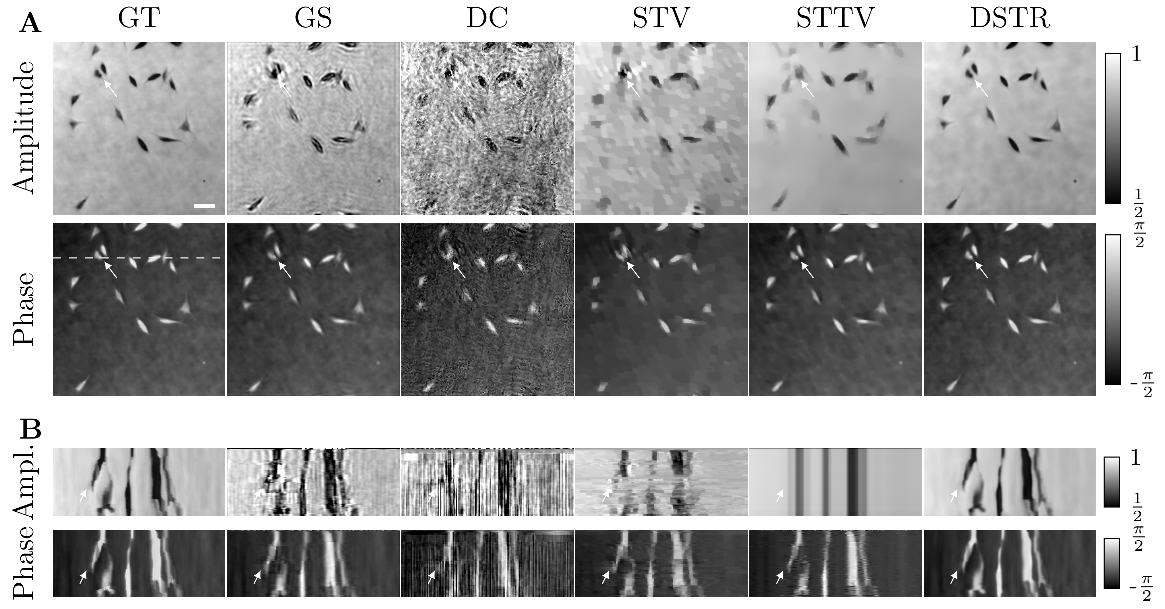


Figure 7.5: Reconstruction from noiseless measurements with a perfectly characterized pupil function. Panel A: XY view for the frame index $q = 26$. Panel B: XT view for the Y position indicated in Panel A (GT, Phase, dashed line). Scale bar: 10 µm.

Perfectly Characterized Pupil Function

We first consider an idealized setting where the pupil function is perfectly characterized and is therefore not estimated during the reconstruction of the images of interest. In this scenario, the DC and STV estimators can be computed in frame-by-frame manner (similar to the GS method) by decomposing the overall optimization problems into $Q = 100$ smaller ones. We solve these by running AMSGrad with a learning rate of 10^{-3} for 1,000 iterations. In order to improve their performance, we initialize the GS, DC, and STV methods for the timestamp t_q with the reconstructed images from the previous timestamp t_{q-1}. The GS solution is used for initializing the STTV method. We solve (7.31) by using AMSGrad for 10,000 epochs with a learning rate of 10^{-3} and a full batch size of 100. The optimal hyperparameters $\{\tau_{\text{amp},q}, \tau_{\text{phase},q}\}_{q=1}^{100}$ and $\{\tau_{\text{amp},s}, \tau_{\text{phase},s}, \tau_{\text{amp},t}, \tau_{\text{phase},t}\}$ for the STV and STTV methods, respectively, are chosen via a grid-search. For DSTR, the network parameters are initialized with the help of the GS solution. We then solve (7.18) by running the AMSGrad optimizer for 10,000 epochs with a learning rate of 5×10^{-5}

Table 7.3: Joint recovery of the dynamic sample and the pupil function from noiseless measurements.

Method	GS	DC	STV	STTV	DSTR
Sequence RSNR [dB]	14.70	14.82	15.88	17.10	**28.04**
Pupil SNR [dB]	N.A.	8.28	9.57	12.74	**31.22**

and a batch size of $B_Q = 10$.

We present the RSNR values for all the methods in Table 7.2. Further, we display some slices of the (2D + time) reconstructions in Figure 7.5. The entire reconstructed sequences can be found at the link given in Appendix 7.6.2. We observe that the proposed method significantly outperforms the GS, DC, STV and STTV methods. Even though only one measurement is acquired per frame, it yields a high-quality reconstruction, unlike the other methods which exhibit various artifacts (for example, the features marked by arrows in Figure 7.5).

Joint Recovery of Dynamic Sample and Pupil Function

Next, we consider a setting where the pupil function is not well-characterized and is therefore estimated jointly with the dynamic sample in our framework and in the DC, STV and STTV methods. (We do not adapt the GS algorithm for the recovery of the pupil function; we simply assume the idealized pupil function $\mathbf{c} = \mathbf{0}$.) For the DC, STV and STTV methods, the sequence of images is initialized with the GS solution and the Zernike coefficients are initialized as $\widetilde{\mathbf{c}} = \mathbf{0}$. We solve (7.29) and (7.30) by running the AMSGrad optimizer for 10,000 epochs with a learning rate of 10^{-3} and a batch size of 10. For solving (7.31), we run AMSGrad for 10,000 epochs with a learning rate of 10^{-3} and a full batch size of 100. In the STV method, we select two global hyperparameters $\{\tau_{\text{amp}}, \tau_{\text{phase}}\}$ via grid search and share them among all frames. The hyperparameters $\{\tau_{\text{amp},s}, \tau_{\text{phase},s}, \tau_{\text{amp},t}, \tau_{\text{phase},t}\}$ for the STTV method are also tuned for best performance with the help of a grid search. In our method, we initialize the network parameters using the GS solution and we initialize the Zernike coefficients as $\widetilde{\mathbf{c}} = \mathbf{0}$. We solve (7.18) by running AMSGrad for 10,000 epochs with a learning rate of 5×10^{-5} and a batch size of $B_Q = 10$.

We present the RSNR and SNR values for the reconstructed sequence and the pupil function, respectively, in Table 7.3. We also show some slices of the (2D + time) reconstructions and the recovered pupil functions (phase) in Figure 7.6, as well as the recovered Zernike coefficients in Figure 7.7. The full reconstructed sequences are provided at the link mentioned in Appendix 7.6.2. Here, the DC, STV and STTV methods fail to recover the Zernike coefficients (*i.e.,* the pupil function) accurately and yield poor reconstructions of the dynamic sample. On the contrary, our method provides a good

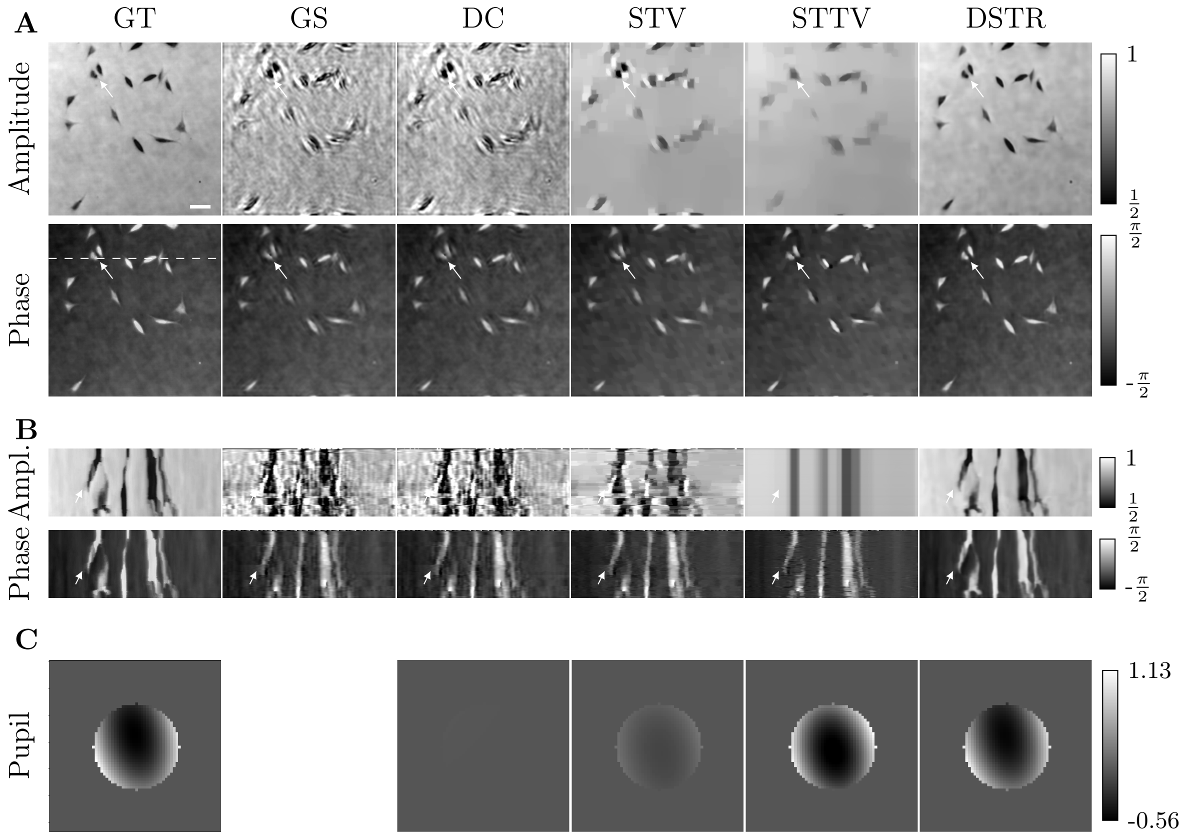


Figure 7.6: Joint recovery of the dynamic sample and the pupil function from noiseless measurements. Panel A: XY view for the frame index $q = 26$. Panel B: XT view for the Y position indicated in Panel A (GT, Phase, dashed line). Panel C: phase of the pupil function. Scale bar (for Panels A and B): 10 µm.

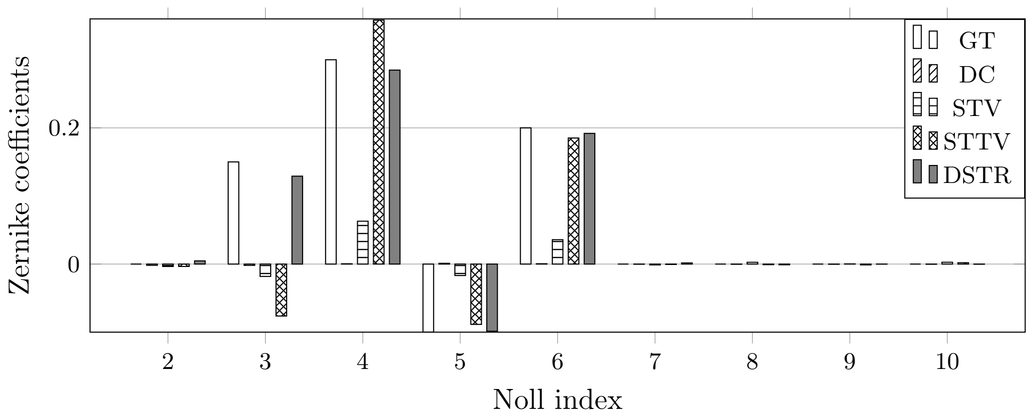


Figure 7.7: Recovered Zernike coefficients from noiseless measurements. The first (Noll index = 1) Zernike mode only contributes a constant phase factor which has no effect on the intensity measurements and thus can be ignored.

Table 7.4: Joint recovery of the dynamic sample and the pupil function from noisy measurements.

Method	GS	DC	STV	STTV	DSTR
Sequence RSNR [dB]	14.09	14.14	14.65	16.39	**24.86**
Pupil SNR [dB]	N.A.	9.36	10.66	14.98	**28.36**

estimate of the pupil function along with a high-quality reconstruction of the moving sample.

7.4.6 Reconstruction from Noisy Measurements

Finally, we consider the joint recovery of the sequence of images and the pupil function from noisy measurements. In this case, we observe that the GS, DC and DSTR methods require early stopping as running the corresponding iterative algorithm beyond a point leads to overfitting the noisy measurements. Thus, we run each method for a sufficiently large number of epochs (= 10,000) and we report the reconstruction that achieves the best RSNR during these epochs. For each method, we use the initialization, optimizer, learning rate and batch size described in Section 7.4.3. The hyperparameters for the STV and STTV methods are also tuned in the same way as in Section 7.4.3.

We summarize the quantitative results for all the methods in Table 7.4. We display some slices of the (2D + time) reconstructions and the estimated pupil functions (phase) in Figure 7.8, and we present the recovered Zernike coefficients in Figure 7.9. The entire reconstructed sequences are available at the link provided in Appendix 7.6.2. In this setting, as shown in Figure 7.4, the dark-field measurements are corrupted by significant amounts of noise, which makes the recovery problem quite challenging. Remarkably, our method still yields reconstructions of very good quality and outperforms the DC, STV and STTV methods by a big margin.

7.4.7 Computational Cost

In all our experiments, we used an Intel Xeon Gold 6240R (2.6 GHz) CPU for the GS method and an NVIDIA V100 GPU for the DC, STV, SSTV and DSTR methods. While DSTR achieves substantially better reconstruction quality than the other methods, its computational cost is also higher. For example, the run time for DSTR was around 5.5 hours as opposed to $3-30$ minutes for the other approaches when jointly estimating the sequence and the pupil function from noiseless measurements.

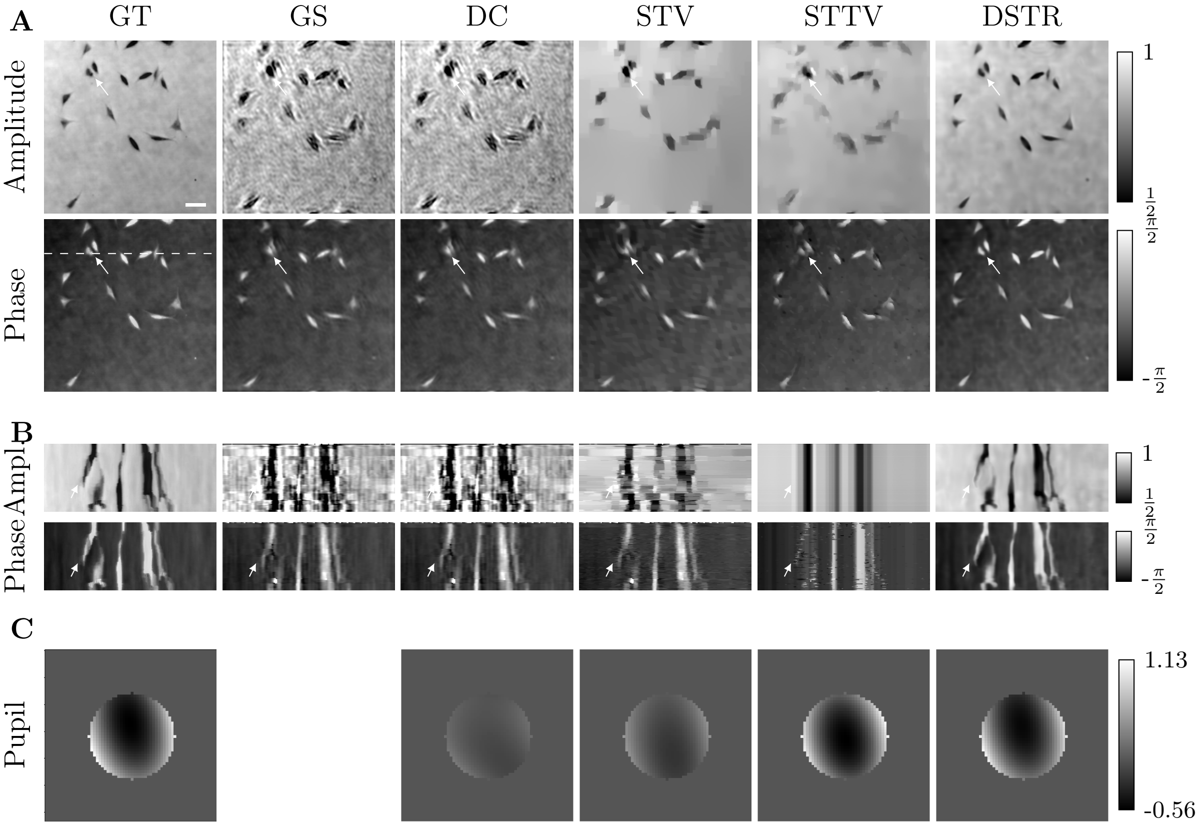


Figure 7.8: Joint recovery of the dynamic sample and the pupil function from noisy measurements. Panel A: XY view for the frame index $q = 26$. Panel B: XT view for the Y position indicated in Panel A (GT, Phase, dashed line). Panel C: phase of the pupil function. Scale bar (for Panels A and B): 10 µm.

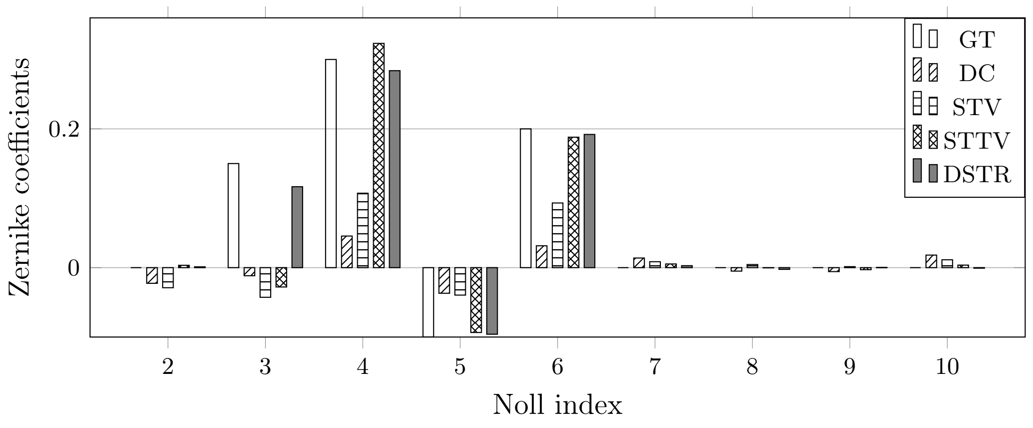


Figure 7.9: Recovered Zernike coefficients from noisy measurements. The first (Noll index = 1) Zernike mode only contributes a constant phase factor which has no effect on the intensity measurements and thus can be ignored.

7.5 Summary

We have presented a neural-network-based framework that does not require training data for the reconstruction of high-resolution complex-valued images of a moving sample in dynamic Fourier ptychography. In our method, we have parameterized the sequence of images to be reconstructed using a shared convolutional network with adjustable parameters. We have encoded the temporal behavior of the sample in the input vectors of the network by constraining them to lie on a one-dimensional manifold. In this manner, we have leveraged both the structural prior of a neural network and the temporal regularity between consecutive frames. Further, we have incorporated the recovery of the pupil function of the microscope within our framework. Finally, with the help of simulations, we have shown that the proposed approach drastically improves the quality of reconstruction over standard frame-by-frame methods.

7.6 Appendix

7.6.1 Zernike Polynomials

In the polar coordinates (ρ, ϕ), the Zernike polynomials are given by

$$Z_v^u(\rho,\phi) = \begin{cases} R_v^{|u|}(\rho)\cos(|u|\phi),\ u \geq 0 \\ R_v^{|u|}(\rho)\sin(|u|\phi),\ u < 0, \end{cases} \tag{7.34}$$

where $u \in \mathbb{Z}$, $v \in \mathbb{N}$, $\rho \in [0,1]$, $\phi \in [0, 2\pi)$, and

$$R_v^{|u|}(\rho) = \begin{cases} \sum_{s=0}^{\frac{(v-|u|)}{2}} \frac{(-1)^s\ (v-s)!}{s!\ \left(\frac{(v+|u|)}{2}-s\right)!\ \left(\frac{(v-|u|)}{2}-s\right)!}\ \rho^{v-2s},\ (v-|u|) \text{ is even,} \\ 0, \qquad (v-|u|) \text{ is odd.} \end{cases} \tag{7.35}$$

For $a \in \mathbb{Z}_+ \setminus \{0\}$, Noll's sequential indexing defines a mapping $Z_v^u \mapsto Z_a$ such that

$$a = \frac{v(v+1)}{2} + |u| + \begin{cases} 0,\ u > 0\ \wedge\ \lfloor v/2 \rfloor \in 2\mathbb{N} \\ 0,\ u < 0\ \wedge\ \lfloor v/2 \rfloor \in 2\mathbb{N}+1 \\ 0,\ u \geq 0\ \wedge\ \lfloor v/2 \rfloor \in 2\mathbb{N}+1 \\ 0,\ u \leq 0\ \wedge\ \lfloor v/2 \rfloor \in 2\mathbb{N}. \end{cases} \tag{7.36}$$

7.6.2 Data Link

The data corresponding to our experiments (the full ground-truth sequences and all the reconstructed sequences) can be found at https://iopscience.iop.org/article/10.1088/1361-6420/acca72/meta.

8 Conclusion

In this thesis, we have presented a collection of novel reconstruction methods for solving ill-posed inverse problems. These methods were developed within the penalized-likelihood-based estimation and Bayesian estimation paradigms and range from those that involve classical sparsity-based signal models to those that exploit the power of neural networks. In this concluding chapter, we first summarize our contributions and then briefly outline some directions for future research.

8.1 Contributions

Part I: The World of Sparsity

In the first part of the thesis, we have focused on sparse signal models in the context of linear inverse problems for 1D signals.

Continuous-Domain L_p-norm Regularization

We have devised an algorithm to numerically solve L_p-regularized generalized-interpolation problems for $p \geq 1$ and with a multi-order derivative regularization operator D^{N_0}. Our method involves the use of splines of degree N_0, with uniformly spaced knots, for an exact discretization. The resulting discrete problem is solved using the alternating direction method of multipliers (ADMM) and a small-enough grid size is picked with the help of a grid-refinement strategy. Through our experiments for spatial and Fourier interpolation, we have established the existence of a continuum of solutions as p goes from ∞ to 1. We have also made insightful observations about properties of L_p-regularized solutions such as sparsity, regularity, and Gibbs-like oscillations and overshoot.

Sparse Stochastic Processes

We have introduced a sparse-stochastic-processes-based framework to objectively benchmark the performance of reconstruction algorithms in the context of 1D linear inverse problems. In particular, our framework facilitates the benchmarking of neural-network-based methods that require large amounts of training data. We have developed customized Gibbs sampling schemes to compute the minimum-mean-square-error (MMSE) estimators for specific classes of sparse processes. These provide an upper bound on reconstruction performance and thus allow us to specify a quantitative measure of the statistical optimality of any given method. We have highlighted the abilities of our framework by benchmarking some iterative sparsity-promoting techniques (such as the total-variation-regularized method) and convolutional neural network (CNN) architectures that perform direct nonlinear reconstructions for deconvolution and Fourier-sampling problems. There, we have observed that, while CNNs outperform these sparsity-based approaches and achieve a near-optimal performance in terms of the mean-square error for a wide range of conditions, they can sometimes fail too, especially for signals with heavy-tailed innovations.

Part II: The Neural Network Revolution

In the second part of the thesis, we have looked at the integration of neural networks into the penalized-likelihood-based estimation and Bayesian estimation paradigms for image reconstruction.

Convergent Iterative Image-Reconstruction Methods

We have developed an efficient module for learning pointwise continuous piecewise-linear activation functions in neural networks. We have shown how our module can be adapted to train powerful 1-Lipschitz denoising neural networks and learnable convex regularizers, which can then be deployed within provably convergent iterative image-recontruction methods.

1. Learning Activation Functions in Neural Networks
We have presented an efficient computational solution based on B-splines to train neural networks with adjustable linear spline activation functions. Through several experiments for classification and signal-recovery, we have demonstrated that our method compares favorably to the widely-used ReLU networks, the improvement being more pronounced for simpler/smaller networks.

2. Lipschitz-Constrained Neural Networks for Plug-and-Play Reconstruction
We have proposed a framework to efficiently train Lipschitz-constrained neural networks with learnable linear-spline activation functions. Empirically, we have shown that our approach outperforms other Lipschitz-constrained neural architectures for a variety of

tasks including plug-and-play image reconstruction.

3. A Neural-Network-Based Convex Regularizer
We have proposed a framework to learn universal convex-ridge regularizers with adaptive profiles. When applied to inverse problems, it is competitive with recent deep-learning-based approaches that also prioritize reliability of the method. Our models are not only faster to train, but they also offer improvements in reconstruction quality.

Deep Generative Priors for Nonlinear Inverse Problems

We have developed a Bayesian reconstruction pipeline for the resolution of nonlinear inverse problems that leverages the power of deep generative models (such as variational autoencoders or generative adversarial networks) as image priors. Specifically, we have designed a tractable posterior-sampling scheme based on the Metropolis-adjusted Langevin algorithm (MALA) for the class of nonlinear inverse problems where the forward model has a neural-network-like computational structure. This class includes a wide variety of practical imaging modalities. We have also proposed the concept of augmented generative models to tackle the problem of the quantitative recovery of images. We have illustrated the advantages of our framework by applying it to phase retrieval and optical diffraction tomography.

Deep Spatiotemporal Regularization for Dynamic Fourier Ptychography

We have presented a neural-network-based framework that does not require training data for the reconstruction of high-resolution complex-valued images of a moving sample in dynamic Fourier ptychography (FP). In our method, we parametrize the sequence of images to be reconstructed as the outputs of a shared untrained deep convolutional network driven by a series of fixed input vectors that lie on a one-dimensional (temporal) manifold. The parameters of the network are then optimized to globally fit the acquired measurements according to a suitable likelihood-based criterion. The architecture of the network and the constraints on the input vectors impose spatiotemporal regularization on the sequence of images. We have also incorporated the estimation of the pupil function of the microscope within our framework. With the help of simulations, we have shown that our approach yields state-of-the-art reconstructions.

8.2 Outlook

In Chapter 3, we designed an exact discretization scheme for the L_p-regularized generalized-interpolation problem in 1D. Similar schemes have also been proposed to discretize MPL estimators (formulated directly in the continuous domain) for 1D linear inverse problems corresponding to the gTV regularization [104, 115]. The advantage of these approaches is that they incur no discretization error and are better matched to the underlying analog signal. A potential direction of research is the development of such exact discretization methods in higher dimensions for sparsity-based models such as the total-variation [25] and Hessian total-variation [335, 336] regularizers.

In Chapter 4, we presented a framework based on 1D first-order sparse stochastic processes to benchmark reconstruction algorithms. One can extend this framework to include hybrid processes (constructed by the superposition of elementary processes), which are suitable for modelling multicomponent signals. Another interesting (and more challenging) avenue for research is to extend our framework to non-separable multidimensional signal models. One possible way to do this would be to define a new class of multidimensional stochastic processes using the spline-operator-based framework of [135, 136].

In Chapter 5, we developed a module for learning component-wise activation functions in neural networks. An interesting direction of future work is the design of an efficient module for learning multivariate activation functions. For example, one could attempt to design nonlinearities $\sigma : \mathbb{R}^2 \to \mathbb{R}^2$ using the box spline representation from [337], with a possible application being the development of powerful complex-valued neural networks.

While our experiments in Chapter 6 demonstrated the potential of our deep-generative-prior-based Bayesian reconstruction framework, in the present form, our scheme lacks theoretical guarantees for MALA to be geometrically ergodic (convergence to the equilibrium distribution at a geometric rate). A topic of future work could be to investigate the imposition of appropriate constraints on the generative model such that the resulting posterior distribution satisfies certain smoothness and tail conditions [290] that ensure geometric ergodicity of MALA. Also, the performance of our scheme heavily relies on how well the prior models the object of interest. Thus, any progress on the side of designing and training high-quality large-scale deep generative models could be translated to our framework. Finally, while the neural-network-like structure of our forward models make our approach tractable, like MCMC methods in general, it requires a lot of computation. It could be interesting to consider alternatives to MALA that might help in speeding up this approach.

In Chapter 7, we presented an untrained-neural-network-based spatiotemporal regularization method for high-quality dynamic FP reconstruction. We deployed our method with early stopping for simulations involving noisy measurements. Alternately, one can adapt the idea in [84] and perform Bayesian estimation for the network parameters with a simple Gaussian prior distribution, which would then eliminate the need for early stopping. Moreover, this approach would also enable us to obtain a pixelwise variance map for each image in the sequence in addition its point estimate.

Bibliography

[1] V. G. Romanov, *Inverse Problems of Mathematical Physics.* Brill, 1987.

[2] C. W. Groetsch and C. Groetsch, *Inverse Problems in the Mathematical Sciences.* Springer, 1993, vol. 52.

[3] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems.* Springer Science & Business Media, 1996, vol. 375.

[4] M. Bertero, P. Boccacci, and C. De Mol, *Introduction to Inverse Problems in Imaging.* CRC press, 2021.

[5] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging.* SIAM, 2001.

[6] B. Huang, M. Bates, and X. Zhuang, "Super-resolution fluorescence microscopy," *Annual Review of Biochemistry*, vol. 78, pp. 993–1016, 2009.

[7] X.-C. Bai, G. McMullan, and S. H. Scheres, "How cryo-em is revolutionizing structural biology," *Trends in Biochemical Sciences*, vol. 40, no. 1, pp. 49–57, 2015.

[8] Z.-P. Liang and P. C. Lauterbur, *Principles of Magnetic Resonance Imaging.* SPIE Optical Engineering Press Belllingham, WA, 2000.

[9] K. Akiyama *et al.*, "First M87 event horizon telescope results. IV. Imaging the central supermassive black hole," *The Astrophysical Journal Letters*, vol. 875, no. 1, p. L4, 2019.

[10] H. Kudo, T. Suzuki, and E. A. Rashed, "Image reconstruction for sparse-view CT and interior CT—introduction to compressed sensing and differentiated back-projection," *Quantitative Imaging in Medicine and Surgery*, vol. 3, no. 3, p. 147, 2013.

[11] R. Gribonval, "Should penalized least squares regression be interpreted as maximum a posteriori estimation?" *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2405–2410, 2011.

[12] R. Gribonval, V. Cevher, and M. E. Davies, "Compressible distributions for high-dimensional statistics," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5016–5034, 2012.

[13] A. N. Tikhonov, "Solution of incorrectly formulated problems and the regularization method," *Soviet Mathematics*, vol. 4, pp. 1035–1038, 1963.

[14] M. Bertero and P. Boccacci, *Introduction to Inverse Problems in Imaging*. CRC press, 1998.

[15] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, Inc., 1993.

[16] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems*, 1996, pp. 514–520.

[17] S. Mallat, *A Wavelet Tour of Signal Processing*. Elsevier, 1999.

[18] M. Lustig, D. L. Donoho, and J. M. Pauly, "Sparse MRI: the application of compressed sensing for rapid MR imaging," vol. 58, no. 6, pp. 1182–1195, 2007.

[19] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

[20] M. Elad, *Sparse and Redundant Representations. From Theory to Applications in Signal and Image Processing*. Springer, 2010.

[21] 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: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.

[22] M. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, 2007.

[23] 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.

[24] S. Boyd, N. Parikh, and E. Chu, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Publishers Inc, 2011.

[25] 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.

[26] 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.

[27] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 3869–3872.

[28] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *Journal of Fourier analysis and applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[29] D. Wipf and S. Nagarajan, "Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 317–329, 2010.

[30] N. Dobigeon, A. O. Hero, and J.-Y. Tourneret, "Hierarchical Bayesian sparse image reconstruction with application to MRFM," *IEEE transactions on Image Processing*, vol. 18, no. 9, pp. 2059–2070, 2009.

[31] F. Krzakala, M. Mézard, F. Sausset, Y. F. Sun, and L. Zdeborová, "Statistical-physics-based reconstruction in compressed sensing," *Phys. Rev. X*, vol. 2, p. 021 005, 2 May 2012.

[32] L. Chaari, J.-Y. Tourneret, and H. Batatia, "Sparse Bayesian regularization using Bernoulli-Laplacian priors," in *21st European Signal Processing Conference (EUSIPCO 2013)*, 2013, pp. 1–5.

[33] M. Amrouche, H. Carfantan, and J. Idier, "Efficient sampling of Bernoulli-Gaussian-mixtures for sparse signal restoration," *IEEE Transactions on Signal Processing*, vol. 70, pp. 5578–5591, 2022.

[34] U. S. Kamilov, P. Pad, A. Amini, and M. Unser, "MMSE estimation of sparse Lévy processes," *IEEE Transactions on Signal Processing*, vol. 61, no. 1, pp. 137–147, 2012.

[35] A. Mohammad-Djafari and M. Dumitru, "Bayesian sparse solutions to linear inverse problems with non-stationary noise with student-t priors," *Digital Signal Processing*, vol. 47, pp. 128–156, 2015.

[36] F. Uribe, Y. Dong, and P. C. Hansen, "Horseshoe priors for edge-preserving linear Bayesian inversion," *SIAM Journal on Scientific Computing*, vol. 45, no. 3, B337–B365, 2023.

[37] M. T. McCann, K. H. Jin, and M. Unser, "Convolutional neural networks for inverse problems in imaging—a review," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 85–95, Nov. 2017.

[38] G. Ongie, A. Jalal, C. A. Metzler, 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.

[39] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[40] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT press Cambridge, 2016, vol. 1.

[41] 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.

[42] H. Chen *et al.*, “Low-dose CT with a residual encoder-decoder convolutional neural network,” *IEEE Transactions on Medical Imaging*, vol. 36, no. 12, pp. 2524–2535, 2017.

[43] 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.

[44] K. Gregor and Y. LeCun, “Learning fast approximations of sparse coding,” in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 399–406.

[45] 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.

[46] Y. Yang, J. Sun, H. Li, and Z. Xu, “Deep ADMM-Net for compressive sensing MRI,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[47] 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, 2018.

[48] J. Adler and O. Öktem, “Learned primal-dual reconstruction,” *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1322–1332, 2018.

[49] 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.

[50] J. Adler and O. Öktem, “Deep Bayesian inversion,” *arXiv preprint arXiv:1811.05910*, 2018.

[51] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, “Plug-and-play priors for model based reconstruction,” in *2013 IEEE Global Conference on Signal and Information Processing*, 2013, pp. 945–948.

[52] 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.

[53] 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, 2018.

[54] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces* (CMS Books in Mathematics), 2nd. Springer, Cham, 2017.

[55] J. Hertrich, S. Neumayer, and G. Steidl, “Convolutional proximal neural networks and plug-and-play algorithms,” *Linear Algebra and Its Applications*, vol. 631, pp. 203–234, 2021.

[56] T. Tirer and R. Giryes, “Image restoration by iterative denoising and backward projections,” *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1220–1234, 2018.

[57] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, “Plug-and-play methods provably converge with properly trained denoisers,” in *International Conference on Machine Learning*, 2019, pp. 5546–5557.

[58] Y. Sun, J. Liu, and U. Kamilov, “Block coordinate regularization by denoising,” in *Advances in Neural Information Processing Systems*, 2019, pp. 382–392.

[59] Z. Wu, Y. Sun, A. Matlock, J. Liu, L. Tian, and U. S. Kamilov, “SIMBA: Scalable inversion in optical tomography using deep denoising priors,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 6, pp. 1163–1175, 2020.

[60] J. Liu, Y. Sun, C. Eldeniz, W. Gan, H. An, and U. S. Kamilov, “RARE: Image reconstruction using deep priors learned without groundtruth,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 6, pp. 1088–1099, 2020.

[61] 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*, 2021.

[62] 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.

[63] M. Terris, A. Repetti, J.-C. Pesquet, and Y. Wiaux, “Building firmly nonexpansive convolutional neural networks,” in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8658–8662.

[64] 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.

[65] S. Hurault, A. Leclaire, and N. Papadakis, “Gradient step denoiser for convergent Plug-and-Play,” in *International Conference on Learning Representations*, 2022.

[66] 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, 17–23 July 2022, pp. 9483–9505.

[67] 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.

[68] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[69] I. Goodfellow *et al.*, “Generative adversarial nets,” *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[70] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, “Compressed sensing using generative models,” in *International Conference on Machine Learning*, PMLR, 2017, pp. 537–546.

[71] P. Hand, O. Leong, and V. Voroninski, “Phase retrieval under a generative prior,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[72] W. Huang, P. Hand, R. Heckel, and V. Voroninski, “A provably convergent scheme for compressive sensing under random generative priors,” *Journal of Fourier Analysis and Applications*, vol. 27, no. 2, pp. 1–34, 2021.

[73] Z. Kadkhodaie and E. P. Simoncelli, “Solving linear inverse problems using the prior implicit in a denoiser,” in *NeurIPS 2020 Workshop on Deep Learning and Inverse Problems*, 2020.

[74] B. Kawar, G. Vaksman, and M. Elad, “SNIPS: solving noisy inverse problems stochastically,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 757–21 769, 2021.

[75] R. Laumont, V. D. Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra, “Bayesian imaging using plug & play priors: when Langevin meets Tweedie,” *SIAM Journal on Imaging Sciences*, vol. 15, no. 2, pp. 701–737, 2022.

[76] D. Patel and A. A. Oberai, “Bayesian inference with generative adversarial network priors,” *arXiv preprint arXiv:1907.09987*, 2019.

[77] K. C. Tezcan, N. Karani, C. F. Baumgartner, and E. Konukoglu, “Sampling possible reconstructions of undersampled acquisitions in MR imaging with a deep learned prior,” *IEEE Transactions on Medical Imaging*, 2022.

[78] M. Holden, M. Pereyra, and K. C. Zygalakis, “Bayesian imaging with data-driven priors encoded by neural networks,” *SIAM Journal on Imaging Sciences*, vol. 15, no. 2, pp. 892–924, 2022.

[79] A. Jalal, M. Arvinte, G. Daras, E. Price, A. G. Dimakis, and J. Tamir, “Robust compressed sensing MRI with deep generative priors,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 938–14 954, 2021.

[80] Y. Song, L. Shen, L. Xing, and S. Ermon, “Solving inverse problems in medical imaging with score-based generative models,” in *International Conference on Learning Representations*, 2022.

[81] M. Zach, F. Knoll, and T. Pock, “Stable deep MRI reconstruction using generative priors,” *IEEE Transactions on Medical Imaging*, 2023.

[82] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9446–9454.

[83] D. Van Veen, A. Jalal, M. Soltanolkotabi, E. Price, S. Vishwanath, and A. G. Dimakis, "Compressed sensing with deep image prior and learned regularization," *arXiv preprint arXiv:1806.06438*, 2018.

[84] Z. Cheng, M. Gadelha, S. Maji, and D. Sheldon, "A Bayesian perspective on the deep image prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5443–5451.

[85] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[86] N. Parikh, S. Boyd, *et al.*, "Proximal algorithms," *Foundations and trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[87] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, pp. 1168–1200, 2005.

[88] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," 1970.

[89] C. J. Geyer, "Practical Markov chain Monte Carlo," *Statistical Science*, pp. 473–483, 1992.

[90] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice.* CRC press, 1995.

[91] D. Gamerman and H. F. Lopes, *Markov chain Monte Carlo: Stochastic Simulation for Bayesian Inference.* CRC press, 2006.

[92] G. O. Roberts and J. S. Rosenthal, "General state space Markov chains and MCMC algorithms," 2004.

[93] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[94] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[95] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[96] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing.* Birkhäuser Basel, 2013, vol. 1.

[97] E. J. Candes, 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.

[98] P. del Aguila Pla, S. Neumayer, and M. Unser, “Stability of image-reconstruction algorithms,” *IEEE Transactions on Computational Imaging*, pp. 1–11, 2023.

[99] R. Heckel and P. Hand, “Deep decoder: concise image representations from untrained non-convolutional networks,” *arXiv preprint arXiv:1810.03982*, 2018.

[100] M. Unser, J. Fageot, and H. Gupta, “Representer theorems for sparsity-promoting ℓ_1-regularization,” *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5167–5180, Sep. 2016.

[101] M. Unser, J. Fageot, and J. P. Ward, “Splines are universal solutions of linear inverse problems with generalized TV regularization,” *SIAM Review*, vol. 59, no. 4, pp. 769–793, Dec. 2017.

[102] P. Bohra and M. Unser, “Continuous-domain signal reconstruction using L_p-norm regularization,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 4543–4554, 2020.

[103] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press, 2001.

[104] H. Gupta, J. Fageot, and M. Unser, “Continuous-domain solutions of linear inverse problems with Tikhonov *versus* generalized TV regularization,” *IEEE Transactions on Signal Processing*, vol. 66, no. 17, pp. 4670–4684, 2018.

[105] S. Fisher and J. Jerome, “Spline solutions to L_1 extremal problems in one and several variables,” *Journal of Approximation Theory*, vol. 13, no. 1, pp. 73–83, Jan. 1975.

[106] C. de Boor, “On “best” interpolation,” *Journal of Approximation Theory*, vol. 16, no. 1, pp. 28–42, 1976.

[107] P. Copley and L. Schumaker, “On pLg-splines,” *Journal of Approximation Theory*, vol. 23, no. 1, pp. 1–28, 1978.

[108] C. Micchelli, P. W. Smith, J. Swetits, and J. D. Ward, “Constrained L_p approximation,” *Constructive Approximation*, vol. 1, no. 1, pp. 93–102, 1985.

[109] A. Pinkus, “On smoothest interpolants,” *SIAM Journal on Mathematical Analysis*, vol. 19, no. 6, pp. 1431–1441, 1988.

[110] J. Favard, “Sur l’interpolation,” *Bulletin de la Société Mathématique de France*, vol. 67, pp. 102–113, 1939.

[111] S. Karlin, “Interpolation properties of generalized perfect splines and the solutions of certain extremal problems. I,” *Transactions of the American Mathematical Society*, vol. 206, pp. 25–66, 1975.

[112] J. E. Lavery, “Univariate cubic L_p splines and shape-preserving, multiscale interpolation by univariate cubic L_1 splines,” *Computer Aided Geometric Design*, vol. 17, no. 4, pp. 319–336, 2000.

[113] P. Auquiert, O. Gibaru, and E. Nyiri, "C^1 and C^2-continuous polynomial parametric L_p splines ($p \geq 1$)," *Computer Aided Geometric Design*, vol. 24, no. 7, pp. 373–394, 2007.

[114] M. Unser and P. Tafti, *An Introduction to Sparse Stochastic Processes*. Cambridge, United Kingdom: Cambridge University Press, 2014, 367 p.

[115] T. Debarre, J. Fageot, H. Gupta, and M. Unser, "B-Spline-based exact discretization of continuous-domain inverse problems with generalized TV regularization," *IEEE Transactions on Information Theory*, pp. 1–1, 2019.

[116] I. J. Schoenberg, "Contributions to the problem of approximation of equidistant data by analytic functions: Part A— On the problem of smoothing or graduation. A first class of analytic approximation formulae," *Quarterly of Applied Mathematics*, vol. 4, no. 1, pp. 45–99, 1946.

[117] I. J. Schoenberg, *Cardinal Spline Interpolation*. SIAM, 1973, vol. 12.

[118] M. Unser and T. Blu, "Cardinal exponential splines: Part I—Theory and filtering algorithms," *IEEE Transactions on Signal Processing*, vol. 53, no. 4, pp. 1425–1438, Apr. 2005.

[119] J. J. Lei, "L_p-approximation by certain projection operators," *Journal of Mathematical Analysis and Applications*, vol. 185, no. 1, pp. 1–14, 1994.

[120] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer, 2011, pp. 185–212.

[121] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l1-ball for learning in high dimensions," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08, Helsinki, Finland: ACM, 2008, pp. 272–279.

[122] G. B. Dantzig, A. Orden, and P. Wolfe, "The generalized simplex method for minimizing a linear form under linear inequality restraints," *Pacific Journal of Mathematics*, vol. 5, no. 2, pp. 183–195, 1955.

[123] E. Soubies *et al.*, "Pocket guide to solve inverse problems with GlobalBioIm," *Inverse Problems*, vol. 35, no. 10, p. 104 006, 2019.

[124] F. Richards, "A Gibbs phenomenon for spline functions," *Journal of Approximation Theory*, vol. 66, no. 3, pp. 334–351, 1991.

[125] Z. Zhang and C. F. Martin, "Convergence and Gibbs' phenomenon in cubic spline interpolation of discontinuous functions," *Journal of Computational and Applied Mathematics*, vol. 87, no. 2, pp. 359–371, 1997.

[126] 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, Aug. 1992.

[127] R. J. Tibshirani, "The lasso problem and uniqueness," *Electronic Journal of statistics*, vol. 7, pp. 1456–1490, 2013.

[128] P. Bohra, P. del Aguila Pla, J.-F. Giovannelli, and M. Unser, "A statistical framework to investigate the optimality of signal-reconstruction methods," *IEEE Transactions on Signal Processing*, 2023.

[129] A. Amini, U. S. Kamilov, E. Bostan, and M. Unser, "Bayesian estimation for continuous-time sparse stochastic processes," *IEEE Transactions on Signal Processing*, vol. 61, no. 4, pp. 907–920, 2012.

[130] M. Unser, "Sampling—50 Years after Shannon," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 569–587, Apr. 2000.

[131] S. Ken-Iti, *Lévy Processes and Infinitely Divisible Distributions*. Cambridge University Press, 1999.

[132] A. Amini and M. Unser, "Sparsity and infinite divisibility," *IEEE Transactions on Information Theory*, vol. 60, no. 4, pp. 2346–2358, Apr. 2014.

[133] A. Amini, M. Unser, and F. Marvasti, "Compressibility of deterministic and random infinite sequences," *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5193–5201, Nov. 2011.

[134] J. R. Fageot, "Gaussian versus sparse stochastic processes: construction, regularity, compressibility," Ph.D. dissertation, Ecole Polytechnique Fédérale de Lausanne, 2017.

[135] R. Parhi and R. D. Nowak, "Banach space representer theorems for neural networks and ridge splines.," *J. Mach. Learn. Res.*, vol. 22, no. 43, pp. 1–40, 2021.

[136] M. Unser, "Ridges, neural networks, and the Radon transform," *arXiv preprint arXiv:2203.02543*, 2022.

[137] M. Unser and M. T. McCann, "Biomedical image reconstruction: from the foundations to deep neural networks," *Foundations and Trends in Signal Processing*, vol. 13, pp. 280–359, 2019.

[138] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 721–741, 1984.

[139] G. Casella and E. I. George, "Explaining the Gibbs sampler," *The American Statistician*, vol. 46, no. 3, pp. 167–174, 1992.

[140] M. A. Tanner and W. H. Wong, "The calculation of posterior distributions by data augmentation," *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 528–540, 1987.

[141] A. Mira and L. Tierney, "On the use of auxiliary variables in Markov chain Monte Carlo sampling," *Scandinavian Journal of Statistics*, 1997.

[142] T. Park and G. Casella, "The Bayesian LASSO," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 681–686, 2008.

[143] D. F. Andrews and C. L. Mallows, "Scale mixtures of normal distributions," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 36, no. 1, pp. 99–102, 1974.

[144] H. Rue, "Fast sampling of Gaussian Markov random fields," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 325–338, 2001.

[145] F. Orieux, O. Féron, and J.-F. Giovannelli, "Sampling high-dimensional Gaussian distributions for general linear inverse problems," *IEEE Signal Processing Letters*, vol. 19, no. 5, pp. 251–254, 2012.

[146] C. Gilavert, S. Moussaoui, and J. Idier, "Efficient Gaussian sampling for solving large-scale inverse problems using MCMC," *IEEE Transactions on Signal Processing*, vol. 63, no. 1, pp. 70–80, 2014.

[147] M. Vono, N. Dobigeon, and P. Chainais, "High-dimensional Gaussian sampling: A review and a unifying approach based on a stochastic proximal point algorithm," *SIAM Review*, vol. 64, no. 1, pp. 3–56, 2022.

[148] L. Devroye, "Random variate generation for the generalized inverse Gaussian distribution," *Statistics and Computing*, vol. 24, no. 2, pp. 239–246, 2014.

[149] C. Fevotte and S. J. Godsill, "A Bayesian approach for blind separation of sparse sources," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 6, pp. 2174–2188, 2006.

[150] D. Ge, J. Idier, and E. Le Carpentier, "Enhanced sampling schemes for MCMC based blind Bernoulli–Gaussian deconvolution," *Signal Processing*, vol. 91, no. 4, pp. 759–772, 2011.

[151] D. A. Van Dyk and T. Park, "Partially collapsed Gibbs samplers: theory and methods," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 790–796, 2008.

[152] K. Monakhova, J. Yurtsever, G. Kuo, N. Antipa, K. Yanny, and L. Waller, "Learned reconstructions for practical mask-based lensless imaging," *Optics Express*, vol. 27, no. 20, pp. 28 075–28 090, 2019.

[153] D. Perdios, M. Vonlanthen, F. Martinez, M. Arditi, and J.-P. Thiran, "CNN-based image reconstruction method for ultrafast ultrasound imaging," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 69, no. 4, pp. 1154–1168, 2021.

[154] 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.

[155] D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[156] 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.

[157] 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.

[158] 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.

[159] 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*, vol. 9, pp. 781–795, 2023.

[160] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[161] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[162] G. Hinton *et al.*, “Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[163] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: convolutional networks for biomedical image segmentation,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, (Cham), vol. 9351, Springer, LNCS, 2015, pp. 234–241.

[164] M. Unser, “A representer theorem for deep neural networks,” *Journal of Machine Learning Research*, vol. 20, no. 110, pp. 1–30, 2019.

[165] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, “On the number of linear regions of deep neural networks,” in *Advances in Neural Information Processing Systems*, 2014, pp. 2924–2932.

[166] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, “Understanding deep neural networks with rectified linear units,” *arXiv preprint arXiv:1611.01491*, 2016.

[167] G. Strang, “The functions of deep learning,” *SIAM News*, vol. 51, no. 10, pp. 1, 4, 2018.

[168] T. Poggio, L. Rosasco, A. Shashua, N. Cohen, and F. Anselmi, “Notes on hierarchical splines, DCLNs and i-theory,” Center for Brains, Minds and Machines (CBMM), Tech. Rep., 2015.

[169] R. Balestriero and R. G. Baraniuk, “A spline theory of deep learning,” in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 374–383.

[170] R. Parhi and R. D. Nowak, “Minimum “norm" neural networks are splines,” *arXiv preprint arXiv:1910.02333*, 2019.

[171] R. Parhi and R. D. Nowak, “Neural networks, ridge splines, and TV regularization in the Radon domain,” *arXiv preprint arXiv:2006.05626*, 2020.

[172] C. T. Chen and W. D. Chang, “A feedforward neural network with function shape autotuning,” *Neural Networks*, vol. 9, no. 4, pp. 627–641, 1996.

[173] S. H. Lane, M. Flax, D. Handelman, and J. Gelfand, “Multi-layer perceptrons with B-spline receptive field functions,” in *Advances in Neural Information Processing Systems*, 1991, pp. 684–692.

[174] L. Vecci, F. Piazza, and A. Uncini, “Learning and approximation capabilities of adaptive spline activation function neural networks,” *Neural Networks*, vol. 11, no. 2, pp. 259–270, 1998.

[175] S. Guarnieri, F. Piazza, and A. Uncini, “Multilayer feedforward networks with adaptive spline activation function,” *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 672–683, 1999.

[176] L. Hou, D. Samaras, T. M. Kurc, Y. Gao, and J. H. Saltz, “Convnets with smooth adaptive activation functions for regression,” *Proceedings of Machine Learning Research*, vol. 54, p. 430, 2017.

[177] R. G. Baraniuk, E. Candès, M. Elad, and M. Yi, “Applications of sparse representation and compressive sensing,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 906–909, 2010.

[178] V. Papyan, Y. Romano, J. Sulam, and M. Elad, “Theoretical foundations of deep learning via sparse representations: a multilayer sparse model and its connection to convolutional neural networks,” *IEEE Signal Processing Magazine*, vol. 35, no. 4, pp. 72–89, 2018.

[179] M. A. Figueiredo and R. D. Nowak, “An EM algorithm for wavelet-based image restoration,” *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 906–916, 2003.

[180] U. Kamilov and H. Mansour, “Learning optimal nonlinearities for iterative thresholding algorithms,” *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 747–751, 2016.

[181] 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, 2018.

[182] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[183] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning*, 2013, p. 3.

[184] 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.

[185] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," in *Proceedings of the International Conference on Learning Representations*, 2015.

[186] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep learning with s-shaped rectified linear activation units," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[187] D. Perekrestenko, P. Grohs, D. Elbrächter, and H. Bölcskei, "The universal approximation power of finite-width deep relu networks," *arXiv preprint arXiv:1806.01528*, 2018.

[188] H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen, "Optimal approximation with sparsely connected deep neural networks," *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 1, pp. 8–45, 2019.

[189] R. Gribonval, G. Kutyniok, M. Nielsen, and F. Voigtlaender, "Approximation spaces of deep neural networks," *arXiv preprint arXiv:1905.01208*, 2019.

[190] J. M. Tarela and M. V. Martinez, "Region configurations for realizability of lattice piecewise-linear models," *Mathematical and Computer Modelling*, vol. 30, no. 11, pp. 17–27, 1999.

[191] S. Wang and X. Sun, "Generalization of hinging hyperplanes," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4425–4431, 2005.

[192] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society. Series B*, vol. 58, no. 1, pp. 265–288, 1996.

[193] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[194] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2014.

[195] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[196] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[197] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[198] N. S. Keskar and R. Socher, “Improving generalization performance by switching from adam to sgd,” *arXiv preprint arXiv:1712.07628*, 2017.

[199] A. Molina, P. Schramowski, and K. Kersting, “Padé activation units: end-to-end learning of flexible activation functions in deep networks,” in *International Conference on Learning Representations*, 2019.

[200] 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, 2020.

[201] 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, pp. 84–98, 2016.

[202] A. Buades, B. Coll, and J.-M. Morel, “Non-local means denoising,” *Image Processing On Line*, vol. 1, pp. 208–212, 2011.

[203] S. Gu, L. Zhang, W. Zuo, and X. Feng, “Weighted nuclear norm minimization with application to image denoising,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2862–2869.

[204] 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.

[205] D. H. Ye, S. Srivastava, J.-B. Thibault, K. Sauer, and C. Bouman, “Deep residual learning for model-based iterative CT reconstruction using plug-and-play framework,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 6668–6672.

[206] 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, pp. 408–423, 2016.

[207] A. Teodoro, J. M. Bioucas-Dias, and M. Figueiredo, “Scene-adapted plug-and-play algorithm with convergence guarantees,” in *Proc. IEEE Int. Workshop on Machine Learning for Signal Processing*, IEEE, 2007.

[208] G. T. Buzzard, S. H. Chan, S. Sreehari, and C. A. Bouman, “Plug-and-play unplugged: optimization-free reconstruction using consensus equilibrium,” *SIAM Journal on Imaging Sciences*, vol. 11, no. 3, pp. 2001–2020, 2018.

[209] 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.

[210] R. G. Gavaskar and K. N. Chaudhury, “Plug-and-play ISTA converges with kernel denoisers,” *IEEE Signal Processing Letters*, vol. 27, pp. 610–614, 2020.

[211] 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.

[212] 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.

[213] 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.

[214] 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.

[215] 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.

[216] G. Nataraj and R. Otazo, “Model-free deep MRI reconstruction: A robustness study,” in *ISMRM Workshop on Data Sampling and Image*, 2020.

[217] 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.

[218] 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.

[219] S. Singla, S. Singla, and S. Feizi, “Householder activations for provable robustness against adversarial attacks,” *arXiv:2108.04062*, 2021.

[220] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *International Conference on Machine Learning*, 2017, pp. 214–223.

[221] C. Villani, *Optimal Transport: Old and New* (Grundlehren der Mathematischen Wissenschaften). Springer, Heidelberg, 2016.

[222] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “Infogan: interpretable representation learning by information maximizing generative adversarial nets,” *Advances in neural information processing systems*, vol. 29, 2016.

[223] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of Wasserstein GANs,” in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017, pp. 2644–2655.

[224] A. Saxe, J. McClelland, and S. Ganguli, “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks,” in *International Conference on Learning Representations*, 2014.

[225] 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.

[226] 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.

[227] 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.

[228] 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.

[229] 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.

[230] 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.

[231] S. Mukherjee, S. Dittmer, Z. Shumaylov, S. Lunz, O. Öktem, and C.-B. Schönlieb, “Learned convex regularizers for inverse problems,” *arXiv:2008.02839*, 2021.

[232] C. McCollough, “TU-FG-207A-04: overview of the low dose CT Grand Challenge,” *Medical Physics*, vol. 43, no. 6Part35, pp. 3759–3760, 2016.

[233] S. Roth and M. J. Black, “Fields of experts,” *International Journal of Computer Vision*, vol. 82, no. 2, pp. 205–229, 2009.

[234] 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.

[235] 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.

[236] S. Lunz, O. Öktem, and C.-B. Schönlieb, “Adversarial regularizers in inverse problems,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[237] M. Duff, N. D. F. Campbell, and M. J. Ehrhardt, “Regularising inverse problems with generative machine learning models,” *arXiv:2107.11191*, 2021.

[238] 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.

[239] R. Fermanian, M. L. Pendu, and C. Guillemot, "Learned gradient of a regularizer for plug-and-play gradient descent," *arXiv:2204.13940*, 2022.

[240] E. Kobler, T. Klatzer, K. Hammernik, and T. Pock, "Variational networks: connecting variational methods and deep learning," in *German conference on pattern recognition*, Springer, 2017, pp. 281–293.

[241] G. Peyré and J. M. Fadili, "Learning analysis sparsity priors," in *SampTA'11*, 2011, p. 4.

[242] 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.

[243] L. B. Willner, "On the distance between polytopes," *Quarterly of Applied Mathematics*, vol. 26, no. 2, pp. 207–212, 1968.

[244] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[245] 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.

[246] 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.

[247] 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.

[248] 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.

[249] M. Unser, "Splines: A perfect fit for signal and image processing," *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, Nov. 1999, IEEE-SPS best paper award.

[250] 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.

[251] 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, 13–18 Jul 2020, pp. 6702–6712.

[252] 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.

[253] 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.

[254] 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.

[255] 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.

[256] 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.

[257] J. R. Chand and M. Jacob, "Multi-scale energy (MuSE) plug and play framework for inverse problems," *arXiv:2305.04775*, 2023.

[258] 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, Jun. 2017, pp. 146–155.

[259] 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.

[260] P. Nair and K. N. Chaudhury, "On the construction of averaged deep denoisers for image regularization," *arXiv:2207.07321*, 2022.

[261] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical imaging and vision*, vol. 20, no. 1, pp. 89–97, 2004.

[262] 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.

[263] 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.

[264] P. Bohra, T.-a. Pham, J. Dong, and M. Unser, "Bayesian inversion for nonlinear imaging models using deep generative priors," *IEEE Transactions on Computational Imaging*, vol. 8, pp. 1237–1249, 2022.

[265] G. O. Roberts and R. L. Tweedie, "Exponential convergence of Langevin distributions and their discrete approximations," *Bernoulli*, pp. 341–363, 1996.

[266] G. O. Roberts and O. Stramer, "Langevin diffusions and Metropolis-Hastings algorithms," *Methodology and Computing in Applied Probability*, vol. 4, no. 4, pp. 337–357, 2002.

[267] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase retrieval with application to optical imaging: A contemporary overview," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 87–109, 2015.

[268] F. Fogel, I. Waldspurger, and A. d'Aspremont, "Phase retrieval for imaging problems," *Mathematical Programming Computation*, vol. 8, no. 3, pp. 311–335, 2016.

[269] R. P. Millane, "Phase retrieval in crystallography and optics," *JOSA A*, vol. 7, no. 3, pp. 394–411, 1990.

[270] A. M. Maiden and J. M. Rodenburg, "An improved ptychographical phase retrieval algorithm for diffractive imaging," *Ultramicroscopy*, vol. 109, no. 10, pp. 1256–1262, 2009.

[271] J. R. Fienup, J. C. Marron, T. J. Schulz, and J. H. Seldin, "Hubble space telescope characterized by using phase-retrieval algorithms," *Applied Optics*, vol. 32, no. 10, pp. 1747–1767, 1993.

[272] W. L. Freedman *et al.*, "Final results from the Hubble space telescope key project to measure the Hubble constant," *The Astrophysical Journal*, vol. 553, no. 1, p. 47, 2001.

[273] M. H. Maleki and A. J. Devaney, "Phase-retrieval and intensity-only reconstruction algorithms for optical diffraction tomography," *JOSA A*, vol. 10, no. 5, pp. 1086–1092, 1993.

[274] T. E. Gureyev and K. A. Nugent, "Rapid quantitative phase imaging using the transport of intensity equation," *Optics Communications*, vol. 133, no. 1-6, pp. 339–346, 1997.

[275] F. Zernike, "Phase contrast, a new method for the microscopic observation of transparent objects—Part II," *Physica*, vol. 9, no. 10, pp. 974–986, 1942.

[276] G. Zheng, R. Horstmeyer, and C. Yang, "Wide-field, high-resolution Fourier ptychographic microscopy," *Nature Photonics*, vol. 7, no. 9, pp. 739–745, 2013.

[277] J. M. Rodenburg and H. M. Faulkner, "A phase retrieval algorithm for shifting illumination," *Applied Physics Letters*, vol. 85, no. 20, pp. 4795–4797, 2004.

[278] E. J. Candès, X. Li, and M. Soltanolkotabi, "Phase retrieval via Wirtinger flow: theory and algorithms," *IEEE Transactions on Information Theory*, vol. 61, no. 4, pp. 1985–2007, 2015.

[279] M. Mondelli and A. Montanari, "Fundamental limits of weak recovery with applications to phase retrieval," in *Conference on Learning Theory*, 2018, pp. 1445–1450.

[280] E. Wolf, "Three-dimensional structure determination of semi-transparent objects from holographic data," *Optics Communications*, vol. 1, no. 4, pp. 153–156, 1969.

[281] A. Devaney, "Inverse-scattering theory within the Rytov approximation," *Optics Letters*, vol. 6, no. 8, pp. 374–376, 1981.

[282] E. Soubies, T.-A. Pham, and M. Unser, "Efficient inversion of multiple-scattering model for optical diffraction tomography," *Optics Express*, vol. 25, no. 18, pp. 21 786–21 800, 2017.

[283] J. Latz, "On the well-posedness of Bayesian inverse problems," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 8, no. 1, pp. 451–482, 2020.

[284] A. Gelman, W. R. Gilks, and G. O. Roberts, "Weak convergence and optimal scaling of random walk metropolis algorithms," *The annals of applied probability*, vol. 7, no. 1, pp. 110–120, 1997.

[285] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[286] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[287] J. Lim, A. Goy, M. H. Shoreh, M. Unser, and D. Psaltis, "Learning tomography assessed using Mie theory," *Phys. Rev. Applied*, vol. 9, p. 034 027, 3 Mar. 2018.

[288] U. S. Kamilov *et al.*, "Learning approach to optical tomography," *Optica*, vol. 2, no. 6, pp. 517–522, 2015.

[289] U. S. Kamilov *et al.*, "Optical tomographic image reconstruction based on beam propagation and sparse regularization," *IEEE Transactions on Computational Imaging*, vol. 2, no. 1, pp. 59–70, 2016.

[290] A. Durmus and É. Moulines, "On the geometric convergence for MALA under verifiable conditions," *arXiv preprint arXiv:2201.01951*, 2022.

[291] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in Neural Information Processing Systems*, vol. 29, pp. 2234–2242, 2016.

[292] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[293] C. Villani, *Optimal Transport: Old and New.* Springer, 2009, vol. 338.

[294] 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, 2017.

[295] P. Bohra, T.-a. Pham, Y. Long, J. Yoo, and M. Unser, "Dynamic Fourier ptychography with deep spatiotemporal priors," *Inverse Problems*, vol. 39, no. 6, p. 064 005, 2023.

[296] Y. Zhang, W. Jiang, L. Tian, L. Waller, and Q. Dai, "Self-learning based Fourier ptychographic microscopy," *Optics Express*, vol. 23, no. 14, pp. 18 471–18 486, 2015.

[297] Y. F. Cheng, M. Strachan, Z. Weiss, M. Deb, D. Carone, and V. Ganapati, "Illumination pattern design with deep learning for single-shot Fourier ptychographic microscopy," *Optics Express*, vol. 27, no. 2, pp. 644–656, 2019.

[298] L. Tian, X. Li, K. Ramchandran, and L. Waller, "Multiplexed coded illumination for Fourier ptychography with an LED array microscope," *Biomedical Optics Express*, vol. 5, no. 7, pp. 2376–2389, 2014.

[299] L. Tian, Z. Liu, L.-H. Yeh, M. Chen, J. Zhong, and L. Waller, "Computational illumination for high-speed in vitro Fourier ptychographic microscopy," *Optica*, vol. 2, no. 10, pp. 904–911, 2015.

[300] J. Sun, C. Zuo, J. Zhang, Y. Fan, and Q. Chen, "High-speed Fourier ptychographic microscopy based on programmable annular illuminations," *Scientific Reports*, vol. 8, no. 1, pp. 1–12, 2018.

[301] M. R. Kellman, E. Bostan, N. A. Repina, and L. Waller, "Physics-based learned design: Optimized coded-illumination for quantitative phase imaging," *IEEE Transactions on Computational Imaging*, vol. 5, no. 3, pp. 344–353, 2019.

[302] R. W. Gerchberg, "A practical algorithm for the determination of phase from image and diffraction plane pictures," *Optik*, vol. 35, pp. 237–246, 1972.

[303] P. Netrapalli, P. Jain, and S. Sanghavi, "Phase retrieval using alternating minimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 18, pp. 4814–4826, 2015.

[304] A. Chai, M. Moscoso, and G. Papanicolaou, "Array imaging using intensity-only measurements," *Inverse Problems*, vol. 27, no. 1, p. 015 005, 2010.

[305] E. J. Candès, T. Strohmer, and V. Voroninski, "Phaselift: exact and stable signal recovery from magnitude measurements via convex programming," *Communications on Pure and Applied Mathematics*, vol. 66, no. 8, pp. 1241–1274, 2013.

[306] I. Waldspurger, A. d'Aspremont, and S. Mallat, "Phase recovery, MaxCut and complex semidefinite programming," *Mathematical Programming*, vol. 149, no. 1, pp. 47–81, 2015.

[307] F. Soulez, É. Thiébaut, A. Schutz, A. Ferrari, F. Courbin, and M. Unser, "Proximity operators for phase retrieval," *Applied Optics*, vol. 55, no. 26, pp. 7412–7421, 2016.

[308] L. Bian *et al.*, "Fourier ptychographic reconstruction using Poisson maximum likelihood and truncated Wirtinger gradient," *Scientific Reports*, vol. 6, no. 1, pp. 1–10, 2016.

[309] Y. Huang, A. C. Chan, A. Pan, and C. Yang, "Memory-efficient, global phase-retrieval of Fourier ptychography with alternating direction method," in *Computational Optical Sensing and Imaging*, 2019, CTu4C–2.

[310] L.-H. Yeh *et al.*, "Experimental robustness of Fourier ptychography phase retrieval algorithms," *Optics Express*, vol. 23, no. 26, pp. 33 214–33 240, 2015.

[311] X. Ou, G. Zheng, and C. Yang, "Embedded pupil function recovery for Fourier ptychographic microscopy," *Optics Express*, vol. 22, no. 5, pp. 4960–4972, 2014.

[312] J. Sun, Q. Chen, Y. Zhang, and C. Zuo, "Efficient positional misalignment correction method for Fourier ptychographic microscopy," *Biomedical Optics Express*, vol. 7, no. 4, pp. 1336–1350, 2016.

[313] R. Eckert, Z. F. Phillips, and L. Waller, "Efficient illumination angle self-calibration in Fourier ptychography," *Applied Optics*, vol. 57, no. 19, pp. 5434–5442, 2018.

[314] G. Zheng, C. Shen, S. Jiang, P. Song, and C. Yang, "Concept, implementations and applications of Fourier ptychography," *Nature Reviews Physics*, vol. 3, no. 3, pp. 207–223, 2021.

[315] C. Kuang *et al.*, "Digital micromirror device-based laser-illumination Fourier ptychographic microscopy," *Optics Express*, vol. 23, no. 21, pp. 26 999–27 010, 2015.

[316] D. Ren, E. Bostan, L.-H. Yeh, and L. Waller, "Total-variation regularized Fourier ptychographic microscopy with multiplexed coded illumination," in *Imaging and Applied Optics 2017*, 2017, p. MM3C.5.

[317] Q. Shi *et al.*, "Under-sampling reconstruction with total variational optimization for Fourier ptychographic microscopy," *Optics Communications*, vol. 492, p. 126 986, 2021.

[318] Y. Zhang *et al.*, "Neural network model assisted Fourier ptychography with Zernike aberration recovery and total variation constraint," *Journal of Biomedical Optics*, vol. 26, no. 3, pp. 1–14, 2021.

[319] Y. Zhang *et al.*, "PgNN: physics-guided neural network for Fourier ptychographic microscopy," *arXiv preprint arXiv:1909.08869*, 2019.

[320] G. Jagatap, Z. Chen, C. Hegde, and N. Vaswani, "Sub-diffraction imaging using Fourier ptychography and structured sparsity," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6493–6497.

[321] Y. Sun, S. Xu, Y. Li, L. Tian, B. Wohlberg, and U. S. Kamilov, "Regularized Fourier ptychography using an online plug-and-play algorithm," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 7665–7669.

[322] M. T. McCann, K. H. Jin, and M. Unser, "Convolutional neural networks for inverse problems in imaging: a review," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 85–95, 2017.

[323] A. Kappeler, S. Ghosh, J. Holloway, O. Cossairt, and A. Katsaggelos, "Ptychnet: CNN based Fourier ptychography," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 1712–1716.

[324] N. Thanh, Y. Xue, Y. Li, L. Tian, and G. Nehmetallah, “Deep learning approach to Fourier ptychographic microscopy,” *Optics Express*, 2018.

[325] J. Zhang, T. Xu, Z. Shen, Y. Qiao, and Y. Zhang, “Fourier ptychographic microscopy reconstruction with multiscale deep residual network,” *Optics Express*, vol. 27, no. 6, pp. 8612–8625, 2019.

[326] F. Shamshad, F. Abbas, and A. Ahmed, “Deep ptych: subsampled Fourier ptychography using generative priors,” in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 7720–7724.

[327] F. Shamshad, A. Hanif, F. Abbas, M. Awais, and A. Ahmed, “Adaptive Ptych: Leveraging image adaptive generative priors for subsampled Fourier ptychography,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2019.

[328] P. C. Konda, L. Loetgering, K. C. Zhou, S. Xu, A. R. Harvey, and R. Horstmeyer, “Fourier ptychography: Current applications and future promises,” *Optics Express*, vol. 28, no. 7, pp. 9603–9630, 2020.

[329] Z. Chen, G. Jagatap, S. Nayer, C. Hegde, and N. Vaswani, “Low rank Fourier ptychography,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6538–6542.

[330] J. Yoo, K. H. Jin, H. Gupta, J. Yerly, M. Stuber, and M. Unser, “Time-dependent deep image prior for dynamic MRI,” *IEEE Transactions on Medical Imaging*, 2021.

[331] D. H. Rapoport, T. Becker, A. M Mamlouk, S. Schicktanz, and C. Kruse, “A novel validation algorithm allows for automated cell tracking and the extraction of biologically meaningful parameters,” *PLOS ONE*, vol. 6, no. 11, pp. 1–16, 2011.

[332] S. J. Reddi, S. Kale, and S. Kumar, “On the convergence of adam and beyond,” in *International Conference on Learning Representations*, 2018.

[333] M. S. Bartlett, “The square root transformation in analysis of variance,” *Supplement to the Journal of the Royal Statistical Society*, vol. 3, no. 1, pp. 68–78, 1936.

[334] F. J. Anscombe, “The transformation of Poisson, binomial and negative-binomial data,” *Biometrika*, vol. 35, no. 3/4, pp. 246–254, 1948.

[335] S. Lefkimmiatis, A. Bourquard, and M. Unser, “Hessian-based norm regularization for image restoration with biomedical applications,” *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 983–995, 2011.

[336] S. Lefkimmiatis, J. P. Ward, and M. Unser, “Hessian Schatten-norm regularization for linear inverse problems,” *IEEE transactions on image processing*, vol. 22, no. 5, pp. 1873–1888, 2013.

[337] 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, 2021.

Pakshal Bohra

Address: EPFL STI IMT LIB, BM 4141, Station 17, CH-1015 Lausanne
E-mail: pakshal.bohra@epfl.ch, pakshalbohra@gmail.com

Education

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland 2018 - Present
Ph.D in Electrical Engineering, Biomedical Imaging Group
Thesis: Statistical Inference for Inverse Problems: From Sparsity-Based Methods to Neural Networks
Advisor: Prof. Michael Unser

Indian Institute of Technology Bombay, Mumbai, India 2013 - 2018
Dual Degree (B.Tech + M.Tech) in Electrical Engineering
Master's Thesis: Poisson Inverse Problems with Performance Bounds
Advisors: Prof. Ajit Rajwade, Prof. Rajbabu Velmurugan

Research Interests

Computational Methods for Inverse Problems, Machine Learning, Applied Probability

Publications

Preprints

1. S. Ducotterd, A. Goujon, **P. Bohra**, D. Perdios, S. Neumayer, and M. Unser, "Improving Lipschitz-Constrained Neural Networks by Learning Activation Functions", *arXiv preprint arXiv:2210.16222*, 2022.

Journals (* denotes equal contribution)

1. 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*, vol. 9, pp. 781-795, 2023.
2. **P. Bohra***, T. -a. Pham*, Y. Long, J. Yoo, and M. Unser, "Dynamic Fourier Ptychography With Deep Spatiotemporal Priors", *Inverse Problems*, vol. 39, no. 6, paper no. 064005, June 2023.
3. **P. Bohra**, P. del Aguila Pla, J. -F. Giovannelli, and M. Unser, "A Statistical Framework To Investigate the Optimality of Signal-Reconstruction Methods", *IEEE Transactions on Signal Processing*, vol. 71, pp. 2043-2055, June 2023.
4. 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.
5. **P. Bohra**, T. -a. Pham, J. Dong, and M. Unser, "Bayesian Inversion for Nonlinear Imaging Models Using Deep Generative Priors", *IEEE Transactions on Computational Imaging*, vol. 8, pp. 1237-1249, 2022.
6. **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, November 2020.
7. **P. Bohra** and M. Unser, "Continuous-Domain Signal Reconstruction Using L_p-Norm Regularization", *IEEE Transactions on Signal Processing*, vol. 68, pp. 4543-4554, August 2020.
8. **P. Bohra***, D. Garg*, K. S. Gurumoorthy, A. Rajwade, "Variance Stabilization-Based Compressive Inversion Under Poisson or Poisson–Gaussian Noise With Analytical Bounds," *Inverse Problems*, vol. 35, no. 10, October 2019.

Conferences and Workshops

1. **P. Bohra**, D. Perdios, A. Goujon, S. Emery, 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*, December 2021.
2. **P. Bohra**, M. Unser, "Computation of "Best" Interpolants in the Lp Sense," *IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP)*, May 2020, pp. 5505-5509.
3. **P. Bohra**, A. Rajwade, "Poisson Low-Rank Matrix Recovery Using the Anscombe Transform, " *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, November 2018, pp. 141-145.

Talks

1. Title: Bayesian Inference for Inverse Problems: From Sparsity-Based Methods to Deep Neural Networks
   Tutorial, *IEEE International Symposium on Biomedical Imaging (ISBI)*, Cartagena de Indias, Republic of Colombia, April 2023.
2. Title: Bayesian Image Reconstruction: From Sparsity-Based Methods to Deep Neural Networks
   Minitutorial, *SIAM Conference on Imaging Science*, Virtual, March 2022.
3. Title: Opportunities and Challenges for Generative Adversarial Reconstruction by Distribution Matching (CryoGAN)
   SIAM Conference on Imaging Science, Virtual, March 2022.

Review Activity

IEEE Transactions on Computational Imaging, IEEE Transactions on Signal Processing, Elsevier Signal Processing, SIAM Journal on Imaging Sciences.

Teaching Experience

EPFL

1. Head Teaching Assistant for Image Processing I (Fall 2021)
2. Teaching Assistant for Image Processing I (Fall 2018-2020)
3. Teaching Assistant for Image Processing II (Spring 2019-2021)

IIT Bombay

1. Teaching Assistant for Advanced Topics in Signal Processing (Fall 2017)
2. Teaching Assistant for Digital Signal Processing (Spring 2018)

Supervised Students

Master's Thesis

1. Maxime Jotterand — Fall 2021
   Title: Learning 2D Activation Functions for Complex-Valued Neural Networks
2. Stanislas Ducotterd — Fall 2021
   Title: Lipschitz-Constrained Deep Spline Neural Networks
3. Sébastien Emery — Fall 2021
   Title: Deep Learning for Enhanced Ultrasound Image Reconstruction
4. Yuxuan Long — Fall 2021
   Title: Computational Methods for Dynamic Fourier Ptychography

Internship

1. Ayoub El Biari — Spring 2023
   Title: Characterization of Random Neural Networks As Stochastic Processes
2. Paul Margain — Spring 2021
   Title: High-Resolution Reconstruction in Single-Particle Cryo-EM With a Multiscale Joint Refinement Scheme
3. Tina Behnia — Summer 2020
   Title: Dictionary Learning With SαS Signal and Noise Models

Master Semester Project

1. Maxime Perret — Spring 2022
   Title: Posterior Integrals With MCMC for Bayesian Tests
2. Sébastien Emery — Spring 2021
   Title: Lipschitz-Constrained Neural Networks for Plug-and-Play Methods
3. Malo Simondin — Fall 2020
   Title: Benchmarking of Proximal Algorithms for Solving Regularized Inverse Problems
4. Louis-Nicolas Douce — Spring 2020
   Title: Slice-Based Dictionary Learning for Computed Tomography

5. Clélie De Witasse Fall 2019
   Title: Dictionary Learning for Limited Angle Computed Tomography
6. Zhiwei Huang Spring 2019
   Title: Benchmarking of Numerical Methods for Inverse Problems

Technical Skills

1. Programming Languages: C/C++, Python, Java
2. Others: MATLAB, GNU Radio, Android Studio, Unity 3D, LaTeX, ImageJ