Università
di Genova

Ph.D. in Mathematics and Applications

Manifold Learning and Sparsity Priors for Inverse Problems

Candidate:
Silvia Sciutto

Supervisors:
Prof. Matteo Santacesaria
Prof. Giovanni S. Alberti

UniGe
DIMA

XXXVI Cycle

To my boyfriend Cesare, nicknamed Ava,
who supported me with patience and love.

Contents

Abstract

In this thesis we investigate two distinct regularizing approaches for solving inverse problems. The first approach involves assuming that the unknown belongs to a manifold represented by a deep generative model. In particular, we use multiresolution analysis tools to design a generative neural network in the continuous setting, where the model's output belongs to an infinite-dimensional function space and we study the injectivity of this generator, as it is a necessary condition for its range to represent a manifold. Furthermore, we derive Lipschitz stability estimates for (possibly nonlinear) infinite-dimensional inverse problems with unknowns that belong to the manifold generated by our generative model. However, this generative model can represent only manifolds that admit a global parameterization. To address this limitation and represent manifolds of arbitrary topology, we propose to learn a mixture model of variational autoencoders, where each encoder-decoder pair represents one chart of a manifold. Moreover, we develop a Riemannian gradient descent algorithm on the learned manifold to solve inverse problems, with the constraint that the unknowns belong to the manifold.

The second approach introduces a probabilistic sparsity prior formulated as a mixture of degenerate Gaussians, capable of modeling sparsity with respect to a generic basis. Within this context, we develop a neural network, serving as the Bayesian estimator for linear inverse problems and we propose both a supervised and an unsupervised training strategies to estimate the parameters of this network.

We demonstrate the performance of both approaches for denoising and deblurring problem with 1D or 2D datasets. We also consider the electrical impedance tomography problem for the mixture model of variational autoencoders technique.

Chapter 1

Introduction

In Section 1.1 we provide a brief overview of inverse problems (IPs) and deep learning, and the use of classical and deep learning techniques for solving IPs. In Section 1.2 we present the thesis's contributions, emphasizing its innovative aspects.

1.1 Inverse Problems and Deep Learning preliminaries

In Section 1.1.1, we initially introduce IPs and discuss the key issues associated with them. Subsequently, we analyze several classical methods for their resolution, including Tikhonov regularization, LASSO and regularization by projection.

Moving on to Section 1.1.2, we provide an introduction to the fundamental concepts of deep learning (DL). Here, we outline the principal neural network (NN) architectures, with a specific focus on generative NNs, and we explain the general strategy for training NNs, highlighting the variational autoencoder (VAE) training method.

Finally, in Section 1.1.3, we conduct a review of the primary DL methods employed for solving IPs. These encompass fully learned techniques that exclusively rely on NNs to derive solutions for IPs, as well as methods that combine DL with classical techniques. Within the latter category, we distinguish four subcategories based on the manner in which NNs are employed: post-processing the solution, learning an iterative scheme, learning a regularization operator, or encoding a priori information about the unknowns.

1.1.1 Inverse Problems

Inverse problems [136] are extensively studied in mathematics due to their prevalence in real-world scenarios, arising when information about a system needs to be extracted from noisy observations. In these problems, we have access to noisy data generated by the system, denoted as y, and possess knowledge of the system's behavior through the forward operator, $\mathcal{F}\colon X \to Y$, which represents how the system functions. The main objective is to determine the unknown quantity of interest, denoted as x. Mathematically, this relationship can be expressed as:

$$y = \mathcal{F}(x) + \varepsilon, \tag{1.1}$$

where ε denotes the noise, often assumed to follow a Gaussian distribution.

Several examples of inverse problems can be found within the imaging domain [41], including denoising [48], deblurring [112], superresolution [86], and inpainting [40] and the medical domain, including computed tomography (CT) [49], magnetic resonance imaging (MRI) [157], positron emission tomography (PET) [28], and electrical impedance tomography (EIT) [62]. In simpler scenarios like denoising and deblurring, the forward operator takes a straightforward form. Specifically, for denoising, the forward operator is represented by the identity function, while for deblurring, it is a convolution with a blur filter. However, in the context of medical applications, the forward operator becomes notably more intricate as it incorporates the fundamental physical laws inherent to the specific problem. For instance, in CT the forward operator needs to incorporate integration over lines, while for EIT it necessitates a representation based on a partial differential equation (PDE) [84]. In the latter case, the nonlinearity of the forward map further adds to the complexity of the problem.

In any case, while solving the forward problem, namely finding y given x, is relatively straightforward, the inverse problem presents significant challenges. Indeed, in nearly all inverse problems, directly inverting the forward operator to reconstruct the unknown x amplifies the effect of the noise ε present in the initial measurement y. Consequently, the reconstruction obtained using this method may substantially deviate from the ground truth, namely the true unknown from which y is measured, and may exhibit artifacts. Even when assuming perfect measurements without any noise, the inverse problems often result "ill posed", term introduced by the mathematician Jacques Hadamard in the 20th century. According to his definition [103], a problem is well-posed if the solution exists, it is unique and it exhibits continuous dependence on the data, namely the solution is stable.

In order to ensure the existence of the solution, one can consider a relaxed version of equation (1.1) by seeking all possible values x for which $\mathcal{F}(x)$ closely approximates y in a suitable metric. This leads to the following minimization problem

$$\min_{x \in X} \mathcal{L}(\mathcal{F}(x), y), \tag{1.2}$$

where $\mathcal{L}(\mathcal{F}(x), y)$ represents the data discrepancy term which quantifies the distance between y and $\mathcal{F}(x)$. When Y is a Hilbert space, a common choice for $\mathcal{L}(\mathcal{F}(x), y)$ is $\|\mathcal{F}(x) - y\|_Y^2$. In this case, a solution of (1.2) is referred to as the Least Squares solution [90]. For the purposes of our analysis, we assume that both X and Y are Hilbert spaces.

1.1.1.1 Classical techniques for Inverse Problems

In order to address the issues of uniqueness and stability in inverse problems, various techniques have been developed over the years. These techniques aim to *regularize* the problems, namely to stabilize the solution, and encompass generalized inverse methods, iterative methods with early stopping, and variational methods. In this section, we provide a brief overview of the three most widely used variational methods: L^2 regularization

(Tikhonov regularization), L^1 regularization (total variation and least absolute shrinkage and selection operator, known as LASSO) and regularization by projection.

Variational methods introduce a penalty/regularization term, which we denote by $J(x)$, into the minimization problem (1.2). This results in the following formulation

$$\min_{x \in X} \mathcal{L}(\mathcal{F}(x), y) + \lambda J(x), \tag{1.3}$$

where $\lambda > 0$ represents a regularization parameter. The penalty term $J(x)$ promotes uniqueness and stability of the solution and it can encode a-priori information about x. For instance, one can choose $J(x) = \|x\|_2^2$ to obtain the smallest norm solution (Tikhonov [220]), or $J(x) = \|x\|_1$ to encourage sparsity (LASSO [219]), or $J(x) = \mathbb{1}_{\mathcal{X}}(x)$ to obtain a solution in $\mathcal{X} \subset X$ (regularization by projection [228]), or $J(x) = \|\nabla x\|_1$ to obtain piecewise constant solutions (total variation [195]). For a detailed review of these methods we refer to the literature [79, 202].

Tikhonov regularization. Consider the minimization problem

$$\min_{x \in X} \frac{1}{2}\|\mathcal{F}(x) - y\|_2^2 + \frac{\lambda}{2}\|x\|_2^2, \tag{1.4}$$

where $\mathcal{F}\colon X \to Y$ represents a forward differentiable operator with X and Y Hilbert spaces, and $\lambda > 0$ is a regularization parameter. As the functional in (1.4) is differentiable, a gradient-type algorithm can be applied to find the solution to the minimization problem (as discussed in [79, Chapter 11.1]). The iterative algorithm proceeds as follows

$$x_{k+1} = x_k - t((\mathcal{F}'(x_k))^*(\mathcal{F}(x_k) - y) + \lambda x_k),$$

where $t > 0$ represents a stepsize and $(\cdot)^*$ denotes the adjoint operator.

If the forward operator $\mathcal{F}(x) = Ax$ is linear and compact, then the solution to (1.4) (as discussed in [79, Chapter 5.1]) is given by

$$\bar{x} = (A^*A + \lambda I)^{-1}A^*y,$$

where A is the linear forward operator.

LASSO regularization. Consider the minimization problem

$$\min_{x \in X} \frac{1}{2}\|\mathcal{F}(x) - y\|_2^2 + \lambda\|x\|_1, \tag{1.5}$$

where $\mathcal{F}\colon X \to Y$ is a forward differentiable operator with $X = \mathbb{R}^n$ and $Y = \mathbb{R}^m$, and $\lambda > 0$ is a regularization parameter. The functional in (1.5) consists of a differentiable term $g(x) = \frac{1}{2}\|\mathcal{F}(x) - y\|_2^2$ and a non-differentiable term $f(x) = \|x\|_1$. Therefore a proximal gradient descent method can be employed (as discussed in [181]). The solution to (1.5) is iteratively obtained using the following update rule

$$x_{k+1} = \text{prox}_{t\lambda f}(x_k - t\nabla g(x_k)),$$

where $t > 0$ represents a stepsize and $\text{prox}_{t\lambda f}(\cdot) := \arg\min_x(f(x) + \frac{1}{2t\lambda}\|x - \cdot\|_2^2)$ indicates the proximal operator of $t\lambda f$. In the specific case of $f(x) = \|x\|_1$, the proximal operator of $t\lambda f$ corresponds to the shrinkage/soft thresholding operator $S_{t\lambda}\colon \mathbb{R}^n \to \mathbb{R}^n$ defined componentwise as

$$S_{t\lambda}(x)^i = \max\{|x^i| - t\lambda, 0\}\,\text{sign}(x^i),$$

where i indicates the component. From this, the proximal gradient method is named the iterative soft thresholding algorithm (ISTA) [70, 107]. In this case the penalty term $f(x) = \|x\|_1$ promotes the sparsity of the solution w.r.t. the canonical basis. However, it is useful to consider a penalty term that promotes sparsity w.r.t. a generic basis by taking $f(x) = \|Mx\|_1$ where M is a matrix representing the change of basis from the canonical basis to the basis on which the unknown is sparse. This approach enables us to represent more general classes of data, particularly those that can be accurately characterized using only a limited number of components or features within a specific domain. This is motivated by the desire to reduce the dimensionality, and, consequently, the complexity of problems. In real-world scenarios, sparsity prior are frequently employed because signals and images are naturally sparse in certain representations.

If the forward operator $\mathcal{F}(x) = Ax$ is linear and compact, then the solution to (1.5) is obtained using the update rule

$$x_{k+1} = S_{t\lambda}(x_k - tA^*(Ax_k - y)),$$

where A is the linear forward operator. Considering instead the penalty term $f(x) = \|Mx\|_1$ discussed above, the update rule becomes

$$x_{k+1} = M^T S_{t\lambda}(M(x_k - tA^*(Ax_k - y))).$$

Regularization by projection. Consider the minimization problem

$$\min_{x\in X} \frac{1}{2}\|y - \mathcal{F}(x)\|_2^2 + \mathbb{1}_{\mathcal{X}}(x) \tag{1.6}$$

where $\mathcal{F}\colon X \to Y$ is a forward differentiable operator with X and Y Hilbert spaces, and $\mathcal{X}$ represents a subspace to which the solution is supposed to belong [79, Section 3.3]. The functional in (1.6) consists of a differentiable term $g(x) = \frac{1}{2}\|\mathcal{F}(x) - y\|_2^2$ and a non-differentiable term $f(x) = \mathbb{1}_{\mathcal{X}}(x)$. Therefore, similar to LASSO, the solution to (1.6) is obtained iteratively using

$$x_{k+1} = \text{prox}_{tf}(x_k - t\nabla g(x_k)),$$

where $t > 0$ represents a stepsize. Here, the penalty term $f(x)$ is the indicator function of the subset $\mathcal{X} \subset X$, defined as

$$\mathbb{1}_{\mathcal{X}}(x) = \begin{cases} 0 & x \in \mathcal{X} \\ +\infty & \text{otherwise} \end{cases}$$

and forces the unknown to belong to $\mathcal{X}$. Assuming that $\mathcal{X}$ is a closed linear subspace of X, the proximal operator of the indicator function is the orthogonal projection onto $\mathcal{X}$, namely

$$\text{prox}_{\mathbb{1}_{\mathcal{X}}}(x) = P_{\mathcal{X}}(x),$$

from which the term *regularization by projection* is derived.

If the forward operator is linear and compact, then the solution to (1.6) is obtained using the updating rule

$$x_{k+1} = P_{\mathcal{X}}(x_k - tA^*(Ax_k - y)),$$

where A is the linear forward operator.

This technique effectively regularizes inverse problems only when the subspace $\mathcal{X}$ is significantly smaller then the entire space X, because dimensionality reduction enhances stability. Moreover, $\mathcal{X}$ should be appropriately chosen based on the prior knowledge on the data. For these reasons, $\mathcal{X}$ is often selected as a low-dimensional manifold of X that is inferred from the data [34, 56, 77].

1.1.2 Deep Learning

Deep learning [92, 147] is a highly influential subfield of machine learning that focuses on the development and application of artificial neural networks with multiple layers. Its primary objective is to tackle complex tasks across various domains, such as computer vision, natural language processing, robotics, healthcare, finance, and more. These neural networks are designed to approximate intricate functions, including the inverse map of an IP [83]. Deep learning has made significant strides in addressing challenges such as image classification [154], object detection [236], speech recognition [74], recommendation systems [235], and autonomous driving [99].

Deep neural networks possess remarkable versatility, enabling them to handle extensive datasets with high dimensionality and extract information from them. They exhibit a remarkable expressive power, allowing them to represent complex relationships within the data. Moreover, deep neural networks are not only applied as discriminative models [119, 126, 145] but also serve as generative models [133, 180, 197]. They have the ability to generate synthetic data that closely resembles the training data, thereby enabling tasks such as data augmentation and simulation [207].

1.1.2.1 Neural Network Architectures

Deep Neural networks are composed by basic building blocks, called *layers*. A layer typically consists of an affine map containing learned weights, which parametrize both the linear term and the bias, and a fixed pointwise nonlinearity, called *activation function*. More precisely, the output of the l-th layer is

$$x_l = \sigma_l(W_l x_{l-1} + b_l),$$

where $x_{l-1} \in X_{l-1}$ is the output of the $l-1$-th layer, X_{l-1} is the output space of the $l-1$-th layer, $\sigma_l \colon \mathbb{R} \to \mathbb{R}$ is the activation function of the l-th layer, $W_l \colon X_{l-1} \to X_l$ is the linear map of the l-th layer and $b_l \in X_l$ is the bias of the l-th layer. Therefore a neural network $\mathcal{N} \colon X \to Y$ composed by L layers can be written as

$$\mathcal{N}(x_0) = \sigma_L(W_L \sigma_{L-1}(W_{L-1} \cdots \sigma_1(W_1 x_0 + b_1) \cdots + b_{L-1}) + b_L), \tag{1.7}$$

where $X = X_0$, $Y = X_L$ and $x_0 \in X_0$ is the input signal. In the discrete setting, $X_l = \mathbb{R}^{d_l}$, where d_l is the output size of the l-th layer. If there are no restrictions on the linear map, the layer is called *dense layer* or *fully connected layer*.

However, many of the most successful architectures for image processing are based on *convolutional layers*, which restrict the linear map in each layer to be a convolution [148, 146]. Convolutional layers respect the spatial structure of the image while having considerably fewer free parameters than the corresponding dense layers. Convolutional networks typically add an additional dimension to the horizontal and vertical image dimensions, called channel dimension, representing the number of images in each layer. Many classical architectures increase the number of channels in deeper layers, while at the same time reducing the spatial dimension or vice versa. The spatial dimension can be reduced by inserting pooling layers, that combine groups of adjacent pixels into a single one by taking the maximum or mean value of the pixels, or using a strided convolutional layer, in which the convolutional filter is moved by a specific step along the input image and the stepsize determines the size of the output image. This latter technique also allows to increase the spatial dimension by applying the adjoint operator of the strided convolution. While dimensional reduction is used for classification and features extraction, increasing the dimension is specific for image generation problems. We will explore some generative models in the following paragraph.

Concerning the activation functions, the most common are the rectified linear unit (ReLU), defined as $\sigma(x) = \max(x, 0)$, and various generalisations, including the exponential linear unit (ELU), the Gaussian error linear unit (GELU) and the leaky ReLU. Other common activation functions are the sigmoid functions, including the logistic function $\sigma(x) = \frac{e^x}{1+e^x}$, the hyperbolic tangent and the arctangent.

Generative Neural Networks. Generative neural networks [133, 180, 197] are a class of models capable to generate new samples that closely resemble a given dataset. These networks learn the complex underlying distribution of the data by estimating their parameters, effectively mapping a simpler probability distribution (such as a Gaussian distribution) to the underlying data distribution. Then synthetic data samples can be generated by applying the generator to samples from the simple distribution. Generative neural networks have diverse applications, including data augmentation [207], image inpainting [230], anomaly detection [199], and text generation [156]. In the context of inverse problems, these networks can be employed to impose prior knowledge on the unknowns, constraining them to lie within the range of the generator.

In many cases the elements within the dataset of interest belong to the same "category" (e.g. a dataset of lungs for the EIT problem [204]), therefore one can assume that they

live approximately on a low-dimensional manifold [34]. This manifold can be learned using one or more generative models [125, 164].

The most famous generative model architectures are

- Variational autoencoders (VAEs) [134]: VAEs consist of two neural networks, namely an encoder and a decoder. The encoder maps the input data distribution to a simpler probability distribution (often Gaussian) in a low-dimensional latent space. The decoder, which assumes the role of the generator in our research, takes samples from this distribution to reconstruct the input data. Unlike traditional autoencoders, VAEs incorporate a probabilistic approach, modeling data as a distribution rather than a single point in the latent space.

- Generative adversarial networks (GANs) [93]: GANs consist of two neural networks, namely a generator and a discriminator. The generator takes random noise as input and aims to produce synthetic samples that resemble the real data, while the discriminator's goal is to differentiate between real and fake samples. These networks are trained in an adversarial manner, competing against each other to enhance their respective performances.

- Normalizing flows (NFs) [192]: NFs employ sequences of invertible and differentiable transformations to map simple probability distributions to the target data distributions. Each transformation in the flow is designed to be easily invertible, enabling efficient sampling and log-likelihood computation. The simple distribution has the same dimension as the data distribution.

- Diffusion models (DMs) [227]: DMs model the diffusion process by gradually introducing noise, allowing them to learn the systematic decay of information caused by the noise. They can subsequently reverse the process and recover the original information from the noisy data.

1.1.2.2 Training a Neural Network

Training a neural network involves determining the parameters θ of a neural network $\mathcal{N}_\theta$ in order to achieve the best possible approximation of the function $\mathcal{F}\colon X \to Y$ that is of interest to us. In the fully connected neural network designed in (1.7), the parameters to learn are the weight matrices W_l and the biases b_l. In the supervised learning setting, we are provided with a labeled training set $\{(x_i, y_i)\}_{i=1}^N$, where x_i are elements of X and y_i approximate $\mathcal{F}(x_i)$. In this scenario, the parameters θ are learned by minimizing the empirical risk, namely

$$\bar{\theta} = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}(\mathcal{N}_\theta(x_i), y_i), \tag{1.8}$$

where $\mathcal{L}\colon Y \times Y \to \mathbb{R}$ denotes a loss function that typically depends on the specific problem. For the binary classification problem, the most common losses are hinge loss $\mathcal{L}(\hat{y}, y) = \max\{0, 1-\hat{y}y\}$ and binary cross-entropy/logistic loss $\mathcal{L}(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$, while, for classification, categorical cross-entropy $\mathcal{L}(\hat{y}, y) = -y\log(\hat{y})$ is the most used.

For regression problem, the most common losses are the mean squared error/L^2 Loss $\mathcal{L}(\hat{y}, y) = \|\hat{y} - y\|_2^2$ and the mean absolute error/L^1 Loss $\mathcal{L}(\hat{y}, y) = \|\hat{y} - y\|_1$.

The existence and uniqueness of the solution of (1.8) are not guaranteed, as the problem is in general non-convex. Anyway, in practice, the minimization process is performed using stochastic gradient descent. This approach involves randomly initializing the parameters θ and iteratively updating them by applying the following rule

$$\theta_{n+1} = \theta_n - \lambda \nabla_\theta \left(\frac{1}{k} \sum_{i \in \mathcal{I}_n} \mathcal{L}(\mathcal{N}_\theta(x_i), y_i) \right)$$

where λ is the stepsize and $\mathcal{I}_n \subset \{1, ..., N\}$ with $|\mathcal{I}_n| = k$ represents randomly selected mini-batches derived from the training set. During each iteration, a mini-batch is randomly chosen. In practical implementations, optimization algorithms commonly incorporate momentum, such as the ADAM algorithm [131]. In any case, it is necessary to compute the gradient $\nabla_\theta \mathcal{N}_\theta$. This can be achieved using automatic differentiation.

VAEs training. For generative models the training set comprises only $\{x_i\}_{i=1}^N$, where the data samples x_i are taken from the data distribution. In this unsupervised setting, where no labels are available, the minimization strategy outlined in (1.8) is no longer applicable. Since in our work we choose to use decoders of VAE models as generators, we review the training strategy employed for these models.

As mentioned earlier, the VAE model consists of an encoder E_ϕ with parameters ϕ, which maps from the data space to the latent space, and a decoder D_θ with parameters θ, which performs the reverse mapping. The encoder approximates the posterior distribution $q_\phi(z|x)$, where z belongs to the latent space and x belongs to the data space, while the decoder represents the likelihood $p_\theta(x|z)$. In standard VAEs the prior distribution is $p(z) \sim \mathcal{N}(\mathbf{0}, I)$ and the new samples are obtained by applying the decoder to a latent vector sampled from $\mathcal{N}(\mathbf{0}, I)$. Therefore the training process for finding the parameters ϕ and θ consists of minimizing both the reconstruction error between the original data samples x_i and their reconstructions $D_\theta(E_\phi(x_i))$ and the distance loss between $q_\phi(z|x)$ and $p_\theta(z|x)$. Mean squared error and cross-entropy are commonly employed as reconstruction loss, while the reverse Kullback–Leibler divergence is widely used as distance loss between two distributions. We refer the reader to [135] for more information.

1.1.3 Deep Learning for Inverse Problems

Deep learning techniques for solving inverse problems often yield better results compared to standard approaches, particularly when large training sets are available [233]. Indeed, they represent the state-of-the-art in many inverse problems, e.g. image denoising [218], deblurring [234], MRI [60] and so on. Thanks to the flexibility and the adaptability of deep neural networks, they can be used in many ways for solving IPs both independently and by combining them with standard techniques. The first approach involves fully learned methods, where the goal is to directly approximate the inverse operator using a neural network. In the second case the neural networks can be used to

1. post process the solution,
2. learn an iterative scheme,
3. learn a regularization operator,
4. encode a-priori information on the unknown.

In our research, our focus lies on the last two topics, but for completeness we provide a brief overview of all the techniques mentioned earlier. For a detailed review on deep learning methods for solving inverse problems we refer the reader to [23, 27, 88, 176, 198, 206].

Fully Learned methods. Fully learned methods aim to approximate the inverse operator that maps from a measurement y to an unknown variable x using a single neural network. The network's parameters are trained using a supervised approach, as described in equation (1.8), without requiring knowledge of the forward operator. Consequently, these methods prove particularly valuable when the forward map is either unknown or only partially known. The effectiveness of these techniques can be influenced by the architecture of the neural network. A widely adopted architecture, introduced in [237], represents the network as a composition of three mappings: $\phi_x \circ g \circ \phi_y^{-1}$. In this formulation, ϕ_x and ϕ_y are maps from the manifolds where the unknowns and the measurements belong, respectively, to Euclidean space, while g is a diffeomorphism between the two manifolds. These maps are parameterized using sequences of fully connected layers (ϕ_y) or sequences of convolutional layers (ϕ_x and g). Fully learned techniques for solving inverse problems are also employed in other works such as [225, 83, 200]. The performances obtained using fully learned methods instead of classical ones for solving inverse problems depend on the specific problem and on the architecture of the network and cannot be generalized.

Post-processing methods. Post-processing methods involve a two-step approach. Firstly, an initial solution to the inverse problem is computed using a standard technique, such as employing the pseudo-inverse for linear IPs. Then, the solution is refined by mitigating artifacts through post-processing, which is accomplished using a neural network trained as a denoiser or deblurring operator. These networks often adopt a convolutional structure, with U-net and similar encoder-decoder architectures being commonly employed [128, 130, 111]. Post-processing techniques find applications in various medical inverse problems, including CT [55], MRI [224], PET [67], EIT [108], among others. In a similar way it is also possible to preprocess the measurements using neural network [16, 120, 150].

Learning an iterative scheme Learning an iterative scheme involves using a neural network with layers designed to mimic the iterations of a traditional iterative scheme. These techniques are often referred to as "unrolling techniques" because the iterative schemes are "unrolled" and replaced with neural networks. In cases where both terms of

the functional in (1.3) are differentiable, these networks can be employed to approximate the gradient of that functional, resulting in a learned gradient descent scheme [3]. Furthermore, neural networks can be employed to learn proximal primal-dual methods [4] or proximal gradient descent algorithms [171]. Another approach involves integrating neural networks into iterative schemes by applying them after each iteration. For instance, in the plug-and-play method [129, 193], each iteration consists of a gradient descent step followed by the application of a neural network denoiser.

Learning a Regularizer. In Section 1.1.1.1 we discussed some possible choices of the regularization term J in (1.3). However, since the appropriate choice of J should align with the specific characteristics of the data, it is possible to learn it from the data through a neural network. More precisely, the penalty term is represented by a neural network J_θ, where θ are the network's parameters. The parameters of the network and the solution of the inverse problem can be found by minimizing $\mathcal{L}(\mathcal{F}(x), y) + J_\theta(x)$ in both variables, θ and x. This can be done, e.g., using bilevel optimization (refer to [23, Section 4.3]).

The methods that involve learning a regularizer differ in terms of architecture and training strategy of the regularizer. Among these techniques, we mention the neural network Tikhonov (NETT) approach [153] and the adversarial regularizer method [162]. The NETT regularizer is composed by an auto-encoding structured network and a regularization functional. The training process for this network closely resembles that of autoencoders. Subsequently, the solution to the inverse problem is obtained by minimizing the variational problem with respect to the only remaining variable, which is x. This is achievable because the network's parameters remain fixed after training. Also for the adversarial regularization method, the training of the regularizer occurs before the minimization of the variational functional. However, as the name suggests, the training process takes an adversarial approach, involving the discrimination between the prior distribution and the distribution of imperfect solutions to the inverse problem.

To ensure convergence guarantees for the variational reconstruction problem, the inclusion of convex regularizers has been explored [175]. In this context, the regularizer is an input convex neural network [19].

Encode a-priori information on the unknown. Generative neural networks are often used for modeling a-priori information on the unknown of inverse problems, namely the unknown is assumed to belong to the image of a generator [24, 44, 104, 204]. This assumption imposes a structure on the unknown, indeed the unknown x is supposed to be $G_\theta(z)$ where $G_\theta \colon Z \to X$ represents a generator depending on the parameters θ and z is sampled from the latent space Z. Therefore, the regularized solution is

$$\mathcal{R}_\theta(y) = G_\theta(z') \quad \text{where } z' \in \operatorname*{arg\,min}_{z \in Z} \mathcal{L}(\mathcal{F}(G_\theta(z)), y).$$

In this approach there is no need for a penalty term, as the regularization is implicitly provided by the generator. There exist other approaches in which generative models without training data are used for modeling prior information [221].

1.2 Thesis overview and contribution

In this section, we provide an overview of the contributions presented in this thesis, which are divided into three chapters. The results from these chapters are discussed, respectively, in the papers

I. G. S. Alberti, M. Santacesaria, and S. Sciutto, *Continuous generative neural networks*, 2022, [12]. Revision submitted to Inverse Problems.

II. G. S. Alberti, J. Hertrich, M. Santacesaria, and S. Sciutto, *Manifold learning by mixture models of VAEs for inverse problems*, 2023, [8]. Revision submitted to Journal of Machine Learning Research.

III. G. S. Alberti, L. Ratti, M. Santacesaria, and S. Sciutto, *Learning a Gaussian mixture for sparsity regularization in inverse problems*, 2024, [9]. Submitted to IMA Journal of Numerical Analysis.

1.2.1 Continuous Generative Neural Networks for Inverse Problems

In Chapter 2, we design a generative neural network in an infinite-dimensional setting and we use it for modeling the unknown variables in inverse problems. The primary contributions in this direction are as follows.

- Designing a generative neural network $G\colon \mathbb{R}^S \to L^2$ that takes low-dimensional vectors as input and generates functions in L^2 as output, namely, a generator in an infinite-dimensional setting that we call continuous generative neural network (CGNN). This is useful because many physical quantities of interest are better modeled as functions than vectors, e.g. solutions of partial differential equations.

- Defining the concept of strided convolution in the L^2 setting, where the dimensions of the spaces of each layer are replaced by the scales of a multiresolution analysis of a compactly supported wavelet. This adaptation is necessary to accommodate the architecture of CGNN, which draws inspiration from DCGAN. It includes a fully connected layer, multiple strided convolutional layers, and nonlinear activation functions.

- Finding a set of sufficient conditions under which a CGNN is injective. These assumptions seem very natural as they entail the linear independence of the scaling coefficients of the convolutional filters and the injectivity of the nonlinearities. Injectivity is fundamental when using CGNN to solve IPs, and it is not trivial in our setup due to the projections needed for our definition of continuous strided convolution and the reduction of the channels in each layer of the network.

- Proving that, if G is injective, $\mathcal{M} := G(\mathbb{R}^S)$ is an S-dimensional differentiable manifold embedded in L^2 whose only chart is G^{-1}.

- Guaranteeing Lipschitz stability for (possibly nonlinear) infinite-dimensional inverse problems where the unknowns belong to the manifold generated by an injective CGNN, under the assumption that the forward map and its derivative are injective.
- Providing numerical comparisons between discrete generative neural networks and our CGNNs, obtained using different Daubechies scaling functions, in signal generation and in a signal deblurring problem. The signal class consists of truncated Fourier series. The CGNN is trained as a decoder of a VAE.

1.2.2 Manifold Learning by Mixture Models of Variational Autoencoders for Inverse Problems

In Chapter 3, our investigation focuses on learning manifolds through mixture models of variational autoencoders. This approach enables the representation of manifolds with arbitrary topologies, as it incorporates multiple charts represented by encoder-decoder pairs of VAEs. This capability is crucial when dealing with real datasets, as they may lack a global parameterization, making it impossible to represent them as a single-chart manifold. The transition from learning one-chart manifolds to learning multiple-charts manifolds can be viewed as an extension of the previous chapter, however, in this context, we present a theoretical framework within the finite-dimensional setting. The contributions of this research are here summerized.

- Representing manifolds of arbitrary dimension and topology by learning a mixture model of VAEs. The manifolds charts are represented by the inverse functions of the injective decoders. In this sense, we extend the theory developed in the previous chapter. However, in this chapter, we consider VAEs in the finite-dimensional setting with a very different architecture from CGNN. Specifically, each decoder is the composition of a normalizing flow, which learns the latent space's structure, and a series of invertible neural networks and fixed linear operators, that increase the dimensionality and map the learned latent space to the data distribution.
- Proposing a novel training strategy for learning the data manifold by deriving a loss function for maximum likelihood estimation of the model weights. The loss considers the probability, estimated using the evidence lower bound (ELBO), of each training data to belong to each chart of the manifold. Furthermore, in accordance with the manifold's definition, we make slight adjustments to the ELBO definition to allow for overlapping charts.
- Proposing a Riemannian gradient descent scheme for minimizing a functional defined on a multiple-charts manifold. The necessity of employing the Riemannian gradient becomes apparent when dealing with multiple-charts manifolds. This is because gradient descent steps taken with respect to different charts may lead in completely different directions, rendering it impractical to establish a coherent gradient descent algorithm. In contrast, the Riemannian gradient remains consistent regardless of the manifold's parameterization, providing a more reliable approach.

- Providing numerical comparisons between a single generator (leading to a manifold with a single chart) and multiple generators (leading to a manifold with multiple charts) in manifold generation (two circles, ring, sphere, swiss roll, torus) and minimization of functionals defined on these manifolds, and for solving the image deblurring problem and the EIT problem.

1.2.3 Learning a Gaussian Mixture for Sparse Optimization in Inverse Problems

In Chapter 4, our focus lies on designing a neural network that represents the Bayesian estimator for a discrete linear inverse problem with Gaussian mixture prior, i.e. using the prior knowledge that the unknown is sampled from a Gaussian mixture model. Additionally, we observe that this choice of prior also aligns with a reasonable sparsity assumption by using degenerate Gaussian components within the mixture. The key contributions of this research are the following.

- Representing the Bayesian estimator for a finite-dimensional linear inverse problem with Gaussian mixture prior as a two-layers feed-forward neural network. The hidden layer involves non-standard operations on the input variables and exhibits similarities with the attention mechanism of the transformer architecture.

- Justifying the usage of a Gaussian Mixture prior as a "group" sparsity prior for inverse problems. Sparsity is obtained by considering degenerate Gaussian into the mixture and the groups are marked by the different Gaussians of the mixture. It's worth noting that we introduced a probabilistic prior that encourages sparsity, even though standard probabilistic priors are not typically well suited for their sparsity-promoting properties.

- Proposing both a supervised and an unsupervised methods for learning the neural network weights, which correspond to the parameters of the Gaussian mixture. The first technique entails minimizing the empirical risk, while the second involves endowing the network with the empirically estimated parameters of the mixture derived from the training set.

- Comparing numerically our proposed methods with other sparsity algorithms, including LASSO, Group LASSO, iterative hard thresholding (IHT) and dictionary learning for 1D denoising and deblurring problem using various types of datasets. These are designed for representing group sparsity: the first dataset is directly generated from a degenerate Gaussian mixture, while the others consist of smooth functions with one or two jumps (discontinuities) with "Gaussian" amplitude. In the latter case, the functions can be partitioned into groups based on the positions of these discontinuities, and the sparsity becomes evident in the wavelets domain. In particular, thanks to the smoothness of the signals, the wavelet coefficients relative to high scales appear sparse. For the numerical experiments, our algorithms, that do not necessitate prior knowledge of the basis on which the dataset exhibits sparsity,

offer superior reconstruction performances compared to classical sparsity-promoting methods that require estimating the basis beforehand.

Chapter 2

Continuous Generative Neural Networks for Inverse Problems

Deep generative models are a large class of deep learning architectures whose goal is to approximate high-dimensional probability distributions [196]. A trained model is then able to easily generate new realistic samples. They have received huge interest in the last decade both for very promising applications in physics [73, 179], medicine [110, 170, 210], computational chemistry [223, 226] and more recently also for their worrying ability in producing realistic fake videos, a.k.a. deepfakes [100]. Several architectures and training protocols have proven to be very effective, including variational autoencoders [134], generative adversarial networks [93], normalising flows [192, 75] and diffusion models [227].

In this chapter we consider a generalization of some of these architectures to a continuous setting, where the samples to be generated belong to an infinite-dimensional function space. One reason is that many physical quantities of interest are better modeled as functions than vectors, e.g. solutions of partial differential equations. In this respect, this work fits in the growing research area of neural networks in infinite-dimensional spaces, often motivated by the study of PDEs, which includes neural operators [142], Deep-O-Nets [161], PINNS [190] and many others. The general goal of these works is to approximate an operator between infinite-dimensional function spaces (e.g. the parameter-to-solution map of a PDE) with a neural network that does not depend on the discretization of the domain.

A second reason concerns the promising applications of generative models in solving inverse problems. A typical inverse problem consists in the recovery of a quantity from noisy observations that are described by a ill-posed operator between function spaces [79]. Virtually, every imaging modality can be modeled in such a way, including computed tomography, magnetic resonance imaging and ultrasonography. In recent years, machine learning based reconstruction algorithms have become the state of the art in most imaging applications [23, 176]. Among these algorithms, the ones combining generative models with classical iterative methods – such as the Landweber scheme – are very promising since they retain most of the *explanaibility* provided by inverse problems theory. However,

despite the impressive numerical results [44, 221, 22, 123, 204, 24, 122], many theoretical questions have not been studied yet, for instance concerning stability properties of the reconstruction. In the context of inverse problems, there are several super-resolution method able to produce a continuous representation of an image [59, 78, 211, 212, 217, 102, 101]. In particular, in [102] the authors consider generative convolutional neural networks in function spaces, but without incorporating the concepts of strided convolution including upscaling and downscaling, typical of discrete convolutional networks. Further, none of these approaches is based on the wavelet decomposition, the tool used in the present work, which naturally deals with continuous signals.

In this work (Section 2.1), we introduce a family of continuous generative neural networks (CGNNs), mapping a finite-dimensional space into an infinite-dimensional function space. Inspired by the architecture of deep convolutional GANs (DCGANs) [189], CGNNs are obtained by composing an affine map with several (continuous) convolutional layers with nonlinear activation functions. The convolutional layers are constructed as maps between the subspaces of a multi-resolution analysis (MRA) at different scales, and naturally generalize discrete convolutions. In our continuous setting, the scale parameter plays the role of the resolution of the signal/image. We note that wavelet analysis has been used in the design of deep learning architectures in the last decade [167, 47, 20, 64, 21].

The main result of this chapter (Section 2.2) is a set of sufficient conditions that the parameters of a CGNN must satisfy in order to guarantee global injectivity of the network. This result is far from trivial because in each convolutional layer the number of channels is reduced, and this has to be compensated by the higher scale in the MRA. Generative models that are not injective are of no use in solving inverse problems or inference problems, or at least it is difficult to study their performance from the theoretical point of view. In the discrete settings, some families of injective networks have been already thoroughly characterized [33, 151, 187, 80, 141, 188, 105, 104]. Note that normalizing flows are injective by construction, yet they are maps between spaces of the same (generally large) dimension, a feature that does not necessarily help with our desired applications.

Indeed, another useful property of CGNNs is dimensionality reduction. For ill-posed inverse problems, it is well known that imposing finite-dimensional priors improves the stability and the quality of the reconstruction [15, 37, 35, 14, 38], also working with finitely-many measurements [10, 113, 6, 11, 5]. In practice, these priors are unknown or cannot be analytically described: yet, they can be approximated by a (trained) CGNN. The second main result of this work (Section 2.3) is that an injective CGNN allows us to transform a possibly nonlinear ill-posed inverse problem into a Lipschitz stable one. Our stability estimate in Theorem 8 is tightly connected to the works on compressed sensing for generative models (e.g. [44]), because they both deal with stability for inverse problems under the assumption that the unknown lies in the image of a generative model. However, in our setup the model is infinite-dimensional, the forward map is possibly nonlinear, and its inverse may not be continuous even with full measurements.

As a proof-of-concept (Section 2.4), we show numerically the validity of CGNNs in

performing signal deblurring on a class of one-dimensional smooth signals. The numerical model is obtained by training a VAE whose decoder is designed with an injective CGNN architecture. The classical Landweber iteration method is used as a baseline to compare CGNNs deriving from different orthogonal wavelets and the correspondent discrete generative neural network. We also provide some qualitative experiments on the expressivity of CGNNs. However, we would like to emphasize that the main contributions of this work are theoretical, and that the main goal of our experiments is not to obtain state of the art results, but only to compare continuous and discrete generative neural networks for a toy class of smooth signals. The application to real-world data and the combination with other methods in order to achieve competitive results are not within the scope of this work and are left to future research.

2.1 Architecture of CGNNs

We first review the architecture of a DCGAN [189], and then present our continuous generalization. For simplicity, the analysis is done for 1D signals, but it can be extended to the 2D case (see Section 2.2.2).

2.1.1 1D discrete generator architecture

A deep generative model can be defined as a map $G_\theta \colon \mathbb{R}^S \to X$, where X is a finite-dimensional space with $S \leq \dim(X)$, constructed as the forward pass of a neural network with parameters θ. Our main motivation being the use of generators in solving ill-posed inverse problems, we consider generators that allow for a dimensionality reduction, i.e. $S \ll \dim(X)$, which will yield better stability in the reconstructions.

As a starting point for our continuous architecture, we then consider the one introduced in [189]. It is a map $G \colon \mathbb{R}^S \to X$ (we drop the dependence on the parameters θ) obtained by composing an affine fully connected (f.c.) layer and L convolutional layers with nonlinear activation functions. More precisely:

$$G \colon \mathbb{R}^S \xrightarrow[\text{f.c.}]{\Psi_1} (\mathbb{R}^{\alpha_1})^{c_1} \xrightarrow[\text{nonlin.}]{\sigma_1} (\mathbb{R}^{\alpha_1})^{c_1} \xrightarrow[\text{conv.}]{\Psi_2} (\mathbb{R}^{\alpha_2})^{c_2} \xrightarrow[\text{nonlin.}]{\sigma_2} (\mathbb{R}^{\alpha_2})^{c_2} \xrightarrow[\text{conv.}]{\Psi_3} \cdots \cdots \xrightarrow[\text{conv.}]{\Psi_L} \mathbb{R}^{\alpha_L} \xrightarrow[\text{nonlin.}]{\sigma_L} \mathbb{R}^{\alpha_L} = X,$$

which can be summarized as

$$G = \left(\bigcirc_{l=L}^{2} \sigma_l \circ \Psi_l \right) \circ (\sigma_1 \circ \Psi_1) .$$

The natural numbers $\alpha_1, \ldots, \alpha_L$ are the vector sizes and represent the resolution of the signals at each layer, while $c_1, \ldots, c_L$ are the number of channels at each layer. The output resolution is α_L. Generally, one has $\alpha_1 < \alpha_2 < \cdots < \alpha_L$, since the resolution increases at each level. Moreover, we impose that α_l is divisible by α_{l-1} for every $l = 2, \ldots, L$. We now describe the components of G.

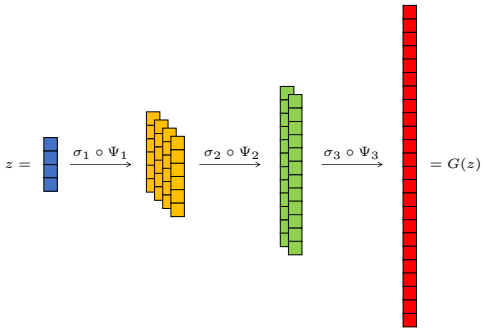


Figure 2.1: Example of discrete generator's architecture. The first fully connected layer maps the latent space $\mathbb{R}^4$ into a 4 channel space of vectors of length 6. Then there are two convolutional layers with stride $\frac{1}{2}$ which halve the number of channels and double the length of the vectors, until obtaining one vector of length 24. In this example, the increase in dimensionality occurs only in the first layer, while in the convolutional layers the dimension of the spaces remains constant.

The nonlinearities. Each layer includes a pointwise nonlinearity $\sigma_l \colon (\mathbb{R}^{\alpha_l})^{c_l} \to (\mathbb{R}^{\alpha_l})^{c_l}$, i.e. a map defined as

$$\sigma_l(x_1, \dots, x_{\alpha_l \cdot c_l}) = (\sigma(x_1), \dots, \sigma(x_{\alpha_l \cdot c_l})),$$

with $\sigma \colon \mathbb{R} \to \mathbb{R}$ nonlinear.

The fully connected layer. The first layer is $\Psi_1 := F \cdot + b$, where $F \colon \mathbb{R}^S \to (\mathbb{R}^{\alpha_1})^{c_1}$ is a linear map and $b \in (\mathbb{R}^{\alpha_1})^{c_1}$ is a bias term.

The convolutional layers. The convolutional layer $\Psi_l \colon (\mathbb{R}^{\alpha_{l-1}})^{c_{l-1}} \to (\mathbb{R}^{\alpha_l})^{c_l}$ represents a fractional-strided convolution with stride $s = \frac{\alpha_{l-1}}{\alpha_l}$ such that $s^{-1} \in \mathbb{N}^*$, where c_{l-1} is the number of input channels and c_l the number of output channels, with $c_l < c_{l-1}$. This convolution with stride s corresponds to the transpose of the discrete convolution with stride s^{-1}, and is often called deconvolution. It is defined by

$$(\Psi_l x)_k := \sum_{i=1}^{c_{l-1}} x_i *_s t^l_{i,k} + b^l_k, \qquad k = 1, ..., c_l,$$

where $t^l_{i,k} \in \mathbb{R}^{\alpha_l}$ are the convolutional filters and $b^l_k \in \mathbb{R}^{\alpha_l}$ are the bias terms, for $i = 1, ..., c_{l-1}$ and $k = 1, ..., c_l$. The operator $*_s$ is defined as

$$(x *_s t)(n) := \sum_{m \in \mathbb{Z}} x(m)\, t(n - s^{-1}m), \tag{2.1}$$

where we extend the signals x and t to finitely supported sequences by defining them zero outside their supports, i.e. $x, t \in c_{00}(\mathbb{Z})$, where $c_{00}(\mathbb{Z})$ is the space of sequences with finitely many nonzero elements.

Note that the most significant dimensional increase occurs in the first layer, the fully connected one. Indeed, after the first layer, in the fractional-strided convolutional layers the increase of the vectors' size is compensated by the decrease of the number of channels; see Figure 2.1 for an illustration. At each layer, the resolution of a signal increases thanks to a deconvolution with higher-resolution filters. The final output is then a single high-resolution signal.

We motivate (2.1) by taking the adjoint of the strided convolutional operator $A_{s^{-1},t}$ with stride $s^{-1} \in \mathbb{N}^*$ and filter t, which is defined as $A_{s^{-1},t}\, x = x *_{s^{-1}} t$, where

$$(x *_{s^{-1}} t)(n) := (x * t)(s^{-1}n) = \sum_{m\in\mathbb{Z}} x(m)\, t(s^{-1}n - m). \tag{2.2}$$

As before, the signals x and t are seen as elements of $c_{00}(\mathbb{Z})$ by extending them to zero outside their supports and the symbol $*$ denotes the discrete convolution

$$x * t := \sum_{m\in\mathbb{Z}} x(m)\, t(\cdot - m), \qquad x, t \in c_{00}(\mathbb{Z}). \tag{2.3}$$

The adjoint of $A_{s^{-1},t}$, $A^*_{s^{-1},t}$, satisfies

$$\langle A^*_{s^{-1},t}\, y, x\rangle_2 = \langle y, A_{s^{-1},t}\, x\rangle_2, \tag{2.4}$$

where $\langle\cdot,\cdot\rangle_2$ is the scalar product on ℓ^2. Using (2.4), we find that

$$(A^*_{s^{-1},t}\, y)(n) = \sum_{m\in\mathbb{Z}} y(m)\, t(s^{-1}m - n). \tag{2.5}$$

However, in order to be consistent with the definition in (2.2) when $s = 1$, we do not define the fractionally-strided convolution as the adjoint given by (2.5), but as in (2.1).

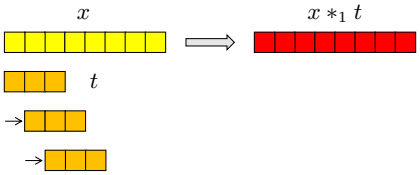
Remark 1. The fractionally-strided convolution defined in (2.1) can be equivalently rewritten as:

$$(x *_s t)(n) = (x * t_r)(k), \tag{2.6}$$

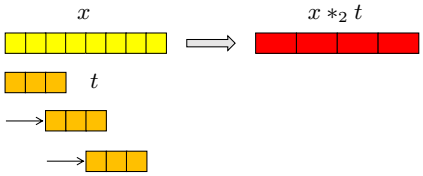
where $n = s^{-1}k + r$ with $r \in \{0, ..., s^{-1} - 1\}$, $k \in \mathbb{Z}$ and $t_r = t(s^{-1}\cdot + r)$ for every $r = 0, ..., s^{-1} - 1$ and the symbol $*$ denotes the discrete convolution defined in (2.3).

Equation (2.6) is useful to interpret Hypothesis 2 on the convolutional filters (Section 2.2). We observe that in our case the convolution is well defined since the signals we consider have a finite number of non-zero entries. However, in general, it is enough to require that $x \in \ell^p$ and $t \in \ell^q$ with $\frac{1}{p} + \frac{1}{q} = 1$ to obtain a well-defined discrete convolution.

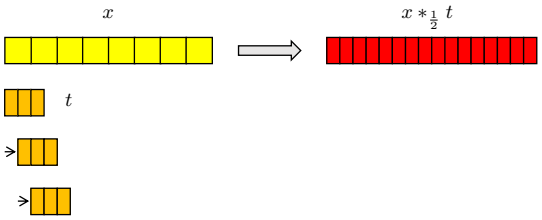
We now present a graphical illustration of three examples of strided convolutions with different strides: $s = 1$ in Figure 2.2a, $s = 2$ in Figure 2.2b, and $s = \frac{1}{2}$ in Figure 2.2c. The input vector x and the filter t have a finite number of non-zero entries indicated with yellow and orange squares/rectangles, respectively, and the output $x *_s t$ has a finite number of non-zero entries indicated with red squares/rectangles. For simplicity, we identify the infinite vectors in $\mathbb{R}^{\mathbb{Z}}$ with vectors in $\mathbb{R}^N$ where N is the number of their non-zero entries. Given the illustrative purpose of these examples, for simplicity we ignore boundary effects. The signals' sizes are:



(a) Discrete convolution with stride $s = 1$ between an input vector $x \in \mathbb{R}^8$ (in yellow) and a filter $t \in \mathbb{R}^3$ (in orange), which gives as output a vector $x *_1 t \in \mathbb{R}^8$ (in red). The input and output sizes are the same.



(b) Discrete convolution with stride $s = 2$ between an input vector $x \in \mathbb{R}^8$ (in yellow) and a filter $t \in \mathbb{R}^3$ (in orange), which gives as output a vector $x *_2 t \in \mathbb{R}^4$ (in red). The output size is half the input size.



(c) Intuition of the convolution with stride $s = \frac{1}{2}$ between an input vector $x \in \mathbb{R}^8$ (in yellow) and a filter $t \in \mathbb{R}^3$ (in orange), which gives as output a vector $x *_{\frac{1}{2}} t \in \mathbb{R}^{16}$ (in red). The output size is twice the input size.

Figure 2.2: Discrete convolutions with strides $s = 1$ and $s = 2$ and intuitive interpretation of convolution with stride $s = \frac{1}{2}$.

(a) $s = 1$, input vector $x \in \mathbb{R}^8$, output vector $x *_1 t \in \mathbb{R}^8$;

(b) $s = 2$, input vector $x \in \mathbb{R}^8$, output vector $x *_2 t \in \mathbb{R}^4$;

(c) $s = \frac{1}{2}$, input vector $x \in \mathbb{R}^8$, output vector $x *_{\frac{1}{2}} t \in \mathbb{R}^{16}$.

When the stride is an integer, equation (2.2) describes what is represented in Figures 2.2a and 2.2b. When the stride is $s = \frac{1}{2}$, as depicted in Figure 2.2c, it is intuitive to consider a filter whose entries are half the size of the input ones. This is equivalent to choosing the filter in a space of higher resolution with respect to the space of the input signal. As a result, the output belongs to the same higher resolution space. For instance, the filter belongs to a space that is twice the resolution of the input space when the stride is $\frac{1}{2}$. This notion of resolution is coherent with the scale parameter used in the continuous setting.

In fact, Figure 2.2c does not represent equation (2.1) exactly when $s = \frac{1}{2}$. However the illustration is useful to model the fractional-strided convolution in the continuous setting. A more precise illustration of the $\frac{1}{2}$-strided convolution of equation (2.2) is presented in Figure 2.3.

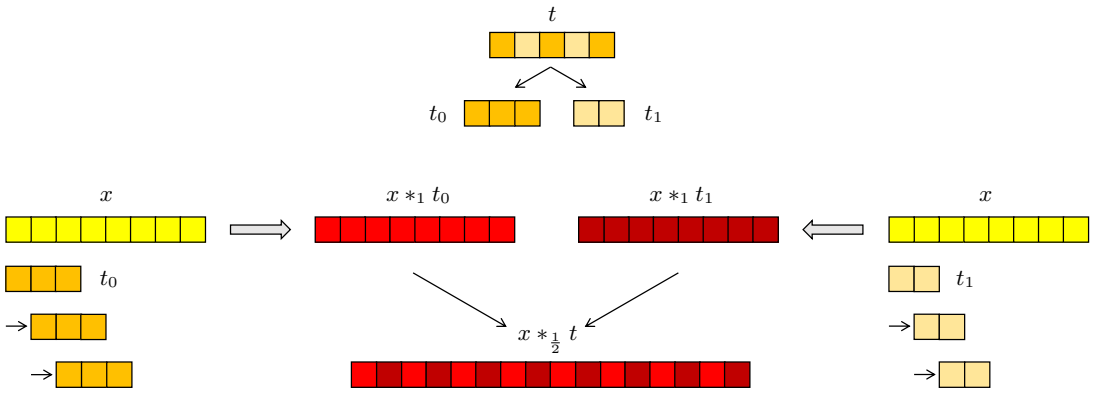


Figure 2.3: Discrete convolution with stride $s = \frac{1}{2}$ between an input vector $x \in \mathbb{R}^8$ (in yellow) and a filter $t \in \mathbb{R}^3$ (in orange), which gives as output a vector $x *_{\frac{1}{2}} t \in \mathbb{R}^{16}$ (in red). The filter t is first divided into two *sub-filters*, t_0 and t_1, containing the odd and even entries, respectively. Then, a convolution with stride 1 between the input vector and each *sub-filter* is performed. Finally, the two output vectors are reassembled to form the final output vector.

2.1.2 1D CGNN architecture

We now describe how to reformulate this discrete architecture in the continuous setting, namely, by considering signals in

$$L^2(\mathbb{R}) := \{f \colon \mathbb{R} \to \mathbb{R} \text{ Lebesgue measurable s.t. } \int_{\mathbb{R}} f^2(x)dx < +\infty\} \,/\sim$$

where $f \sim g$ if and only if $f - g = 0$ a.e. The resolution of these continuous signals is modelled through wavelet analysis. Indeed, the higher the resolution of a signal, the finer the scale of the space to which the signal belongs. The idea to link multi-resolution analysis to neural networks is partially motivated by scattering networks [47].

Basic notions of wavelet theory. We give a brief review of concepts from wavelet analysis: in particular the definitions and the meaning of scaling function spaces and Multi-Resolution analysis in the 1D case. See [69, 114, 166] for more details.

Given a function $\phi \in L^2(\mathbb{R})$, we define

$$\phi_{j,n}(x) = 2^{\frac{j}{2}}\phi(2^j x - n), \qquad x \in \mathbb{R}, \tag{2.7}$$

for every $j, n \in \mathbb{Z}$. The integers j and n are the scale and the translation parameters, respectively, where the scale is proportional to the speed of the oscillations of ϕ (the larger j, the finer the scale, the faster the oscillations).

Definition 1. A *Multi-Resolution Analysis (MRA)* is an increasing sequence of subspaces $\{V_j\} \subset L^2(\mathbb{R})$ defined for $j \in \mathbb{Z}$

$$... \subset V_{-1} \subset V_0 \subset V_1 \subset ...$$

together with a function $\phi \in L^2(\mathbb{R})$ such that

1. $\bigcup_{j\in\mathbb{Z}} V_j$ is dense in $L^2(\mathbb{R})$ and $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$;
2. $f \in V_j$ if and only if $f(2^{-j}\cdot) \in V_0$;
3. and $\{\phi_{0,n}\}_{n\in\mathbb{Z}} = \{\phi(\cdot - n)\}_{n\in\mathbb{Z}}$ is an orthonormal basis of V_0.

The function ϕ is called *scaling function* of the MRA.

Intuitively, the space V_j contains functions for which the finest scale is j.

The spaces. In the discrete formulation, the intermediate spaces $\mathbb{R}^{\alpha_1}, \dots, \mathbb{R}^{\alpha_L}$, with $\alpha_1 < \cdots < \alpha_L$, describe vectors of increasing resolution. In the continuous setting, it is natural to replace these spaces by using a MRA of $L^2(\mathbb{R})$, namely, by using the spaces

$$V_{j_1} \subset V_{j_2} \subset \cdots \subset V_{j_L},$$

with $j_1 < \cdots < j_L$, representing an increasing (finite) sequence of scales. We have that $f \in V_j$ if and only if $f(2\,\cdot) \in V_{j+1}$, so that V_{j+1} contains signals at a resolution that is twice that of the signals in V_j. Thus, the relation between the indexes α_l and j_l is

$$\alpha_l = 2^\nu \alpha_{l-1} \iff j_l = \nu + j_{l-1},$$

where $\nu \in \mathbb{N}$ is a free parameter, or, equivalently,

$$j_l - j_{l-1} = \log_2 \frac{\alpha_l}{\alpha_{l-1}} = \log_2(s^{-1}). \tag{2.8}$$

Similarly to the discrete case, the intermediate spaces are $(V_{j_l})^{c_l}$ for $l = 1, ..., L$, with $c_1 > \cdots > c_L$. The norm in these spaces is

$$\|f\|_2^2 = \sum_{i=1}^{c_l} \|f_i\|^2_{L^2(\mathbb{R})} = \sum_{i=1}^{c_l} \int_{\mathbb{R}} |f_i(x)|^2 dx, \qquad f \in (V_{j_l})^{c_l}.$$

The nonlinearities. The nonlinearities σ_l act on functions in $(L^2(\mathbb{R}))^{c_l}$ by pointwise evaluation:

$$\sigma_l(f)(x) = \sigma_l(f(x)), \qquad \text{a.e. } x \in \mathbb{R}.$$

Note that this map is well defined if there exists L_l such that $|\sigma_l(x)| \leq L_l|x|$ for every $x \in \mathbb{R}$. Indeed, in this case, $\sigma_l(f) \in L^2(\mathbb{R})$ for $f \in L^2(\mathbb{R})$. Moreover, if $f = g$ a.e., then $\sigma_l(f) = \sigma_l(g)$ a.e. It is worth observing that, in general, this nonlinearity does not preserve the spaces $(V_{j_l})^{c_l}$, namely, $\sigma_l((V_{j_l})^{c_l}) \not\subset (V_{j_l})^{c_l}$. However, in the case when the MRA is associated to the Haar wavelet, the spaces $(V_{j_l})^{c_l}$ consist of dyadic step functions, and so they are preserved by the action of σ_l.

The fully connected layer. The map in the first layer is given by

$$\Psi_1 = F \cdot + b, \tag{2.9}$$

where $F\colon \mathbb{R}^S \to (V_{j_1})^{c_1}$ is a linear map and $b \in (V_{j_1})^{c_1}$.

The convolutional layers. We first need to model the convolution in the continuous setting. A convolution with stride $s = 2^\nu$ that maps functions from the scale $j+\nu$ to the scale j with filter $g \in V_{j+\nu} \cap L^1(\mathbb{R})$ can be seen as the map

$$\cdot *_{j+\nu\to j} g\colon L^2(\mathbb{R}) \to L^2(\mathbb{R}), \qquad f *_{j+\nu\to j} g = P_{V_j}(P_{V_{j+\nu}} f * g),$$

where $*\colon L^2(\mathbb{R}) \times L^1(\mathbb{R}) \to L^2(\mathbb{R})$ denotes the continuous convolution and $P_V\colon L^2(\mathbb{R}) \to L^2(\mathbb{R})$ denotes the orthogonal projection onto the closed subspace $V \subset L^2(\mathbb{R})$. In other words,

$$\cdot *_{j+\nu\to j} g = P_{V_j} \circ (\cdot * g) \circ P_{V_{j+\nu}}. \tag{2.10}$$

The orthogonal projections allow us to fix the desired input and output spaces for the continuous strided convolution, analogously to the discrete case (see Figures 2.2 and 2.3). As a consequence, the corresponding deconvolution (i.e. a convolution with stride $2^{-\nu}$) is given by its adjoint, which can be easily computed since projections are self-adjoint and the adjoint of a convolution with filter g is a convolution with filter $\tilde{g}(x) := g(-x)$. Therefore, by renaming $\tilde{g}$ with g, we obtain:

$$\cdot *_{j\to j+\nu} g = P_{V_{j+\nu}} \circ (\cdot * g) \circ P_{V_j}\colon L^2(\mathbb{R}) \to L^2(\mathbb{R}). \tag{2.11}$$

We are now able to model a convolutional layer. The l-th layer of a CGNN, for $l \geq 2$, is

$$\sigma_l \circ \bar{\Psi}_l\colon (L^2(\mathbb{R}))^{c_{l-1}} \to (L^2(\mathbb{R}))^{c_l},$$

where σ_l is the nonlinearity defined above and $\bar{\Psi}_l$ are the convolutions with stride $2^{-\nu}$. In view of the above discussion, and of the discrete counterpart explained in Section 2.1.1, we define

$$\bar{\Psi}_l = P_{(V_{j_l})^{c_l}} \circ \Psi_l \circ P_{(V_{j_{l-1}})^{c_{l-1}}},$$

where the convolution $\Psi_l\colon (L^2(\mathbb{R}))^{c_{l-1}} \to (L^2(\mathbb{R}))^{c_l}$ is given by

$$(\Psi_l(x))_k := \sum_{i=1}^{c_{l-1}} x_i * t^l_{i,k} + b^l_k, \qquad k = 1, ..., c_l, \tag{2.12}$$

with filters $t^l_{i,k} \in V_{j_l} \cap L^1(\mathbb{R})$ and biases $b^l_k \in V_{j_l}$.

Summing up. Altogether, the full architecture in the continuous setting may be written as

$$\begin{aligned} G\colon \mathbb{R}^S \xrightarrow[\text{f.c.}]{\Psi_1} (V_{j_1})^{c_1} \xrightarrow[\text{nonlin.}]{\sigma_1} (L^2(\mathbb{R}))^{c_1} \xrightarrow[\text{proj.}]{P_{(V_{j_1})^{c_1}}} (V_{j_1})^{c_1} \xrightarrow[\text{conv.}]{\Psi_2} (L^2(\mathbb{R}))^{c_2} \\ \xrightarrow[\text{proj.}]{P_{(V_{j_2})^{c_2}}} (V_{j_2})^{c_2} \xrightarrow[\text{nonlin.}]{\sigma_2} (L^2(\mathbb{R}))^{c_2} \xrightarrow[\text{proj.}]{P_{(V_{j_2})^{c_2}}} (V_{j_2})^{c_2} \xrightarrow[\text{conv.}]{\Psi_3} \cdots \\ \cdots \xrightarrow[\text{conv.}]{\Psi_L} L^2(\mathbb{R}) \xrightarrow[\text{proj.}]{P_{V_{j_L}}} V_{j_L} \xrightarrow[\text{nonlin.}]{\sigma_L} L^2(\mathbb{R}) \xrightarrow[\text{proj.}]{P_{V_{j_L}}} V_{j_L}, \end{aligned}$$

which can be summarized as

$$G = \left(\bigcirc_{l=L}^{2} \tilde{\sigma}_l \circ \tilde{\Psi}_l \right) \circ (\tilde{\sigma}_1 \circ \Psi_1), \tag{2.13}$$

where

$$\tilde{\Psi}_l := P_{(V_{j_l})^{c_l}} \circ \Psi_l \colon (V_{j_{l-1}})^{c_{l-1}} \to (V_{j_l})^{c_l}, \qquad l = 2, ..., L, \tag{2.14}$$

and

$$\tilde{\sigma}_l := P_{(V_{j_l})^{c_l}} \circ \sigma_l \colon (V_{j_l})^{c_l} \to (V_{j_l})^{c_l}, \qquad l = 1, ..., L. \tag{2.15}$$

Remark 2 (On the finite-dimensionality of the network G)**.** Even though the scale j is fixed, the spaces V_j in (2.14) and (2.15) are infinite-dimensional because of the infinite translations in (2.7). However, when restricting to functions with a fixed compact support (as is done in practice, and as we will do below), every layer contains maps between finite-dimensional spaces. On the one hand, this will allow for a relatively simple implementation of the network, similarly to a discrete neural network (see §2.1.3 below). On the other hand, this architecture avoids any further arbitrary (e.g. pixel-based) discretization, yielding better results for continuous signals. This aspect is related to the discretization issue in operator learning, see [30].

Remark 3 (Idea behind the continuous strided convolution)**.** Let us focus on the case with stride $s = 2$ for simplicity. The cases with $s = 2^\nu$ with $\nu \geq 2$ are analogous, while the corresponding deconvolutions ($s = 2^\nu$ with $\nu \leq -1$) are simply obtained by taking the adjoint operator, as the discrete deconvolution is obtained by taking the transpose of the convolution. The discrete convolution with stride $s = 2$ (see equation (2.2)) is obtained by

1. doing a standard discrete convolution (see equation (2.2));
2. and keeping only the even entries of the resulting vector.

As a consequence, the resolution of the output vector is half that of the input vector (ignoring boundary effects).

Our definition of the continuous strided convolution (2.10) generalizes these operations. If we start with an input signal $f \in V_{j+1}$ and a filter $g \in V_{j+1}$, the resulting convolution is $P_{V_j}(f * g)$, namely we

(i) take a continuous convolution $f * g$;

(ii) and project the resulting signal onto V_j.

Here, (i) is the natural continuous version of 1. In (ii), the projection onto V_j consists of local averages, which correspond to step 2., where, instead of taking averages, only every second entry of the output vector was kept. Further, the input vector belongs to V_{j+1} and the output vector to V_j, and so the resolution of the latter is half that of the former, as in the discrete case. Finally, in order to define the convolution on the whole space $L^2(\mathbb{R})$, the input vector is first projected onto V_{j+1}.

Remark 4 (Relation between discrete and continuous strided convolutions)**.** If we apply the continuous strided convolution (2.11) to a function $f = \sum_{n\in\mathbb{Z}} c_n \phi_{j,n} \in V_j$, we obtain

$$f *_{j\to j+\nu} g = \sum_{m\in\mathbb{Z}} \langle f * g, \phi_{j+\nu,m}\rangle_2 \phi_{j+\nu,m},$$

where $g = \sum_{p\in\mathbb{Z}} d_p \phi_{j+\nu,p} \in V_{j+\nu} \cap L^1(\mathbb{R})$ represents a filter. It can be shown that

$$\langle f * g, \phi_{j+\nu,m}\rangle_2 = 2^{-\frac{j}{2}} \sum_{n\in\mathbb{Z}} c_n d_\eta (m - 2^\nu n) = 2^{-\frac{j}{2}} (c *_{\frac{1}{2^\nu}} d_\eta)(m), \tag{2.16}$$

where $*_s$ is the discrete strided convolution with stride s defined in (2.1), $d_\eta := d *_1 \eta_\nu$ and η_ν is the sequence defined as

$$\eta_\nu(r) := \int_{\mathbb{R}^2} \phi(t)\phi(z)\phi(z - 2^\nu t + r)dzdt, \qquad r \in \mathbb{Z}. \tag{2.17}$$

Therefore, the coefficients of the output signal (with respect to $\{\phi_{j+\nu,m}\}$) are obtained by taking a discrete strided convolution of the coefficients c of the input signal (with respect to $\{\phi_{j,m}\}$) with the (discrete) filter d_η, which is obtained by taking a discrete convolution of the coefficients of the filter g with η_ν. In other words, a continuous strided convolution between two signals can be seen as a discrete strided convolution between the corresponding coefficients in the natural basis.

For proving (2.16), we first prove that $\langle \phi_{j,n} * \phi_{j+\nu,p}, \phi_{j+\nu,m}\rangle_2$ depends only on the scaling function ϕ, the scale j and the coefficient $m - p - 2^\nu n$. Indeed, we have

$$\begin{aligned}
\langle \phi_{j,n} * \phi_{j+\nu,p}, \phi_{j+\nu,m}\rangle_2 &= \int_{\mathbb{R}^2} \phi_{j,n}(y)\phi_{j+\nu,p}(x - y)\phi_{j+\nu,m}(x)dxdy \\
&= \int_{\mathbb{R}^2} 2^{\frac{j}{2}} \phi\Big(2^j y - n\Big) 2^{\frac{j+\nu}{2}} \phi(2^{j+\nu}(x - y) - p) 2^{\frac{j+\nu}{2}} \phi(2^{j+\nu}x - m)dxdy \\
&= 2^{-\frac{j}{2}} \int_{\mathbb{R}^2} \phi(t)\phi(z)\phi(z - 2^\nu t + m - 2^\nu n - p)dzdt \\
&= 2^{-\frac{j}{2}} \eta_\nu(m - p - 2^\nu n),
\end{aligned} \tag{2.18}$$

where $\phi_{j,n} = 2^{\frac{j}{2}}\phi(2^j \cdot -n)$ and $\eta_\nu(r) = \int_{\mathbb{R}^2} \phi(t)\phi(z)\phi(z - 2^\nu t + r)dzdt$, as defined in (2.7) and (2.17), respectively. Then we obtain

$$\begin{aligned}
\langle f * g, \phi_{j+\nu,m}\rangle_2 &= \sum_{n\in\mathbb{Z}}\sum_{p\in\mathbb{Z}} c_n d_p \langle \phi_{j,n} * \phi_{j+\nu,p}, \phi_{j+\nu,m}\rangle_2 \\
&= 2^{-\frac{j}{2}} \sum_{n\in\mathbb{Z}} c_n \sum_{p\in\mathbb{Z}} d_p \eta(m - p - 2^\nu n) \\
&= 2^{-\frac{j}{2}} \sum_{n\in\mathbb{Z}} c_n d_\eta(m - 2^\nu n) \\
&= 2^{-\frac{j}{2}} (c *_{\frac{1}{2^\nu}} d_\eta)(m),
\end{aligned}$$

where $*_s$ is the discrete strided convolution with stride s defined in (2.1), $d_\eta := d *_1 \eta_\nu$ and $\eta_\nu(r)$ is the sequence defined in (2.17).

A simple example: the Haar case. Let V_j be the spaces of the MRA associated to the Haar scaling function $\phi = \mathbb{1}_{[0,1]}$. This simple setting naturally extends the discrete case to the continuous one. Indeed, given a signal $f \in V_j$, namely a piecewise constant function on dyadic intervals, its coefficients with respect to the family $\{\phi_{j,m}\}$ are the values of f itself (up to a normalization factor), and so discrete and continuous convolutions almost coincide (see (2.16)). The only difference being in the filter, since in this case $\eta_1 = [..., 0, 0.25, 0.5, 0.25, 0, ...] \in \mathbb{R}^{\mathbb{Z}}$, and so $d \neq d_\eta$. We would have exact correspondence if η were the Dirac delta, i.e. $\eta = [..., 0, 0, 1, 0, 0, ...] \in \mathbb{R}^{\mathbb{Z}}$ (but this cannot be obtained with any choice of wavelet), so that $d_\eta = d$. Moreover, the Haar scaling function makes it possible to simplify the structure of a CGNN. Indeed $\sigma_l(V_{j_l}) \subset V_{j_l}$, thanks to the form of ϕ and the fact that ϕ_{j_l,k_1} and ϕ_{j_l,k_2} have disjoint support for every $k_1 \neq k_2$. Therefore, in this setting, the projections after the nonlinearities can be removed.

2.1.3 Details on the implementation of the network

In order to implement our generator (2.13), which is written as a composition of maps between infinite-dimensional spaces, $\tilde{\Psi}_l$ and $\tilde{\sigma}_l$, we need to find a proper discretization of the network, ideally avoiding fine discretizations of the space.

Implementation of $\tilde{\Psi}_l$. The continuous strided convolution $\tilde{\Psi}_l$, namely a continuous convolution followed by the projection, may be efficiently computed by using (2.16), if we set the bias term to be zero for each output channel. Indeed, we require only one computation of a continuous integral (i.e. the integral that defines η_ν in (2.17), which depends only on the choice of the wavelet) and a series of discrete convolutions. In the case when the bias term is not trivial, this can be dealt with directly at the level of the wavelet coefficients.

Implementation of $\tilde{\sigma}_l$. The computation of $\tilde{\sigma}_l$, i.e. the nonlinearity followed by the projection, is more subtle (apart in the case of the Haar wavelet, where it is enough to apply

the nonlinearity to the scaling coefficients). The most straightforward implementation would be to consider a fine discretization of the space, which would allow for the computation of the pointwise nonlinearity and of the integrals involved in the scalar products related to the projection. However, this would be computationally heavy. Instead, we propose to approximate the pointwise values of a signal belonging to a certain V_j by using its scaling coefficients in V_{j+M} with M sufficiently large. We apply the nonlinearity directly to these coefficients, and then we project back to V_j. Note that going from V_j to V_{j+M} and back can be performed very efficiently by using the fast wavelet transform, and the operations of upsampling and downsampling (see [69, Section 5.6]).

We now clarify how the scaling coefficients in V_{j+M} for large M provide a (pointwise) discretization of the original signal in V_j. We have

$$\lim_{j\to+\infty} \frac{2^{\frac{j}{2}}\langle f, \phi_{j,2^j b}\rangle_2}{\int_{\mathbb{R}} \phi \, dx} = f(b), \qquad \text{a.e. } b \in \mathbb{R}, \tag{2.19}$$

whenever $f \in L^2(\mathbb{R})$ and ϕ is compactly supported, bounded and $\int_{\mathbb{R}} \phi \, dx \neq 0$. If f is continuous, equation (2.19) holds for every $b \in \mathbb{R}$. In other words, the scaling coefficients at fine scales approximate the pointwise values of the function, up to a constant. In the Haar case, equation (2.19) is a simple consequence of the Lebesgue differentiation theorem, while in the general case, it follows from an extension (see [96, Corollary 2.1.19]).

2.2 Injectivity of CGNNs

We are interested in studying the injectivity of the continuous generator (2.13) to guarantee uniqueness in the representation of the signals. The injectivity will also allow us, as a by-product, to obtain stability results for inverse problems using generative models, as in Section 2.3.

We consider here the 1D case with stride $s = \frac{1}{2}$ ($\nu = 1$); then, applying (2.8) iteratively, we obtain $j_l = j_1 + l - 1$ for $l = 1, \dots, L$. We also consider non-expansive convolutional layers, i.e. $c_l = \frac{c_{l-1}}{2} = sc_{l-1} = \frac{c_1}{2^{l-1}}$. We note that the same result holds also with expansive convolutional layers, arbitrary stride (possibly dependent on l) and in the 2D case (see Section 2.2.2).

We make the following assumptions.

Assumptions on the scaling spaces V_{j_l}.

Hypothesis 1. The spaces $V_{j_l} \subset L^2(\mathbb{R})$, with $j_l \in \mathbb{N}$, belong to an MRA (see Definition 1), whose scaling function ϕ is compactly supported and bounded. Furthermore, there exists $r \in \mathbb{Z}$ such that

$$\eta_1(r) \neq 0, \tag{2.20}$$

where $\eta_1(r)$ is defined in (2.17).

We observe that Hypothesis 1 implies that the sequence η_ν defined in (2.17) has a finite number of non-zero elements. As a consequence, when g is compactly supported, the filters d_η can be represented by finite-dimensional vectors, as for discrete neural networks. However, as highlighted in Remark 4, the continuous strided convolution involves a double convolution (2.16), which is not present in the discrete strided convolution.

Remark 5 (Haar and Daubechies scaling functions)**.** For positive functions, such as the Haar scaling function, i.e. $\phi = \mathbb{1}_{[0,1]}$, condition (2.20) is easily satisfied. For the Daubechies scaling functions with N vanishing moments for $N = 1, 2, ..., 45$ ($N = 1$ corresponds to the Haar scaling function), we verified condition (2.20) numerically. We believe that this condition is satisfied for every scaling function ϕ, but have not been able to prove this rigorously.

We note, however, that Hypothesis 1 is clearly needed for the injectivity of the convolutional layers. Indeed if we had $\eta_1(r) = 0$ for every $r \in \mathbb{Z}$, then

$$\langle \phi_{j,n} * \phi_{j+1,p}\,,\, \phi_{j+1,m} \rangle_2 = 0, \qquad j, n, p, m \in \mathbb{Z}.$$

Therefore, Hypothesis 1 guarantees that the continuous deconvolution $\cdot *_{j \to j+1} g$ is not identically 0 (at least for some filter $g \in V_{j+1}$).

Assumptions on the convolutional filters. The following hypothesis asks that, at each convolutional layer, the filters are compactly supported, where $\bar{p} + 1$ represents the filters' size. Generally, the convolutional filters act locally, so it is natural to assume that they have compact support. Furthermore, we ask the filters to be linearly independent, in a suitable sense: this is needed for the injectivity of the convolutional layers.

Hypothesis 2. Let $\bar{p} \in \mathbb{N}$. For every $l = 2, ..., L$, the convolutional filters $t^l_{i,k} \in V_{j_l}$ of the l-th convolutional layer (2.12) satisfy

$$t^l_{i,k} = \sum_{p=0}^{\bar{p}} d^l_{p,i,k} \phi_{j_l,p}, \qquad i = 1, ..., c_{l-1}, \qquad k = 1, ..., c_l,$$

where $d^l_{p,i,k} \in \mathbb{R}$, and $\det(D^l) \neq 0$, where D^l is the $\frac{c_1}{2^{l-1}} \times \frac{c_1}{2^{l-1}}$ matrix defined by

$$(D^l)_{i,k} := \begin{cases} d^l_{0,i,k} & k = 1, ..., \frac{c_1}{2^l}, \\ d^l_{1,i,k-\frac{c_1}{2^l}} & k = \frac{c_1}{2^l} + 1, ..., \frac{c_1}{2^{l-1}}. \end{cases} \tag{2.21}$$

Remark 6. The condition $\det(D^l) \neq 0$ is sufficient for the injectivity of the convolutional layers, but not necessary. The necessary condition is given in 2.2.1, and consists in requiring the rank of a certain block matrix to be maximum, in which D^l is simply the first block. We note that this condition is independent of the scaling function ϕ, but depends only on the filters' coefficients $d^l_{p,i,k}$.

Remark 7 (Analogy between continuous and discrete case)**.** The splitting operation of the filters' scaling coefficients in odd and even entries, as in Hypothesis 2, reminds the expression of the discrete convolution with stride $s = \frac{1}{2}$. Indeed, a discrete filter t is split into t_0, containing the even entries of t, and t_1, containing the odd ones (2.6).

Assumptions on the nonlinearity. For simplicity, we consider the same nonlinearity $\sigma\colon \mathbb{R} \to \mathbb{R}$ in each layer (the generalization to the general case is straightforward). The following conditions guarantee that $\tilde{\sigma}_l$ is injective.

Hypothesis 3. We assume that

1. $\sigma \in C^1(\mathbb{R})$ is injective and $M_1 \leq \sigma'(x) \leq M_2$ for every $x \in \mathbb{R}$, for some $M_1, M_2 > 0$;

2. $\sigma(0) = 0$ and σ preserves the sign, i.e. $x \cdot \sigma(x) \geq 0$ for every $x \in \mathbb{R}$;

Note that these conditions ensure that $|\sigma(x)| \leq L|x|$ for every $x \in \mathbb{R}$, for some $L > 0$, and so $\sigma_l(f) \in (L^2(\mathbb{R}))^{c_l}$ for every $f \in (L^2(\mathbb{R}))^{c_l}$. It is also straightforward to check that the injectivity of $\sigma\colon \mathbb{R} \to \mathbb{R}$ ensures the injectivity of

$$\sigma_l\colon (V_{j_l})^{c_l} \to (L^2(\mathbb{R}))^{c_l}, \qquad f \mapsto \sigma_l(f).$$

Remark 8. In the Haar case, the projection after the nonlinearity can be removed, as explained at the end of Section 2.1.2, and we need to verify only the injectivity of σ_l instead of that of $\tilde{\sigma}_l = P_{(V_{j_l})^{c_l}} \circ \sigma_l$. As noted above, a sufficient condition to guarantee the injectivity of σ_l is the injectivity of σ. So, in the Haar case, Hypothesis 3 can be relaxed and replaced by:

1. σ is injective;

2. There exists $L > 0$ such that $|\sigma(x)| \leq L|x|$ for every $x \in \mathbb{R}$.

Hypothesis 3 is satisfied for example by the function $\sigma_{\text{hp}}(x) = |x| \arctan(x)$. Its relaxed version, in the Haar case, is satisfied by some commonly used nonlinearities, such the Sigmoid, the Hyperbolic tangent, the Softplus, the Exponential linear unit (ELU) and the Leaky rectified linear unit (Leaky ReLU). Our approach does not allow us to consider non-injective σ's, such as the ReLU [187].

For simplicity, in Hypothesis 3, we require that $\sigma \in C^1(\mathbb{R})$ and that σ' is strictly positive everywhere. This allows us to use Hadamard's global inverse function theorem [94] to obtain the injectivity of the generator. However, thanks to a generalized version of Hadamard's theorem [186], we expect to be able to relax the conditions by requiring only that σ is Lipschitz and its generalized derivative is strictly positive everywhere. In this way, the Leaky ReLU would satisfy the assumptions.

Assumptions on the fully connected layer. We impose the following natural hypothesis on the fully connected layer.

Hypothesis 4. We assume that

1. The linear function $F\colon \mathbb{R}^S \to (V_{j_1})^{c_1}$ is injective;

2. There exists $N \in \mathbb{N}$ such that $b \in (\text{span}\{\phi_{j_1,n}\}_{n=-N}^{N})^{c_1}$ and $\text{Im}(F) \subset (\text{span}\{\phi_{j_1,n}\}_{n=-N}^{N})^{c_1}$.

The inclusion $F(\mathbb{R}^S)+b \subset (V_{j_1})^{c_1}$ is natural, since we start with low-resolution signals. The second condition in Hypothesis 4 means that the image of the first layer, $F(\mathbb{R}^S)+b$, contains only compactly supported functions. This is natural since we deal with signals of finite size.

Even the injectivity of F is non-restrictive, since we choose the dimension S of the latent space to be much smaller than the dimension of $\mathrm{Im}(F) = (\mathrm{span}\{\phi_{j_1,n}\}_{n=-N}^{N})^{c_1}$, which is $c_1(2N+1)$. So, F maps a low-dimensional space into a higher dimensional one.

The injectivity theorem. The main result of this section reads as follows.

Theorem 1. *Let $L \in \mathbb{N}^*$ and $j_1 \in \mathbb{Z}$. Let $c_1 = 2^{L-1}$, $c_l = \frac{c_1}{2^{l-1}}$ and $j_l = j_1 + l - 1$ for every $l = 1, ..., L$. Let V_{j_l} be the scaling function space arising from an MRA, and $t_{i,k}^l \in V_{j_l}$ for every $l = 2, ..., L$, $i = 1, ..., c_{l-1}$ and $k = 1, ..., c_l$. Let $\tilde{\Psi}_l$ and $\tilde{\sigma}_l$ be defined as in (2.14) and (2.15), respectively. Let Ψ_1 be defined as in (2.9). If Hypotheses 1, 2, 3 and 4 are satisfied, then the generator G defined in (2.13) is injective.*

Sketch of the proof. Consider (2.13), (2.14) and (2.15). Note that Ψ_1 is injective by Hypothesis 4. If we also show that $\tilde{\Psi}_l$ is injective for every $l = 2, \ldots, L$ and that $\tilde{\sigma}_l$ is injective for every $l = 1, \ldots, L$, then the injectivity of G will immediately follow.

The injectivity of $\tilde{\Psi}_l$ is a consequence of Hypothesis 2 (together with Hypothesis 1). The injectivity of $\tilde{\sigma}_l$ follows from Hypothesis 3 (together with Hypotheses 1 and 4) and from Hadamard's global inverse function theorem applied to $\tilde{\sigma}_l$. The full proof is presented in 2.2.1. □

Remark 9 (Simplified CGNN architecture)**.** It is possible to consider a simplified CGNN architecture in which the nonlinearities σ are applied on the signal scaling coefficients, i.e. $\sigma(f) = \sum_{n\in\mathbb{Z}} \sigma(c_n)\phi_{j,n}$ with $f = \sum_{n\in\mathbb{Z}} c_n\phi_{j,n} \in V_j$. In this case, the projections onto the scaling spaces following the nonlinearities are not needed because $\sigma(f) \in V_j$. Therefore, the injectivity of $\tilde{\sigma}_\ell$ is guaranteed by assuming only the injectivity of σ. Then, the injectivity of the simplified CGNN follows from Theorem 1 by replacing Hypothesis 3 with the injectivity of σ.

2.2.1 Proof of Theorem 1

Due to the intricate nature of the proof of Theorem 1, we have allocated a dedicated section to present it. Readers who are not interested in the technical details of the proof may choose to skip this section.

We begin with some preliminary technical lemmas.

Lemma 2. *Let Hypothesis 1 holds. Then*

$$\hat{\eta}_1(\xi) \neq 0 \text{ for a.e. } \xi \in [0,1],$$

where $\hat{\eta}_1(\xi)$ is the Fourier series of $(\eta_1(r))_{r\in\mathbb{Z}}$, defined as

$$\hat{\eta}_1(\xi) := \sum_{r\in\mathbb{Z}} \eta_1(r)e^{-2\pi i\xi r}, \qquad \xi \in [0,1]. \tag{2.22}$$

Proof. By Hypothesis 1, ϕ is compactly supported, thus the series $(\eta_1(r))_{r\in\mathbb{Z}}$ has a finite number of non-zero entries. Then, $\hat{\eta}_1$ is an analytic function, with $\hat{\eta}_1 \not\equiv 0$, since by Hypothesis 1, there exists $r \in \mathbb{Z}$ such that $\eta_1(r) \neq 0$. Therefore $\hat{\eta}_1(\xi) \neq 0$ for a.e. $\xi \in [0,1]$. □

Lemma 3. *If Hypotheses 1, 2 and 4 are satisfied, then the image of each layer of the generator G defined in (2.13) contains only compactly supported functions. More precisely:*

$$\left(\bigcirc_{\tilde{l}=l}^{2} \tilde{\sigma}_{\tilde{l}} \circ \tilde{\Psi}_{\tilde{l}}\right) \circ (\tilde{\sigma}_1 \circ \Psi_1)(\mathbb{R}^S) \subset W_l, \quad l = 2, ..., L, \qquad (\tilde{\sigma}_1 \circ \Psi_1)(\mathbb{R}^S) \subset W_1,$$

$$\tilde{\Psi}_{l+1} \circ \left(\bigcirc_{\tilde{l}=l}^{2} \tilde{\sigma}_{\tilde{l}} \circ \tilde{\Psi}_{\tilde{l}}\right) \circ (\tilde{\sigma}_1 \circ \Psi_1)(\mathbb{R}^S) \subset W_{l+1}, \quad l = 2, ..., L-1, \qquad \tilde{\Psi}_2 \circ (\tilde{\sigma}_1 \circ \Psi_1)(\mathbb{R}^S) \subset W_2,$$

where, for every $l = 1, ..., L$, $W_l := (\mathrm{span}\{\phi_{j_l,n}\}_{n=-N^l}^{N^l})^{c_l}$ for some $N^l \in \mathbb{N}$.

Proof. By Hypothesis 2, the filters of each layer are compactly supported and, by Hypothesis 4, the same happens to the functions in the image of the first (fully connected) layer, $F(\mathbb{R}^S) + b$. Moreover, the nonlinearities do not change the support of the functions since they act pointwisely, and each V_j is spanned by the translates of a fixed compactly supported function. Therefore, the image of each layer contains only compactly supported functions. □

In the rest of this section, with an abuse of notation, we indicate with σ both the scalar function $\sigma\colon \mathbb{R} \to \mathbb{R}$ and the operator $\sigma\colon V_j^c \to (L^2(\mathbb{R}))^c$.

Lemma 4. *Let $\sigma\colon \mathbb{R} \to \mathbb{R}$ satisfy Hypothesis 3. Let W be a finite-dimensional subspace of $(L^2(\mathbb{R}))^c$ of the form $W_1 \times ... \times W_c$, where W_i are finite-dimensional subspaces of $L^2(\mathbb{R}) \cap L^\infty(\mathbb{R})$ for every $i = 1, ..., c$. Then $\sigma\colon W \subset L^2(\mathbb{R})^c \to L^2(\mathbb{R})^c$ is Fréchet differentiable and its Fréchet derivative in $g \in W$ is*

$$\begin{aligned} \sigma'(g)\colon W &\to (L^2(\mathbb{R}))^c \\ f &\mapsto \sigma'(g)[f] \end{aligned}$$

where

$$\begin{aligned} (\sigma'(g)[f])_i\colon \mathbb{R} &\to \mathbb{R} \\ x &\mapsto \sigma'(g_i(x)) f_i(x), \end{aligned}$$

for every $i = 1, ..., c$. Moreover, $\sigma \in C^1(W)$.

Proof. Without loss of generality, set $c = 1$. Let $g \in W$. First, we observe that $\sigma'(g)$, as defined above, is linear. It is also bounded i.e. there exists $C \in \mathbb{R}^+$ such that $\|\sigma'(g)[f]\|_2^2 \leq C\|f\|_2^2$. Indeed, by Hypothesis 3 we have

$$\|\sigma'(g)[f]\|_2^2 := \int_{\mathbb{R}} (\sigma'(g(x))f(x))^2 dx \leq M_2^2 \int_{\mathbb{R}} f(x)^2 dx = M_2^2 \|f\|_2^2.$$

Next, we show that $\sigma'(g)$ is the Fréchet derivative of σ, namely

$$\lim_{t\to 0} \sup_{\|f\|_2=1, f\in W} \frac{\|\sigma(g+tf) - \sigma(g) - t\,\sigma'(g)[f]\|_2}{t} = 0,$$

which is equivalent to

$$\lim_{t\to 0} \sup_{\|f\|_2=1, f\in W} \frac{\|\sigma(g+tf) - \sigma(g) - t\,\sigma'(g)[f]\|_2^2}{t^2} = 0. \tag{2.23}$$

Indeed, fixing $x \in \mathbb{R}$, since $\sigma \in C^1(\mathbb{R})$, the mean value theorem yields

$$\sigma(g(x) + tf(x)) - \sigma(g(x)) = tf(x)\sigma'(g(x) + \tau f(x)),$$

where $0 \le \tau \le t \le 1$. Then, (2.23) becomes

$$\lim_{t\to 0} \sup_{\|f\|_2=1, f\in W} \int_{\mathbb{R}} f(x)^2(\sigma'(g(x)) - \sigma'(g(x) + \tau f(x)))^2 dx.$$

We observe that the space where σ' is evaluated is

$$\{g(x) + \tau f(x) : x \in \mathbb{R}, 0 \le \tau \le 1, f \in W \text{ s.t. } \|f\|_2 = 1\},$$

which is contained in $K = [-(\|g\|_\infty + \tilde{C}), \|g\|_\infty + \tilde{C}]$ where $\|g\|_\infty$ is finite since $g \in W \subset L^\infty(\mathbb{R})$, and $\tilde{C} = \sup_{f\in W, \|f\|_2=1} \|f\|_\infty$, which is finite since W is a finite-dimensional subspace of $L^2(\mathbb{R}) \cap L^\infty(\mathbb{R})$, therefore $\|\cdot\|_2$ is equivalent to $\|\cdot\|_\infty$ in W.

Moreover, σ' is uniformly continuous in K, because $\sigma \in C^1(\mathbb{R})$ by Hypothesis 3. Therefore, there exists a modulus of continuity $\omega\colon \mathbb{R}^+ \to \mathbb{R}$ such that

$$|\sigma'(x) - \sigma'(y)| \le \omega(|x-y|), \qquad x, y \in K,$$

and

$$\lim_{t\to 0^+} \omega(t) = \omega(0) = 0. \tag{2.24}$$

We notice also that we can choose ω as an increasing function. Therefore, we obtain

$$\begin{aligned}
&\lim_{t\to 0} \sup_{\|f\|_2=1, f\in W} \int_{\mathbb{R}} f(x)^2(\sigma'(g(x)) - \sigma'(g(x) + \tau f(x)))^2 dx \\
&\le \lim_{t\to 0} \sup_{\|f\|_2=1, f\in W} \int_{\mathbb{R}} f(x)^2 \omega(|\tau f(x)|)^2 dx \\
&\le \lim_{t\to 0} \sup_{\|f\|_2=1, f\in W} \int_{\mathbb{R}} f(x)^2 \omega(t\|f\|_\infty)^2 dx \\
&\le \lim_{t\to 0} \omega(t\tilde{C})^2 \\
&= 0.
\end{aligned}$$

Finally, we show that $\sigma \in C^1(W)$. The continuity of σ' in g is verified if

$$\lim_{\|f\|_2 \to 0, f \in W} \|\sigma'(g+f) - \sigma'(g)\|^2_{\mathcal{L}(W, L^2(\mathbb{R}))} = 0.$$

We have

$$\begin{aligned}
&\|\sigma'(g+f) - \sigma'(g)\|^2_{\mathcal{L}(W, L^2(\mathbb{R}))} \\
&:= \sup_{\|h\|_2 = 1, h \in W} \|\sigma'(g+f)[h] - \sigma'(g)[h]\|_2^2 \\
&= \sup_{\|h\|_2 = 1, h \in W} \int_{\mathbb{R}} h(x)^2 (\sigma'(g(x) + f(x)) - \sigma'(f(x)))^2 dx.
\end{aligned}$$

We consider $\|f\|_2 \leq 1$, since $\|f\|_2 \to 0$. Using an argument similar to the one above, we observe that the space where σ' is evaluated is compact in $\mathbb{R}$. Thus, there exists an increasing modulus of continuity, as explained before. Therefore, we obtain

$$\begin{aligned}
&\sup_{\|h\|_2 = 1, h \in W} \int_{\mathbb{R}} h(x)^2 (\sigma'(g(x) + f(x)) - \sigma'(f(x)))^2 dx \\
&\leq \sup_{\|h\|_2 = 1, h \in W} \int_{\mathbb{R}} h(x)^2 \omega(|f(x)|)^2 dx \\
&\leq \omega(\|f\|_\infty)^2,
\end{aligned}$$

and $\omega(\|f\|_\infty)^2 \to 0$ as $\|f\|_2 \to 0$ by (2.24), since $\|\cdot\|_2$ and $\|\cdot\|_\infty$ are equivalent norms in W. □

We recall a classical result that will be used in the proof: Hadamard's global inverse function theorem.

Theorem 5 ([94]). *A C^1 map $f\colon \mathbb{R}^N \to \mathbb{R}^N$ is a diffeomorphism if and only if the Jacobian never vanishes, i.e. $\det(J_f(x)) \neq 0$ for every $x \in \mathbb{R}^N$, and $|f(x)| \to +\infty$ as $|x| \to +\infty$.*

We are now able to prove the main theorem of this chapter.

Proof of Theorem 1. Let us recall the network G in (2.13):

$$G = \left(\bigcirc_{l=L}^{2} \tilde{\sigma}_l \circ \tilde{\Psi}_l \right) \circ (\tilde{\sigma}_1 \circ \Psi_1),$$

where

$$\begin{aligned}
\tilde{\Psi}_l &:= P_{(V_{j_l})^{c_l}} \circ \Psi_l \colon (V_{j_{l-1}})^{c_{l-1}} \to (V_{j_l})^{c_l}, \qquad l = 2, ..., L, \\
\tilde{\sigma}_l &:= P_{(V_{j_l})^{c_l}} \circ \sigma_l \colon (V_{j_l})^{c_l} \to (V_{j_l})^{c_l}, \qquad l = 1, ..., L.
\end{aligned}$$

Note that $\Psi_1 = F \cdot + b$ is injective by Hypothesis 4. We will show that the restriction of $\tilde{\Psi}_l$ to the image of the previous layer is injective for every $l = 2, \ldots, L$ and that the restriction of $\tilde{\sigma}_l$ to the image of the previous layer is injective for every $l = 1, \ldots, L$. The injectivity of G will immediately follow.

Let us fix a layer l. The proof holds for every $l = 2, ..., L$ for $\tilde{\Psi}_l$ and for every $l = 1, ..., L$ for $\tilde{\sigma}_l$. To simplify the notation, we omit the dependence of the quantities on l and we denote the number of input channels by c and the input scale by j.

Step 1: *Injectivity of* $\tilde{\Psi}|_W$. Let $W = W_{l-1}$, where W_{l-1} is defined in Lemma 3. Therefore, there exists $N \in \mathbb{N}$ such that $W = (\text{span}\{\phi_{j,n}\}_{n=-N}^{N})^c$. There are $\frac{c}{2}$ output channels and the output scale is $j+1$. Take $h, g \in W$ such that $\tilde{\Psi}(h) = \tilde{\Psi}(g)$, i.e.

$$P_{V_{j+1}}\Big(\sum_{i=1}^{c} t_{i,k} * h_i + b_k\Big) = P_{V_{j+1}}\Big(\sum_{i=1}^{c} t_{i,k} * g_i + b_k\Big), \quad k = 1, ..., \frac{c}{2}.$$

We need to show that $h = g$. By the linearity of the projections and of the convolutions, we obtain

$$P_{V_{j+1}}\Big(\sum_{i=1}^{c} t_{i,k} * f_i\Big) = 0, \quad k = 1, ..., \frac{c}{2}, \tag{2.25}$$

where $f = h - g$. We need to show that $f_i = 0$ for every $i = 1, ..., c$.

Recall that $\{t_{i,k}\}_{i,k}$ satisfy Hypothesis 2, and $f \in W$, i.e.

$$t_{i,k} = \sum_{p=0}^{\bar{p}} d_{p,i,k}\phi_{j+1,p}, \qquad d_{p,i,k} = \langle t_{i,k}, \phi_{j+1,p}\rangle_2,$$

and

$$f_i = \sum_{n=-N}^{N} v_{n,i}\phi_{j,n}, \qquad v_{n,i} = \langle f_i, \phi_{j,n}\rangle_2.$$

Since $\{\phi_{j+1,m}\}_{m\in\mathbb{Z}}$ is an orthonormal basis of V_{j+1}, we reformulate (2.25) in the following way

$$\sum_{i=1}^{c}\sum_{n=-N}^{N}\Big(\sum_{p=0}^{\bar{p}} d_{p,i,k}\langle F_{n,p}, \phi_{j+1,m}\rangle_2\Big)v_{n,i} = 0, \quad k = 1, ..., \frac{c}{2}, \quad m \in \mathbb{Z}, \tag{2.26}$$

where $F_{n,p} := \phi_{j,n} * \phi_{j+1,p}$. We need to show that $v_{n,i} = 0$ for every $i = 1, ..., c$ and every $n = -N, ..., N$. Setting

$$A_{(m,k),(n,i)} := \sum_{p=0}^{\bar{p}} d_{p,i,k}\langle F_{n,p}, \phi_{j+1,m}\rangle_2, \tag{2.27}$$

equation (2.26) becomes

$$\sum_{i=1}^{c}\sum_{n=-N}^{N} A_{(m,k),(n,i)}v_{n,i} = 0, \quad k = 1, ..., \frac{c}{2}, \quad m \in \mathbb{Z}. \tag{2.28}$$

We observe that $\langle F_{n,p}, \phi_{j+1,m}\rangle_2$ depends only on the scaling function ϕ, the scale j and the coefficient $m - p - 2n$ as show in (2.18), i.e. $\langle F_{n,p}, \phi_{j+1,m}\rangle_2 = 2^{-\frac{j}{2}}\eta_1(m - p - 2n)$ with η_1 defined in (2.17). Since ϕ is compactly supported, η_1 has a finite number of non-zero entries, namely $\eta_1 \in c_{00}(\mathbb{Z})$, and, by Hypothesis 2, the same holds for $d_{\cdot,i,k}$ (suitably extended by 0 outside its support). This allows us to rewrite (2.27) as

$$A_{(m,k),(n,i)} = 2^{-\frac{j}{2}}\sum_{p\in\mathbb{Z}} d_{p,i,k}\eta_1(m - p - 2n) = 2^{-\frac{j}{2}}(d_{\cdot,i,k} * \eta)(m - 2n) = 2^{-\frac{j}{2}}\, d^{\eta}_{i,k}(m - 2n),$$

where $d^{\eta}_{i,k} := (d_{\cdot,i,k} * \eta_1) \in c_{00}(\mathbb{Z})$ and $*$ represents the discrete convolution defined in (2.3).

Then (2.28) is equivalent to

$$\sum_{i=1}^{c} \sum_{n\in\mathbb{Z}} d^{\eta}_{i,k}(m-2n)v_{n,i} = 0, \qquad m \in \mathbb{Z}, \quad k = 1, ..., \frac{c}{2},$$

where $v_{\cdot,i} \in c_{00}(\mathbb{Z})$. Therefore

$$\sum_{i=1}^{c} \sum_{n,m\in\mathbb{Z}} d^{\eta}_{i,k}(m-2n)v_{n,i}e^{-2\pi i\xi m} = 0, \qquad \text{a.e. } \xi \in [0,1]. \tag{2.29}$$

Rewriting (2.29) as

$$\sum_{i=1}^{c} \sum_{n\in\mathbb{Z}} \Big(\sum_{m\in\mathbb{Z}} d^{\eta}_{i,k}(m-2n)e^{-2\pi i\xi(m-2n)} \Big) v_{n,i}e^{-2\pi i\xi 2n} = 0$$

and defining the Fourier series of $(g(r))_{r\in\mathbb{Z}} \in \ell^2(\mathbb{Z})$ as in (2.22), we obtain

$$\sum_{i=1}^{c} \widehat{d^{\eta}_{i,k}}(\xi)\widehat{v_i}(2\xi) = 0, \qquad \text{a.e. } \xi \in [0,1],$$

where $v_i(n) := v_{n,i}$. Applying the Convolution Theorem, we obtain

$$\sum_{i=1}^{c} \hat{\eta}_1(\xi)\widehat{d_{i,k}}(\xi)\widehat{v_i}(2\xi) = 0, \qquad \text{a.e. } \xi \in [0,1], \tag{2.30}$$

where $d_{i,k}(p) := d_{p,i,k}$.

Thanks to Lemma 2, equation (2.30) is equivalent to

$$\sum_{i=1}^{c} \widehat{d_{i,k}}(\xi)\widehat{v_i}(2\xi) = 0, \qquad \text{a.e. } \xi \in [0,1], \qquad k = 1, ..., \frac{c}{2}.$$

We can rewrite this condition on the coefficients, by writing the definition of the Fourier series and using the orthonormality of $\{e^{2\pi i\xi r}\}_{r\in\mathbb{Z}}$, which gives

$$\sum_{i=1}^{c} \sum_{n\in\mathbb{Z}} d_{m-2n,i,k}v_{n,i} = 0, \qquad m \in \mathbb{Z}, \qquad k = 1, ..., \frac{c}{2}. \tag{2.31}$$

Considering the odd and the even entries of $d_{\cdot,i,k}$ separately, we can split (2.31) in this way

$$\sum_{i=1}^{c} \sum_{n\in\mathbb{Z}} \tilde{d}_{m-n,i,k}v_{n,i} = 0, \qquad m \in \mathbb{Z}, \qquad k = 1, ..., c, \tag{2.32}$$

where

$$\tilde{d}_{p,i,k} := \begin{cases} d_{2p,i,k} & k = 1, ..., \frac{c}{2}, \\ d_{2p+1,i,k-\frac{c}{2}} & k = \frac{c}{2}+1, ..., c. \end{cases}$$

We notice that in (2.32) we have $k = 1, ..., c$, while previously $k = 1, ..., \frac{c}{2}$. This is due to the splitting operation that *doubles* the equations (2.31). Now the indices i and k take values in the same range.

Without loss of generality, let us assume $\bar{p}$ odd. Using again Hypothesis 2, we have that $\tilde{d}_{p,i,k} = 0$ if $p \notin \{0, ..., \frac{\bar{p}-1}{2}\}$ and $v_{n,i} = 0$ if $n \notin \{-N, ..., N\}$. So, we can rewrite (2.32) in a compact form as $\tilde{D}v = 0$, where $\tilde{D}$ is a matrix of size $c(2N + \frac{\bar{p}+1}{2}) \times c(2N + 1)$ of the form

$$\tilde{D} = \begin{pmatrix} \tilde{D}_0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ \tilde{D}_{\frac{\bar{p}-1}{2}} & \cdots & \tilde{D}_0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \tilde{D}_{\frac{\bar{p}-1}{2}} & \cdots & \tilde{D}_0 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \tilde{D}_{\frac{\bar{p}-1}{2}} & \cdots & \tilde{D}_0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & \tilde{D}_{\frac{\bar{p}-1}{2}} \end{pmatrix}, \tag{2.33}$$

where $\tilde{D}_p$ is the $c \times c$ matrix defined by $(\tilde{D}_p)_{i,k} := \tilde{d}_{p,i,k}$, and v is a vector of size $c(2N+1)$ such that

$$v = \begin{pmatrix} v_{-N} \\ \vdots \\ v_N \end{pmatrix},$$

where v_n is a vector of size c such that $(v_n)_i := v_{n,i}$. To clarify the block structure of $\tilde{D}$, the (m, n) block of size $c \times c$ is $\tilde{D}_{m-n}$, defined as $(\tilde{D}_{m-n})_{i,k} := \tilde{d}_{m-n,i,k}$, if $0 \leq m-n \leq \frac{\bar{p}-1}{2}$ and the zero matrix otherwise, where $n = -N, ..., N$ and $m = -N, ..., N + \frac{\bar{p}-1}{2}$. We observe that the matrix D^l defined in (2.21) corresponds to $\tilde{D}_0$ defined above and by Hypothesis 2 $\det(D^l) = \det(\tilde{D}_0) \neq 0$. Therefore, the rank of $\tilde{D}$ is maximum, since the determinant of the $c(2N + 1) \times c(2N + 1)$ block-triangular matrix

$$\begin{pmatrix} \tilde{D}_0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ \tilde{D}_{\frac{\bar{p}-1}{2}} & \cdots & \tilde{D}_0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \tilde{D}_{\frac{\bar{p}-1}{2}} & \cdots & \tilde{D}_0 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \tilde{D}_{\frac{\bar{p}-1}{2}} & \cdots & \tilde{D}_0 \end{pmatrix}$$

is not zero. Hence, $v_{n,i} = 0$ for every $n \in \mathbb{Z}$ and $i = 1, ..., c$.

Step 2: *Injectivity of* $\tilde{\sigma}|_W$. Let $W = W_l$, where W_l is defined in Lemma 3. Therefore, there exists $N \in \mathbb{N}$ such that $W = (\text{span}\{\phi_{j,n}\}_{n=-N}^{N})^c$. We prove that $\sigma_W := P_W \circ \sigma|_W : W \to W$ is injective. This implies that $\tilde{\sigma}|_W = P_{(V_j)^c} \circ \sigma|_W : W \to (V_j)^c$ is injective. Without loss of generality, set $c = 1$.

To do this, we use Theorem 5. By Hypothesis 1, we have $W \subset L^2(\mathbb{R}) \cap L^\infty(\mathbb{R})$. Thanks to Lemma 3, we can identify any function $f \in W$ with a vector $y \in \mathbb{R}^{2N+1}$ whose entries are $\langle f, \phi_{j,n} \rangle_2$ for $n = -N, ..., N$. Therefore, σ_W can be seen as a map

$$\sigma_W : \mathbb{R}^{2N+1} \to \mathbb{R}^{2N+1}.$$

To prove the injectivity of σ_W, we verify the conditions of Theorem 5.

1. The function $\sigma_W \in C^1(W)$, because it is the composition of a linear function, the projection P_W, and the nonlinearity, σ, which is of class C^1 thanks to Lemma 4.

2. We now show that $\det(J_{\sigma_W}(y)) \neq 0$. Using $g \in W$ instead of the corresponding vector $y \in \mathbb{R}^{2N+1}$, we observe that

$$J_{\sigma_W}(g) = (P_W \circ \sigma')(g),$$

thanks to the linearity of the projection. In order to show that

$$\det(J_{\sigma_W}(g)) \neq 0$$

for every $g \in W$, by Lemma 4 we need to prove that $P_W(\sigma'(g(\cdot))f) = 0$ a.e. implies $f = 0$ a.e. Thanks to the fact that $f \in W$, we have

$$\langle \sigma'(g(\cdot))f, f \rangle_2 = 0.$$

But $\sigma'(x) > 0$ for every $x \in \mathbb{R}$, and so the last equation implies $f = 0$ a.e.

3. We now verify that $|\sigma_W(y)|_{\mathbb{R}^{2N+1}} \to +\infty$ as $|y|_{\mathbb{R}^{2N+1}} \to +\infty$. We observe that $|y|_{\mathbb{R}^{2N+1}} \to +\infty$ implies that $\|f\|_2 \to +\infty$, where $f \in W$ has been identified with y. Moreover,

$$\|\sigma_W(f)\|_2 = \|(P_W \circ \sigma)(f)\|_2 \geq \langle \sigma(f), \frac{f}{\|f\|_2} \rangle_2,$$

because $\frac{f}{\|f\|_2} \in W$ with unitary norm. Using the fact that $x \cdot \sigma(x) \geq 0$ for every $x \in \mathbb{R}$ (see Condition 2 of Hypothesis 3), we have

$$\langle \sigma(f), \frac{f}{\|f\|_2} \rangle_2 = \frac{1}{\|f\|_2} \int_{\mathbb{R}} \sigma(f(x))f(x)dx = \frac{1}{\|f\|_2} \int_{\mathbb{R}} |\sigma(f(x))||f(x)|dx.$$

Consider $x \geq 0$. Applying the mean value theorem to σ in the interval $[0, x]$ we have that $\sigma(x) = \sigma'(\xi)x$ with $\xi \in [0, x]$. Thanks to the fact that $\sigma'(x) \geq M_1$ for

every $x \in \mathbb{R}$ (see Condition 1 of Hypothesis 3), we obtain that $|\sigma(x)| \geq M_1|x|$ for every $x \geq 0$. The same holds for $x < 0$. Therefore,

$$\frac{1}{\|f\|_2}\int_{\mathbb{R}} |\sigma(f(x))||f(x)|dx \geq \frac{M_1}{\|f\|_2}\int_{\mathbb{R}} (f(x))^2 dx = M_1\|f\|_2.$$

Summing up, we have

$$\|\sigma_W(f)\|_2 \geq M_1\|f\|_2 \to +\infty, \qquad \|f\|_2 \to +\infty,$$

and then we obtain the thesis.

□

Remark 10. As we can see in the first step of the proof, the second condition in Hypothesis 2 is sufficient but not necessary. It is indeed enough to ask that the rank of $\tilde{D}$, defined in (2.33), is maximum.

Moreover, we observe that the maximum rank condition may not hold if we have $\det(\tilde{D}_0) = 0$, but $\det(\tilde{D}_p) \neq 0$ for some $p \in \{1, ..., \bar{p} - 1\}$. Indeed, if we consider the matrix

$$\begin{pmatrix} A & 0 & 0 \\ I & A & 0 \\ A & I & A \\ 0 & A & I \\ 0 & 0 & A \end{pmatrix}$$

where $0 \in \mathcal{M}_{2\times 2}$ is the zero matrix, $I \in \mathcal{M}_{2\times 2}$ is the identity matrix and $A \in \mathcal{M}_{2\times 2}$ is such that

$$\begin{pmatrix} 1/\sqrt{2} & 0 \\ 0 & 0 \end{pmatrix},$$

it is easy to verify that the rank is not maximum.

2.2.2 Extensions

The formulation of Theorem 1 is presented within the context of a simplified framework, which is as follows:

1. non-expansive convolutional layers;
2. stride $s = \frac{1}{2}$ for each convolutional layer;
3. one-dimensional signals.

In the following subsections we extend our theory by weakening these assumptions.

2.2.2.1 Expansive convolutional layers and arbitrary stride

In Section 2.2, we considered the case where the stride is $s = \frac{1}{2}$ for each layer and where the number of channels scaled exactly by the factor s at each layer, i.e. $c_l = sc_{l-1}$. Here, we generalize Theorem 1 to the case of stride $s_l = \frac{1}{2^{\nu_l}}$, with $\nu_l \in \mathbb{N}$, for the l-th convolutional layer and by considering expansive layers, i.e. $c_l \geq sc_{l-1}$. In this case, D^l are not necessarily square matrices, so we need to impose a condition on their rank.

Hypothesis 5. Let $\bar{p} \in \mathbb{N}$. For every $l = 2, ..., L$, the convolutional filters $t^l_{i,k} \in V_{j_l}$ of the l-th convolutional layer (2.12) satisfy

$$t^l_{i,k} = \sum_{p=0}^{\bar{p}} d^l_{p,i,k} \phi_{j_l,p}, \qquad i = 1, ..., c_{l-1}, \qquad k = 1, ..., c_l,$$

where $d^l_{p,i,k} \in \mathbb{R}$, and the rank of D^l is maximum, where D^l is a $2^{\nu_l} c_l \times c_{l-1}$ matrix defined by

$$(D^l)_{i,k} := \begin{cases} d^l_{0,i,k} & k = 1, ..., c_l, \\ d^l_{1,i,k-c_l} & k = c_l + 1, ..., 2c_l, \\ d^l_{2,i,k-2c_l} & k = 2c_l + 1, ..., 3c_l, \\ \vdots & \\ d^l_{2^{\nu_l}-1,i,k-(2^{\nu_l}-1)c_l} & k = (2^{\nu_l} - 1)c_l + 1, ..., 2^{\nu_l} c_l. \end{cases}$$

Theorem 6. *Let $L \in \mathbb{N}^*$ and $j_1 \in \mathbb{Z}$. Let $c_1, \ldots, c_L \in \mathbb{N}^*$ and $\nu_2, \ldots, \nu_L \in \mathbb{N}$ such that $c_L = 1$, c_{l-1} is divisible by 2^{ν_l} and $c_l \geq \frac{c_{l-1}}{2^{\nu_l}}$. Set $j_l = j_{l-1} + \nu_l$ for every $l = 2, ..., L$. Let V_{j_l} be the spaces of an MRA and $t^l_{i,k} \in V_{j_l}$ for every $l = 2, ..., L$, $i = 1, ..., c_{l-1}$ and $k = 1, ..., c_l$. Let $\tilde{\Psi}_l$ and $\tilde{\sigma}_l$ be defined as in (2.14) and (2.15), respectively. Let Ψ_1 be defined as in (2.9). If Hypotheses 1, 3, 4 and 5 are satisfied, then the generator G defined in (2.13) is injective.*

Sketch of the proof. The proof of the injectivity of $\tilde{\sigma}$ is the same as in Theorem 1. Moreover, the injectivity of $\tilde{\Psi}$ is guaranteed by Hypothesis 5 by following the same argument used in the proof of Theorem 1. □

We observe that, thanks to equation (2.8), choosing $j_l = j_{l-1} + \nu_l$ is equivalent to imposing that the stride of the l-th convolutional layer is $s_l = \frac{1}{2^{\nu_l}}$ with $\nu_l \in \mathbb{N}$.

2.2.2.2 2D CGNNs

We recall the principal concepts of 2D wavelet analysis (see [166, Section 7.7] for more details). In 2D, the scaling function spaces become $V_j \otimes V_j$ with $j \in \mathbb{Z}$, with orthonormal basis given by $\{\phi_{j,(n_1,n_2)}\}_{(n_1,n_2)\in\mathbb{Z}^2}$, where

$$\phi_{j,(n_1,n_2)}(x_1, x_2) = \phi_{j,n_1}(x_1)\phi_{j,n_2}(x_2),$$

and $\{\phi_{j,n}\}_{n\in\mathbb{Z}}$ is an orthonormal basis of V_j. We recall that

$$V \otimes W = \overline{\text{span}\{f \otimes g : f \in V, g \in W\}},$$

where $(f \otimes g)(x) = f(x_1)g(x_2)$. As in 1D, the MRA properties of Definition 1 hold:

1. $\bigcup_{j\in\mathbb{Z}} (V_j \otimes V_j)$ is dense in $L^2(\mathbb{R}^2) \cong L^2(\mathbb{R}) \otimes L^2(\mathbb{R})$ and $\bigcap_{j\in\mathbb{Z}} (V_j \otimes V_j) = \{0\}$;
2. $f \in V_j \otimes V_j$ if and only if $f(2^{-j}\cdot, 2^{-j}\cdot) \in V_0 \otimes V_0$;
3. and $\{\phi_{0,(n_1,n_2)}\}_{n_1,n_2\in\mathbb{Z}}$ is an orthonormal basis of $V_0 \otimes V_0$.

Moreover, $V_j \otimes V_j \subset V_{j+1} \otimes V_{j+1}$ for every $j \in \mathbb{Z}$, as in 1D.

Now, we can specify the structure of the nonlinearities, the fully connected layer and the convolutional ones in the 2D setting.

The nonlinearities σ_l act on functions in $L^2(\mathbb{R}^2)^{c_l}$ by pointwise evaluation:

$$\sigma_l(f)(x) = \sigma_l(f(x)), \qquad \text{a.e. } x \in \mathbb{R}^2.$$

The fully connected layer is defined as $\Psi_1 := F \cdot + b$, where $F\colon \mathbb{R}^S \to (V_{j_1} \otimes V_{j_1})^{c_1}$ is a linear map and $b \in (V_{j_1} \otimes V_{j_1})^{c_1}$.

The convolutional layers are

$$\bar{\Psi}_l = P_{(V_{j_l}\otimes V_{j_l})^{c_l}} \circ \Psi_l \circ P_{(V_{j_{l-1}}\otimes V_{j_{l-1}})^{c_{l-1}}},$$

where the convolution $\Psi_l\colon L^2(\mathbb{R}^2)^{c_{l-1}} \to L^2(\mathbb{R}^2)^{c_l}$ is given by

$$(\Psi_l(x))_k := \sum_{i=1}^{c_{l-1}} x_i * t^l_{i,k} + b^l_k, \qquad k = 1, ..., c_l, \tag{2.34}$$

with filters $t^l_{i,k} \in (V_{j_l} \otimes V_{j_l}) \cap L^1(\mathbb{R}^2)$ and biases $b^l_k \in V_{j_l} \otimes V_{j_l}$. In (2.34), the symbol $*$ denotes the usual convolution $L^2(\mathbb{R}^2) \times L^1(\mathbb{R}^2) \to L^2(\mathbb{R}^2)$. Note that, in analogy to the 1D case, the convolution of a filter in $V_{j+\nu} \otimes V_{j+\nu}$ with a function in $V_j \otimes V_j$ produces a function that does not necessarily belong to $V_{j+\nu} \otimes V_{j+\nu}$. In view of identity (2.8), this corresponds to a stride $s = (\frac{1}{2^\nu}, \frac{1}{2^\nu})$.

We can finally define the CGNN architecture in 2D:

$$\begin{aligned} G\colon \mathbb{R}^S \xrightarrow[\text{f.c.}]{\Psi_1} (V_{j_1} \otimes V_{j_1})^{c_1} \xrightarrow[\text{nonlin.}]{\sigma_1} L^2(\mathbb{R}^2)^{c_1} \xrightarrow[\text{proj.}]{P_{(V_{j_1}\otimes V_{j_1})^{c_1}}} (V_{j_1} \otimes V_{j_1})^{c_1} \xrightarrow[\text{conv.}]{\Psi_2} L^2(\mathbb{R}^2)^{c_2} \\ \xrightarrow[\text{proj.}]{P_{(V_{j_2}\otimes V_{j_2})^{c_2}}} (V_{j_2} \otimes V_{j_2})^{c_2} \xrightarrow[\text{nonlin.}]{\sigma_2} L^2(\mathbb{R}^2)^{c_2} \xrightarrow[\text{proj.}]{P_{(V_{j_2}\otimes V_{j_2})^{c_2}}} (V_{j_2} \otimes V_{j_2})^{c_2} \xrightarrow[\text{conv.}]{\Psi_3} ... \\ ... \xrightarrow[\text{conv.}]{\Psi_L} L^2(\mathbb{R}^2) \xrightarrow[\text{proj.}]{P_{V_{j_L}\otimes V_{j_L}}} V_{j_L} \otimes V_{j_L} \xrightarrow[\text{nonlin.}]{\sigma_L} L^2(\mathbb{R}^2) \xrightarrow[\text{proj.}]{P_{V_{j_L}\otimes V_{j_L}}} V_{j_L} \otimes V_{j_L}, \end{aligned}$$

which can be summarized as

$$G = \left(\bigcirc_{l=L}^{2} \tilde{\sigma}_l \circ \tilde{\Psi}_l \right) \circ (\tilde{\sigma}_1 \circ \Psi_1), \tag{2.35}$$

where

$$\tilde{\Psi}_l := P_{(V_{j_l} \otimes V_{j_l})^{c_l}} \circ \Psi_l \colon (V_{j_{l-1}} \otimes V_{j_{l-1}})^{c_{l-1}} \to (V_{j_l} \otimes V_{j_l})^{c_l}, \qquad l = 2, ..., L, \tag{2.36}$$

and

$$\tilde{\sigma}_l := P_{(V_{j_l} \otimes V_{j_l})^{c_l}} \circ \sigma_l \colon (V_{j_l} \otimes V_{j_l})^{c_l} \to (V_{j_l} \otimes V_{j_l})^{c_l}, \qquad l = 1, ..., L. \tag{2.37}$$

For simplicity, we state our injectivity result using non-expansive convolutional layers and stride $(\frac{1}{2}, \frac{1}{2})$ for each convolutional layer. The result can be extended to arbitrary strides and expansive layers by adapting the arguments in 2.2.2.1.

Hypothesis 6. Let $\bar{p} \in \mathbb{N}$. For every $l = 2, ..., L$, the convolutional filters $t^l_{i,k} \in V_{j_l} \otimes V_{j_l}$ of the l-th convolutional layer satisfy

$$t^l_{i,k} = \sum_{p_1=0}^{\bar{p}} \sum_{p_2=0}^{\bar{p}} d^l_{p,i,k} \phi_{j_l,p},$$

where $d^l_{p,i,k} \in \mathbb{R}$ and $p = (p_1, p_2)$, and $\det(D^l) \neq 0$, where D^l is a $\frac{c_1}{2^{2l-2}} \times \frac{c_1}{2^{2l-2}}$ matrix defined by

$$(D^l)_{i,k} := \begin{cases} d^l_{(0,0),i,k}, & k = 1, ..., \frac{c_1}{2^2 2^{l-1}}, \\ d^l_{(1,0),i,k-\frac{c_1}{2^2 2^{l-1}}}, & k = \frac{c_1}{2^2 2^{l-1}} + 1, ..., \frac{c_1}{2^{2l-1}}, \\ d^l_{(0,1),i,k-\frac{2c_1}{2^2 2^{l-1}}}, & k = \frac{c_1}{2^{2l-1}} + 1, ..., \frac{3c_1}{2^2 2^{l-1}}, \\ d^l_{(1,1),i,k-\frac{3c_1}{2^2 2^{l-1}}}, & k = \frac{3c_1}{2^2 2^{l-1}} + 1, ..., \frac{c_1}{2^{l-1}}. \end{cases}$$

Hypothesis 7. We assume that

- The linear function $F \colon \mathbb{R}^S \to (V_{j_1})^{c_1} \otimes (V_{j_1})^{c_1}$ is injective;
- There exists $N \in \mathbb{N}$ such that $b \in (\text{span}\{\phi_{j_1,n}\}^N_{n_1,n_2=-N})^{c_1}$ and $\text{Im}(F) \subset (\text{span}\{\phi_{j_1,n}\}^N_{n_1,n_2=-N})^{c_1}$ with $n = (n_1, n_2)$.

Theorem 7. *Let $L \in \mathbb{N}^*$ and $j_1 \in \mathbb{Z}$. Let $c_1 = 2^{2L-2}$, $c_l = \frac{c_1}{2^{2l-2}}$ and $j_l = j_1 + l - 1$ for every $l = 1, ..., L$. Let V_{j_l} be scaling function spaces arising from an MRA and $t^l_{i,k} \in V_{j_l} \otimes V_{j_l}$ for every $l = 2, ..., L$, $i = 1, ..., c_{l-1}$ and $k = 1, ..., c_l$. Let $\tilde{\Psi}_l$ and $\tilde{\sigma}_l$ be defined as in (2.36) and (2.37), respectively. Let $F \colon \mathbb{R}^S \to (V_{j_1})^{c_1} \otimes (V_{j_1})^{c_1}$ be a linear map and $b \in (V_{j_1})^{c_1} \otimes (V_{j_1})^{c_1}$. If Hypotheses 1, 3, 6 and 7 are satisfied, then the generator G defined in (2.35) is injective.*

Sketch of the proof. The proof follows from the same arguments of the proof of Theorem 1, by considering $n = (n_1, n_2), m = (m_1, m_2), p = (p_1, p_2) \in \mathbb{Z}^2$ and $\xi = (\xi_1, \xi_2) \in [0, 1]^2$. □

2.3 Stability of inverse problems with generative models

We now show how an injective CGNN can be used to solve ill-posed inverse problems. The purpose of a CGNN is to reduce the dimensionality of the unknown to be determined, and the injectivity is the main ingredient to obtain a rigorous stability estimate.

We consider an inverse problem of the form

$$y = \mathcal{F}(x), \tag{2.38}$$

where $\mathcal{F}$ is a possibly nonlinear map between Banach spaces, X and Y, modeling a measurement (forward) operator, $x \in X$ is a quantity to be recovered and $y \in Y$ is the noisy data. Typical inverse problems are ill-posed (e.g. CT, accelerated MRI, or electrical impedance tomography), meaning that the noise in the measurements is amplified in the reconstruction. For instance, in the linear case, this instability corresponds to having an unbounded (namely, not Lipschitz) inverse $\mathcal{F}^{-1}$. The ill-posedness is classically tackled by using regularization, which often leads to an iterative method, as the gradient-type Landweber algorithm [79]. This can be very expensive if X has a large dimension.

However, in most of the inverse problems of interest, the unknown x can be modeled as an element of a low-dimensional manifold in X. We choose to use a generator $G\colon \mathbb{R}^S \to X$ to perform this dimensionality reduction and therefore our problem reduces to finding $z \in \mathbb{R}^S$ such that

$$y = \mathcal{F}(G(z)). \tag{2.39}$$

In practice, the map G is found via an unsupervised training procedure, starting from a training dataset. From the computational point of view, solving (2.39) with an iterative method is clearly more advantageous than solving (2.38), because z belongs to a lower dimensional space. We note that the idea of solving inverse problems using deep generative models has been considered in [44, 204, 22, 123, 122, 176, 221, 24].

The dimensionality reduction given by the composition of the forward operator with a generator as in (2.39), has a regularizing/stabilizing effect that we aim to quantify. More precisely, we show that an injective CGNN yields a Lipschitz stability result for the inverse problem (2.39); in other words, the inverse map is Lipschitz continuous, and noise in the data is not amplified in the reconstruction. For simplicity, we consider the 1D case with stride $\frac{1}{2}$ and non-expansive convolutional layers, but the result can be extended to the 2D case and arbitrary stride as done in 2.2.2 for Theorem 1.

Theorem 8. *Let $X = L^2(\mathbb{R})$ and G be a CGNN satisfying Hypotheses 1, 2, 3 and 4. Let $\mathcal{M} := G(\mathbb{R}^S)$, $K \subset \mathcal{M}$ be a compact set, Y be a Banach space and $\mathcal{F}\colon X \to Y$ be a C^1 map (possibly nonlinear). Assume that $\mathcal{F}$ is injective and $\mathcal{F}'(x)|_{T_x\mathcal{M}}$ is injective for every $x \in \mathcal{M}$. Then there exists a constant $C > 0$ such that*

$$\|x - y\|_X \leq C\|\mathcal{F}(x) - \mathcal{F}(y)\|_Y, \qquad x, y \in K.$$

The same result can be obtained also in the case of non-expansive convolutional layers and arbitrary stride, under the hypotheses of Theorem 6. Moreover, it can be extended to the 2D case under the hypotheses of Theorem 7.

For proving Theorem 8, we first prove that the Fréchet derivative of an injective CGNN is injective as well.

Proposition 9. *Let $X = L^2(\mathbb{R})$ and G be a CGNN satisfying the hypotheses of Theorem 1. Then the generator G is injective, of class C^1 and $G'(z)$ is injective for every $z \in \mathbb{R}^S$.*

Proof. The injectivity of G is proved in Theorem 1, and the continuous differentiability of G follows from the fact that it is a composition of continuously differentiable functions. We only need to prove the injectivity of G'. Let W_l be defined as in Lemma 3, for $l = 1, \dots, L$. The derivative of G can be written as

$$G' = \left(\bigcirc_{l=L}^{2} (P_{(V_{j_l})^{c_l}} \circ \sigma_l)' (P_{(V_{j_l})^{c_l}} \circ \Psi_l)' \right) (P_{(V_{j_1})^{c_1}} \circ \sigma_1)' F', \tag{2.40}$$

where, for simplicity, we omitted the arguments of each term. The injectivity of $G'(z)$ for every $z \in \mathbb{R}^S$ follows from the injectivity of each component of (2.40). Indeed, $F' = F$ is injective for every $z \in \mathbb{R}^S$ by Hypothesis 4. Moreover, for every $l = 2, ..., L$, $(P_{(V_{j_l})^{c_l}} \circ \Psi_l)'|_{W_{l-1}} = P_{(V_{j_l})^{c_l}} \circ \Psi_l|_{W_{l-1}}$ is injective, as we prove in Step 1 of the proof of Theorem 1.

Finally, we prove that $\tilde{\sigma}_l'|_{W_l}(g) = (P_{(V_j)^c} \circ \sigma_l|_{W_l})'(g)$ is injective for every $l = 1, ..., L$. To do this, we observe that a stronger condition holds i.e. $\sigma'_{W_l}(g) := (P_{W_l} \circ \sigma_l|_{W_l})'(g)$ is injective for every $l = 1, ..., L$. Indeed, the deteminant of its Jacobian is not zero for every $g \in W_l$, as shown in Step 2 of the proof of Theorem 1. □

Now we are able to prove Theorem 8.

Proof of Theorem 8. Thanks to Proposition 9, G is injective, of class C^1 and $G'(z)$ is injective for every $z \in \mathbb{R}^S$. Therefore, $\mathcal{M} = G(\mathbb{R}^S)$ is a S-dimensional differentiable manifold embedded in $X = L^2(\mathbb{R})$, considering as atlas the one formed by only one chart $\{\mathcal{M}, G^{-1}\}$. Moreover, $\mathcal{M}$ is a Lipschitz manifold, since G is a composition of Lipschitz maps. Indeed, it is composed by affine maps and Lipschitz nonlinearities (see Hypothesis 3). Then the result immediately follows from [5, Theorem 2.2]. □

The Lipschitz stability estimate provided in Theorem 8 can also be obtained in the case when only finite measurements are available, i.e. a suitable finite-dimensional approximation of $\mathcal{F}(x)$, thanks to [5, Theorem 2.5]. A similar estimate can be derived for $\|G^{-1}(x) - G^{-1}(y)\|_{\mathbb{R}^S}$ [5, Proof of Theorem 2.2], even though this bound in the latent space is less relevant for inverse problems.

Moreover, this Lipschitz stability estimate ensures the convergence of the Landweber algorithm (see [72]), which can be used as a reconstruction method. In our setting, this algorithm is applied to the functional $\mathcal{F} \circ G$ and, given an initial guess z_0, it produces a sequence of iterations z_k

$$z_k = z_{k-1} - h\nabla(\mathcal{F} \circ G)(z_{k-1}), \qquad k \geq 1, \tag{2.41}$$

where $h > 0$ is the stepsize.

Although the injectivity of the generator is not a necessary condition for solving inverse problems of the type (2.39), in order to prove theoretical results about inverse problems, it is still a mandatory assumption, because of the proof techniques. It is possible nevertheless that global injectivity of the generator is not needed, and a weaker local one might suffice. This could be justified by using a differential geometric approach, as in [5]. In Remark 11 below, we provide a toy example in which a non-injective generator makes the Landweber iteration not convergent. However, it is still not clear, from the theoretical point of view, whether non-injective models can be successfully used to solve inverse problems.

Remark 11. We demonstrate how a non-injective generative model can lead to difficulties in solving inverse problems. For simplicity, we consider a simple one-dimensional toy example, with a ReLU activation function σ, which is not injective. Let

$$G\colon \mathbb{R} \to \mathbb{R}, \qquad G(z) = \sigma(-\sigma(z) + 1) = (-z_+ + 1)_+,$$

where $z_+ = \max(z, 0)$. This is a simple 2-layer neural network, where the affine map in the first layer is $z \mapsto z$, and the affine map of the second layer is $x \mapsto -x + 1$. It is immediate to see that this network generates the set $[0, 1] \subset \mathbb{R}$. However, it is not injective:

$$G(z) = \begin{cases} 1 & \text{if } z \leq 0, \\ 0 & \text{if } z \geq 1. \end{cases} \tag{2.42}$$

Suppose now we wish to solve the inverse problem $\mathcal{F}(x) = y$, as in (2.38) with any map $\mathcal{F}$, for $x \in [0, 1]$. Writing $x = G(z)$, we are reduced to solving (2.39), namely $\mathcal{F}(G(z)) = y$. If we solve this by using any iterative method, as (2.41), with an initial guess $z_0 \in \mathbb{R} \setminus [0, 1]$, we have $z_k = z_0$ for every k, since G is constant in a neighborhood of z_0 by (2.42). Therefore, in general, it will not be possible to solve the inverse problem with a non-injective generative model.

2.4 Numerical results

We present here numerical results validating our theoretical findings. In Section 2.4.1 we describe how we train a CGNN, in Section 2.4.2 we compare a CGNN-based reconstruction algorithm with a standard discrete generator one for a signal deblurring problem, and in Section 2.4.3 we show qualitative results on the generation and reconstruction capabilities of CGNNs. The treatment of more complicated inverse problems such as nonlinear ones (e.g. EIT problem) has not been treated for computational reasons.

2.4.1 Training

The conditions for injectivity given in Theorem 1 are not very restrictive and we can use an unsupervised training protocol to choose the parameters of a CGNN. Even though our theoretical results concern only the injectivity of CGNNs, we numerically verified that training a generator to also well-approximate a probability distribution gives better

reconstructions for inverse problems. For this reason, we choose to train CGNNs as parts of variational autoencoders [134, 135], a popular architecture for generative modeling. In particular, our VAEs are designed so that the corresponding decoder has a CGNN architecture. Note that there is growing numerical evidence showing that an untrained convolutional network is competitive with trained ones for solving inverse problems with generative priors [221, 24].

In our experiment, the training is done on smooth signals. We create a dataset of smooth signals constructed by randomly sampling the first 5 low-frequency Fourier coefficients. Each of these is taken from a Gaussian distribution with zero mean and variance that decreases as the frequency increases. Mathematically, the dataset contains signals of the form

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{2} a_n \cos(2\pi n t) + b_n \sin(2\pi n t),$$

where $a_n, b_n \sim N\left(0, \frac{1}{(n+1)^6}\right)$. In order to show the validity of our method in a high resolution setting, the support of the signals $[0, 1]$ is finely discretized with 4096 equidistant points. We divide the dataset in 10000 signals to train the network and 2000 to test it.

Our VAE consists of a decoder with 3 non-expansive transposed convolutional layers without bias terms and an encoder with a similar, yet mirrored, structure. For the decoder, we consider the simplified CGNN architecture described in Remark 9. Differently from the implementation described in Section 2.1.3, we do not pass from the scaling coefficients in V_j to the ones in V_{j+M} and vice-versa at every layer. Here, both the nonlinearity and the convolution are applied to the scaling coefficients in V_j. Therefore, our VAE takes as input the scaling coefficients of our signals obtained by doing 6 downscalings, which already constitute a significant dimensionality reduction with respect to the original finely discretized signals (the scaling coefficients are approximately $\frac{4096}{2^6} = 64$). We numerically verified that the simplified CGNN architecture provides very similar results to the original architecture. In addition, we also verified that it is less computationally expensive, since the whole network acts only on the scaling coefficients. For our experiments, we choose the stride $s = \frac{1}{2}$, the latent space dimension $S = 15$ and the leaky-ReLU as nonlinear activation function. Moreover we consider Daubechies scaling function spaces with vanishing moments $N = 1, 2, 6, 10$.

The training is done with the Adam optimizer using a learning rate of 0.01 and the loss function, commonly used for VAEs, given by the weighted sum of two terms: the Mean Square Error (MSE) between the original and the generated signals and the Kullback-Leibler Divergence (KLD) between the standard Gaussian distribution and the one generated by the encoder in the latent space[1].

The injectivity of the decoder, i.e. of the generator, is guaranteed if Hypotheses 1, 2 and 4 are satisfied and if σ is injective (see Remark 9). We test Hypothesis 2 a posteriori,

[1]All computations were implemented with Python3, running on a workstation with 256GB of RAM and 2.2 GHz AMD EPYC 7301 CPU and Quadro RTX 6000 GPU with 22GB of memory. All the codes are available at https://github.com/ContGenMod/Continuous-Generative-Neural-Network

i.e. after the training, and, in our cases, it is always satisfied. The other assumptions are verified using the leaky-ReLU as nonlinearity and the Daubechies scaling functions for the spaces V_j. We mention that leaky-ReLU was chosen not only because it satisfied the assumptions but also because it obtained better performance compared to other activation functions such as ReLU and ELU.

2.4.2 Deblurring with generative models

Deblurring problem. We consider the following deblurring problem

$$y = f * x + e,$$

where

- x is a smooth signal from the test set.
- f is the Gaussian blurring filter obtained by evaluating a $N(0, 4)$ in 1501 equidistant point in $[-4, 4]$.
- e is a weighted 4096 random Gaussian noise, i.e. $e = \tau\varepsilon$, where $\tau \geq 0$ is a weight and $\varepsilon \sim \mathcal{N}(0, I_{4096})$. We consider two different levels of noise corresponding to $\tau = 0$ (no noise, only blurring filter) and $\tau = 0.3$.
- The symbol $*$ represents the discrete convolution defined in equation (2.3).

In order to solve this deblurring problem, we compare the two following approaches.

Deep Landweber for discrete generators. We consider the Landweber algorithm applied to $y = f * G(z) + e$, where G is a discrete injective generative NN (as described in Section 2.1.1) giving as output discretized signals in $\mathbb{R}^{4096}$. In this case, the forward operator is $z \mapsto f * G(z)$ and the space in which the iterations are performed is the low-dimensional latent space $\mathbb{R}^{15}$, to which z belongs. In order to choose an appropriate initial value, one option is to generate vectors $z_i \in \mathbb{R}^{15}$ with random Gaussian entries and compute the MSE between the data y and $f * G(z_i)$ for every i. Then, we choose the z_i that minimizes the MSE. Otherwise, if G is the decoder of a discrete VAE, another option is to choose the initial guess as $E(y)$, where E is the encoder of the discrete VAE. However, we notice that in practice, for our deblurring problem, the algorithm converges to a good solution with almost every initial guess $z \in \mathbb{R}^{15}$ with random Gaussian entries. Therefore, our results are shown in this setting. The iterative step becomes

$$z_k = z_{k-1} - h(G'(z_{k-1}))^t f * (f * G(z_{k-1}) - y), \qquad k \geq 1, \qquad h = 0.0005 \text{ fixed.}$$

We run the algorithm until $\frac{\|z_{k+1} - z_k\|}{\|z_k\|} < \delta$, with $\delta = 10^{-12}$.

Deep Landweber for continuous generators. Next, we consider the Landweber algorithm applied to $y = f * G(z) + e$, where G is a simplified injective CGNN, as described in Remark 9. The generator may be decomposed as $G = \mathcal{W} \circ \tilde{G}$, where $\tilde{G}$ gives as outputs the scaling coefficients of the signals in V_{j_L} and $\mathcal{W}$ is an operator that synthesizes the scaling coefficients at level j_L, mapping them to a function in $L^2(\mathbb{R})$. As for the discrete case above, in practice the algorithm converges to the solution with almost every initial guess $z \in \mathbb{R}^{15}$ with random Gaussian entries and we show the results in this case. The iterative steps are

$$z_k = z_{k-1} - h(\tilde{G}'(z_{k-1}))^t \mathcal{W}^t \circ f * (f * G(z_{k-1}) - y), \qquad k \geq 1, \qquad h = 0.0005 \text{ fixed.}$$

The stopping criterion is the same as that of deep Landweber for discrete generators.

Theoretically, $\mathcal{W}$ maps a sequence of scaling coefficients $(c_n)_{n\in\mathbb{Z}} \in \mathbb{R}^{\mathbb{Z}}$ at level $j = j_L$ into $\sum_{n\in\mathbb{Z}} c_n \phi_{j,n} \in V_j \subset L^2(\mathbb{R})$ and the corresponding $\mathcal{W}^t$ maps a function $f \in L^2(\mathbb{R})$ into the sequence of scaling coefficients $(\langle f, \phi_{j,n}\rangle_{L^2(\mathbb{R})})_{n\in\mathbb{Z}}$. In practice, as explained in Section 2.1.3, we use the scaling coefficients in V_{j+M} for large M to approximate the function $\sum_{n\in\mathbb{Z}} c_n \phi_{j,n}$. Therefore, $\mathcal{W}$ consists of an upsampling transformation repeated M times, assuming all the detail coefficients are equal to zero, while $\mathcal{W}^t$ is an M-times downsampling. For our examples in Section 2.4.2, we consider $M = 6$.

Comparisons between continuous and discrete generator reconstruction algorithms. In Figure 2.4 we provide a comparison between the continuous generator approach using CGNNs living on Daubechies scaling spaces with vanishing moments $N = 1, 2, 6, 10$ in the reconstruction of blurry signals, with or without noise. We also compare them with the discrete generator approach where the generator is the decoder of a discrete VAE taking as input directly the finely discretized function, instead of the scaling coefficients. Although having the same basic structure, this discrete VAE will contain significantly more parameters (50 times more than the ones of the continuous VAEs whose decoders are the CGNNs living on Daubechies scaling spaces). In both cases, the inverse problem was solved by applying the iterative scheme (2.41) with a random initial guess as explained in the previous two paragraphs.

The original signal x is taken from our test set and the data y is obtained by blurring it with a Gaussian blurring operator, and by possibly adding Gaussian noise. We compare the original signal, the corrupted one and the reconstruction, by measuring the relative MSE.

We notice that the discrete VAE yields more irregular reconstructions. This may be due to the significantly higher number of parameters in this network, which is therefore more prone to overfitting. On the contrary, the networks acting on the scaling coefficients at low scales, thanks to the smoothness of Db6 and Db10, show smoother reconstructions, coherently with the signals' class. As one would expect, the shape of the wavelet affects the final reconstructions.

In Figure 2.5 we show two examples of reconstruction obtained with the iterative algorithm (2.41) starting from different initial guesses, with the Db6 scaling function

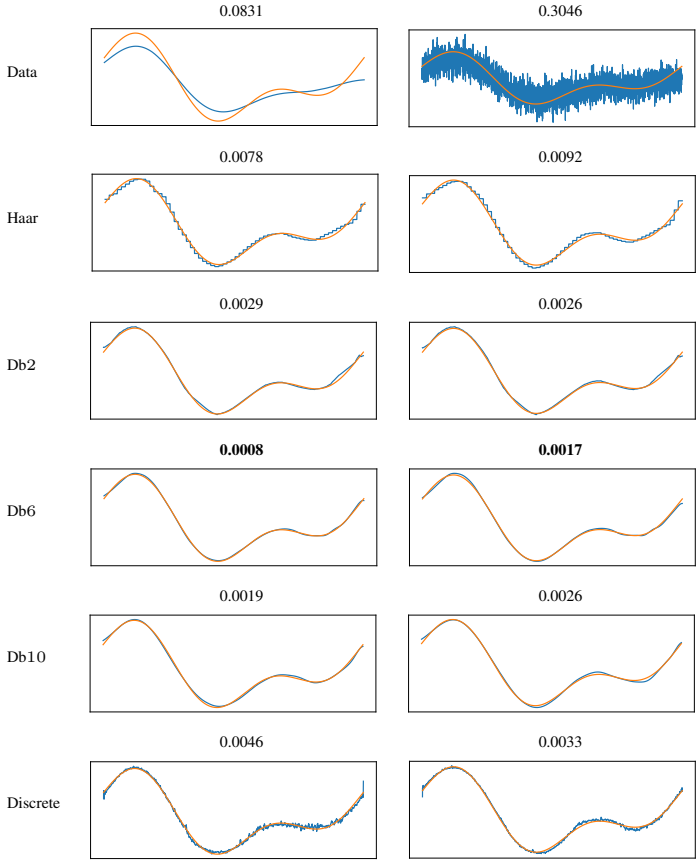


Figure 2.4: Image deblurring. First column: noiseless case. Second column: noisy case. In the first row: original signal x (orange) and blurry one y (blue). In the other rows: original signal x (orange) and reconstructed one (blue) using algorithm (2.41) where the generators are either CGNNs with different scaling functions or the discrete injective generative NN (last row). At the top of each picture: relative MSE between the two signals.

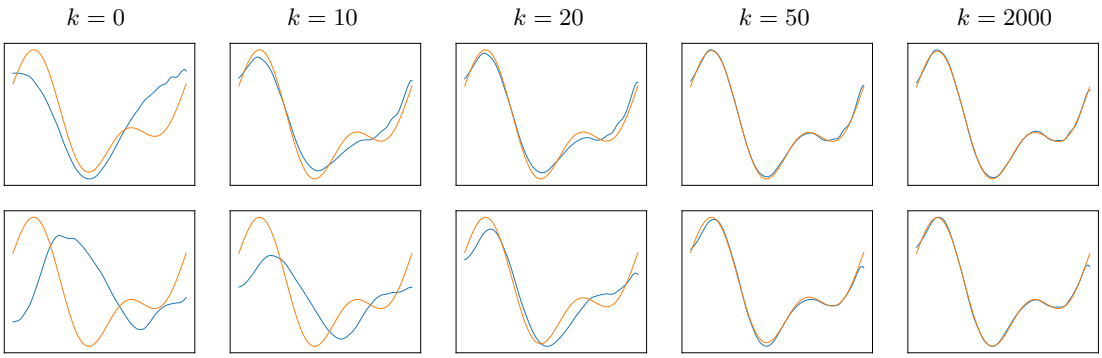


Figure 2.5: In each row: original signal x (orange) and reconstruction (blue) after k iterations of algorithm (2.41) where the generators are CGNNs with the Db6 scaling function. In this case the data is not corrupted by noise, but the same results are obtained also with Gaussian noise.

and blurry data y with no noise. We notice that the algorithm converges to the solution starting from an arbitrary initial guess.

2.4.3 Generation and reconstruction power of CGNNs

To assess the quality of the generation, in Figure 2.6 we show random samples of signals from trained injective CGNNs with different Daubechies scaling spaces and from the discrete injective generative NN.

In order to evaluate the reconstruction power of our trained VAEs (Figure 2.7), we compute the mean and the variance, over the 200 signals of the test set, of the MSE between the true signal and the reconstructed one, obtained by applying the full VAE to the true signal. We observe that although the discrete VAE has 50 times the parameters of the VAEs acting on the scale coefficients, the values of the MSE are comparable (especially in the case of smooth Daubechies wavelets such as db6 and db10). We also show qualitatively some examples of reconstructed signals in Figure 2.8. Even in these cases, the comparison takes into account the VAEs with different Daubechies scaling spaces and the discrete VAE described in the previous section.

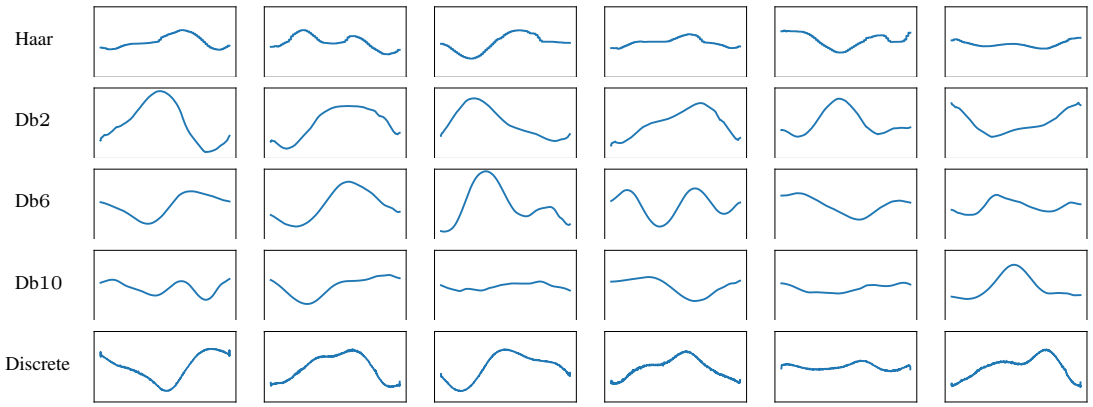


Figure 2.6: Random samples from trained injective CGNNs with different scaling functions and discrete injective generative NN.

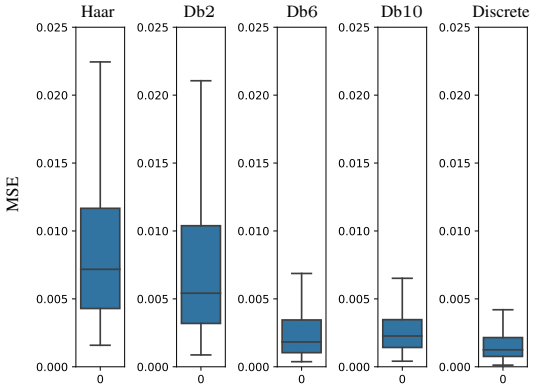


Figure 2.7: Boxplots of the MSE between the original image and the reconstructed one over 200 images of the test set using VAEs with different scaling functions and the discrete VAE.

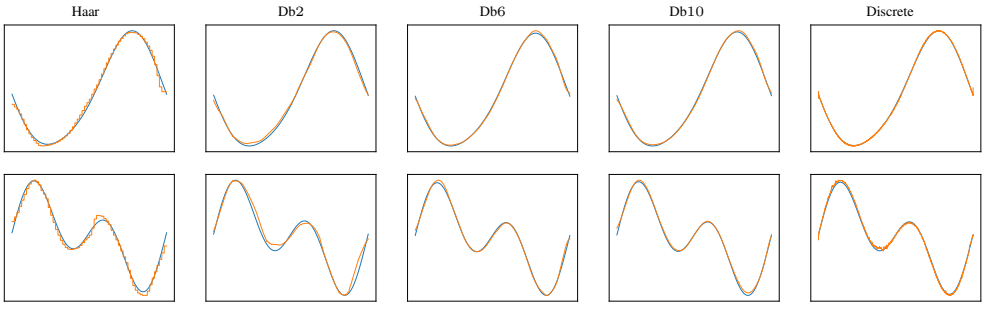


Figure 2.8: Original signal (blue) and reconstructed one (orange) using VAEs with different scaling functions and the discrete VAE.

Chapter 3

Manifold Learning by Mixture Models of Variational Autoencoders for Inverse Problems

The treatment of high-dimensional data is often computationally costly and numerically unstable. Therefore, in many applications, it is important to find a low-dimensional representation of high-dimensional datasets. Classical methods, like the principal component analysis (PCA) [182], assume that the data is contained in a low-dimensional subspace. However, for complex datasets this assumption appears to be too restrictive, particularly when working with image datasets. Therefore, recent methods rely on the so-called manifold hypothesis [34], stating that even complex and high-dimensional datasets are contained in a low-dimensional manifold. Based on this hypothesis, in recent years, many successful approaches have been based on generative models, able to represent high dimensional data in $\mathbb{R}^n$ by a generator $D\colon \mathbb{R}^d \to \mathbb{R}^n$ with $d \ll n$: these include generative adversarial networks [93], variational autoencoders [134], injective flows [141] and score-based diffusion models [214, 117]. Indeed under the assumption that D is injective, the set of generated points $\{D(z) : z \in \mathbb{R}^d\}$ forms a manifold that approximates the training set. However, this requires that the data manifold admits a global parameterization. In particular, it must not be disconnected or contain holes. In order to model disconnected manifolds, [81, 127, 138, 184] propose to model the latent space of a VAE by a Gaussian mixture model. This enables the authors to capture multimodal probability distributions. However, this approach struggles with modelling manifolds with holes since either the injectivity of the generator is violated or it is impossible to model overlapping charts. Similarly, the authors of [71, 173, 191] propose latent distributions defined on Riemannian manifolds for representing general topologies. In [172], the manifold is embedded into a higher-dimensional space, in the spirit of Whitney embedding theorem. However, these approaches have the drawback that the topology of the manifold has to be known a-priori,

which is usually not the case in practice.

Here, we focus on the representation of the data manifold by several charts. A chart provides a parameterization of an open subset from the manifold by defining a mapping from the manifold into a Euclidean space. Then, the manifold is represented by the collection of all of these charts, which is called atlas. For finding these charts, the authors of [65, 85, 185, 208] propose the use of clustering algorithms. By default, these methods do not provide an explicit formulation of the resulting charts. As a remedy, [45, 185] use linear or kernelized embeddings. The authors of [208] propose to learn for each chart again a generative model. However, these approaches often require a large number of charts and are limited to relatively low data dimensions. The idea of representing the charts by generative models is further elaborated in [139, 140, 201]. Here, the authors proposes to train at the same time several (non-variational) autoencoders and a classification network that decides for each point to which chart it belongs. In contrast to the clustering-based algorithms, the computational effort scales well for large data dimensions. On the other hand, the numerical examples in the corresponding papers show that the approach already has difficulties to approximate small toy examples like a torus.

In this chapter, we propose to approximate the data manifold by a mixture model of VAEs. Using Bayes theorem and the evidence lower bound (ELBO) approximation of the likelihood term we derive a loss function for maximum likelihood estimation of the model weights. Mixture models of generative models for modeling disconnected datasets were already considered in [29, 118, 158, 215, 229]. However, they are trained in a different way and to the best of our knowledge none of those is used for manifold learning.

As in the previous chapter, the learned manifold is used to encode a-priori information on the unknowns of inverse problems. We consider an observation y which is generated by the inverse problem

$$y = \mathcal{G}(x) + \eta,$$

where $\mathcal{G}\colon \mathbb{R}^n \to \mathbb{R}^m$ is an ill-posed or ill-conditioned, possibly nonlinear, forward operator and η represents additive noise. Reconstructing the input x directly from the observation y is usually not possible due to the ill-posed operator and the high dimension of the problem. As a remedy, the incorporation of prior knowledge is required. This is usually achieved by using regularization theory, namely, by minimizing the sum of a data fidelity term $F(x)$ and a regularizer $R(x)$, where F describes the fit of $\mathcal{G}(x)$ to y and R incorporates the prior knowledge. With the success of deep learning, data-driven regularizers became popular [17, 23, 95, 116, 162].

In this chapter, we consider a regularizer which constraints the reconstruction x to a learned data manifold $\mathcal{M}$. More precisely, we consider the optimization problem

$$\hat{x} = \arg\min_x F(x) \quad \text{subject to } x \in \mathcal{M},$$

where $F(x) = \frac{1}{2}\|\mathcal{G}(x) - y\|^2$ is a data-fidelity term. This corresponds to the regularizer $R(x)$ which is zero for $x \in \mathcal{M}$ and infinity otherwise. When the manifold admits a global parameterization given by one single generator D, the authors of [12, 56, 77, 91] propose

to reformulate the problem as $\hat{x} = D(\hat{z})$, where $\hat{z} \in \arg\min_z F(D(z))$. Since this is an unconstrained problem, it can be solved by gradient based methods. However, since we consider manifolds represented by several charts, this reformulation cannot be applied. As a remedy, we propose to use a Riemannian gradient descent scheme. In particular, we derive the Riemannian gradient using the decoders and encoders of our manifold and propose two suitable retractions for applying a descent step into the gradient direction.

To emphasize the advantage of using multiple generators, we demonstrate the performance of our method on numerical examples. We first consider some two- and three-dimensional toy examples. Finally, we apply our method to deblurring and to electrical impedance tomography, a nonlinear inverse problem consisting in the reconstruction of the leading coefficient of a second order elliptic PDE from the knowledge of the boundary values of its solutions [62]. The code of the numerical examples is available online[1].

The chapter is organized as follows. In Section 3.1, we revisit VAEs and fix the corresponding notations. Afterwards, in Section 3.2, we introduce mixture models of VAEs for learning embedded manifolds of arbitrary dimensions and topologies. Here, we focus particularly on the derivation of the loss function and of the architecture, which allows us to access the charts and their inverses. For minimizing functions defined on the learned manifold, we propose a Riemannian gradient descent scheme in Section 3.3. We provide numerical toy examples for one and two dimensional manifolds in Section 3.4. In Section 3.5, we discuss the applications to deblurring and to electrical impedance tomography.

3.1 Background on Variational Autoencoders and Manifolds

In this section, we revisit the technical backgrounds of the chapter. First, we recall the concept of VAEs and their training procedure. Afterwards, we present our model for the latent distribution based on a nomalizing flow, a short literature review on manifold learning with VAEs and some basic definition from differential geometry.

3.1.1 Variational Autoencoders for Manifold Learning

In this chapter, we assume that we are given data points $x_1, \dots, x_N \in \mathbb{R}^n$ for a large dimension n. In order to reduce the computational effort and to regularize inverse problems, we assume that these data-points are located in a lower-dimensional manifold. We aim to learn the underlying manifold from the data points $x_1, \dots, x_N$ with a VAE ([134, 135]).

A VAE aims to approximate the underlying high-dimensional probability distribution P_X of the random variable X with a lower-dimensional latent random variable $Z \sim P_Z$ on $\mathbb{R}^d$ with $d < n$, by using the data points $x_1, \dots, x_N$. To this end, we define a decoder

[1] https://github.com/johertrich/Manifold_Mixture_VAEs

$D\colon \mathbb{R}^d \to \mathbb{R}^n$ and an encoder $E\colon \mathbb{R}^n \to \mathbb{R}^d$. The decoder approximates P_X by the distribution $P_{\tilde{X}}$ of a random variable $\tilde{X} \coloneqq D(Z) + \eta$, where $\eta \sim \mathcal{N}(0, \sigma_x^2 I_n)$. Vice versa, the encoder approximates P_Z from P_X by the distribution $P_{\tilde{Z}}$ of the random variable $\tilde{Z} \coloneqq E(X) + \xi$ with $\xi \sim \mathcal{N}(0, \sigma_z^2 I_d)$. Now, decoder and encoder are trained such that we have $P_X \approx P_{\tilde{X}}$ and $P_Z \approx P_{\tilde{Z}}$. To this end, we aim to maximize the log-likelihood function $\ell(\theta) = \sum_{i=1}^N \log(p_{\tilde{X}}(x_i))$, where θ denotes the parameters D and E depend upon.

The log-density $\log(p_{\tilde{X}}(x))$ induced by the model is called the evidence. However, for VAEs the computation of the evidence is intractable. Therefore, [134] suggest to approximate it by the *evidence lower bound* given by

$$\mathrm{ELBO}(x|\theta) \coloneqq \mathbb{E}_{\xi\sim\mathcal{N}(0,I_d)}[\log(p_Z(E(x)+\sigma_z\xi)) - \tfrac{1}{2\sigma_x^2}\|D(E(x)+\sigma_z\xi) - x\|^2]. \tag{3.1}$$

Indeed, by Jensen's inequality the evidence can be lower-bounded by

$$\begin{aligned}
\log(p_{\tilde{X}}(x)) &= \log\Big(\int_{\mathbb{R}^d} p_{Z,\tilde{X}}(z,x)\,\mathrm{d}z\Big)\\
&= \log\Big(\int_{\mathbb{R}^d} \frac{p_{Z,\tilde{X}}(z,x)}{p_{\tilde{Z}|X=x}(z)} p_{\tilde{Z}|X=x}(z)\,\mathrm{d}z\Big)\\
&\geq \int_{\mathbb{R}^d} \log\Big(\frac{p_{Z,\tilde{X}}(z,x)}{p_{\tilde{Z}|X=x}(z)}\Big) p_{\tilde{Z}|X=x}(z)\,\mathrm{d}z\Big)\\
&= \mathbb{E}_{z\sim P_{\tilde{Z}|X=x}}\Big[\log\Big(\frac{p_Z(z)p_{\tilde{X}|Z=z}(x)}{p_{\tilde{Z}|X=x}(z)}\Big)\Big]\\
&= \mathbb{E}_{z\sim P_{\tilde{Z}|X=x}}[\log(p_Z(z)) + \log(p_{\tilde{X}|Z=z}(x)) - \log(p_{\tilde{Z}|X=x}(z))].
\end{aligned}$$

Accordingly to the definition of $\tilde{Z}$ and $\tilde{X}$, we have that $p_{\tilde{X}|Z=z}(x) = \mathcal{N}(x; D(z), \sigma_x^2 I_n)$ and $p_{\tilde{Z}|X=x}(z) = \mathcal{N}(z; E(x), \sigma_z^2 I_d)$. Thus, the above formula is, up to a constant, equal to

$$\mathbb{E}_{z\sim P_{\tilde{Z}|X=x}}[\log(p_Z(z)) - \tfrac{1}{2\sigma_x^2}\|x - D(z)\|^2 - \log(\mathcal{N}(z; E(x), \sigma_z^2 I_d))].$$

Considering the substitution $\xi = (z - E(x))/\sigma_z$, we obtain

$$\mathbb{E}_{\xi\sim\mathcal{N}(0,I_d)}[\log(p_Z(E(x)+\sigma_z\xi)) - \tfrac{1}{2\sigma_x^2}\|D(E(x)+\sigma_z\xi) - x\|^2 - \log(\mathcal{N}(\xi; 0, I_d))].$$

Note that also the last summand does not depend on D and E. Thus, we obtain, up to a constant, the ELBO (3.1).

Finally, a VAE is trained by minimizing the loss function which sums up the negative ELBO values of all data points, i.e.,

$$\mathcal{L}_{\mathrm{VAE}}(\theta) = -\sum_{i=1}^N \mathrm{ELBO}(x_i|\theta).$$

Learned Latent Space. It is a known issue of VAEs that the inferred probability distribution is often more blurry than the ground truth distribution of the data. A detailed discussion of this issue can be found in [135, Section 2.8.2]. As a remedy, the authors suggest to choose a more flexible model. One possibility is to combine VAEs with normalizing flows, as proposed in [192] or [68]. Following these approaches, we increase the flexibility of the model by using a latent space learned by a normalizing flow. The idea is based on the observation that transforming probability distributions in low-dimensional spaces is much cheaper than in high-dimensional spaces. Consequently, modelling the low-dimensional latent space can be more effective than learning probability transformations in the high-dimensional data space. Here, we employ the specific loss function from [104, 106] for training the arising model. More precisely, we choose the latent distribution

$$P_Z = \mathcal{T}_\# P_\Xi,$$

where $\mathcal{T}\colon \mathbb{R}^d \to \mathbb{R}^d$ is an invertible neural network, called normalizing flow. In this way, P_Z is the push-forward of a fixed (known) distribution P_Ξ. Then, the density p_Z is given by

$$p_Z(z) = p_\Xi(\mathcal{T}^{-1}(z))|\det(\nabla \mathcal{T}^{-1}(z))|.$$

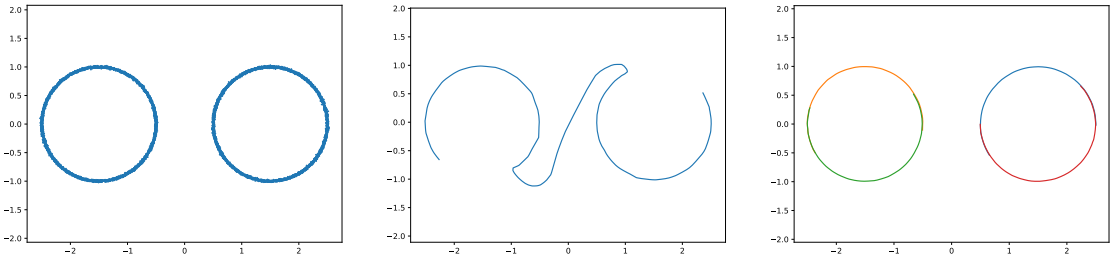
The parameters of $\mathcal{T}$ are considered as trainable parameters. Then, the ELBO reads as

$$\begin{aligned} \mathrm{ELBO}(x|\theta) \coloneqq \mathbb{E}_{\xi\sim\mathcal{N}(0,I_d)}[&\log(p_\Xi(\mathcal{T}^{-1}(E(x)+\sigma_z\xi))) \\ &+ \log(|\det(\nabla \mathcal{T}^{-1}(E(x)+\sigma_z\xi))|) - \tfrac{1}{2\sigma_x^2}\|D(E(x)+\sigma_z\xi)-x\|^2], \end{aligned} \tag{3.2}$$

where θ are the parameters of the decoder, the encoder and of the normalizing flow $\mathcal{T}$.

In the literature, there exist several invertible neural network architectures based on coupling blocks ([75, 132]), residual networks ([33, 57, 115]), ODE representations ([58, 97, 177]) and autoregressive flows ([121]). In our numerics, we use the coupling-based architecture from [22].

Manifold Learning with VAEs. In order to obtain a lower-dimensional representation of the data points, some papers propose to approximate the data-manifold by $\mathcal{M} \coloneqq \{D(z) : z \in \mathbb{R}^d\}$, see e.g. [12, 56, 77, 91]. However, this is only possible if the data-manifold admits a global parameterization, i.e., it can be approximated by one generating function. This assumption is often violated in practice. As a toy example, consider the one-dimensional manifold embedded in $\mathbb{R}^2$ that consists of two circles, see Figure 3.1a. This manifold is disconnected and contains "holes". Consequently, the topologies of the manifold and of the latent space $\mathbb{R}$ do not coincide, so that the manifold cannot be approximated by a VAE. Indeed, this can be verified numerically. When we learn a VAE for approximating samples from this manifold, we observe that the two (generated) circles are not closed and that both components are connected, see Figure 3.1b. As a remedy, in the next section, we propose the use of multiple generators to resolve this problem, see Figure 3.1c. For this purpose, we need the notion of charts and atlases.



(a) Noisy samples from the manifold.

(b) Approximation with one chart.

(c) Approximation with four charts.

Figure 3.1: Example of a one-dimensional manifold that admits no global parameterization.

3.1.2 Embedded Manifolds

A subset $\mathcal{M} \subseteq \mathbb{R}^n$ is called a d-dimensional embedded differentiable manifold if there exist a set of indices I and a family $(U_k, \varphi_k)_{k \in I}$ of relatively open sets $U_k \subseteq \mathcal{M}$ with $\bigcup_{k \in I} U_k = \mathcal{M}$ and mappings $\varphi_k \colon U_k \to \mathbb{R}^d$ such that it holds for every $k, l \in I$ that

- φ_k is a homeomorphism between U_k and $\varphi_k(U_k)$;
- the inverse $\varphi_k^{-1} \colon \varphi_k(U_k) \to U_k$ is continuously differentiable;
- the transition maps $\varphi_k \circ \varphi_l^{-1} \colon \varphi_l(U_k \cap U_l) \to \varphi_k(U_k \cap U_l)$ are continously differentiable;
- and the Jacobian $\nabla \varphi_k^{-1}(x)$ of φ_k^{-1} at x has full column-rank for any $x \in \varphi_k(U_k)$.

We call the mappings φ_k charts and the family $(U_k, \varphi_k)_{k \in I}$ an atlas. With an abuse of notation, we sometimes also call the set U_k or the pair (U_k, φ_k) a chart. Every compact manifold admits an atlas with finitely many charts $(U_k, \varphi_k)_{k=1}^K$, by definition of compactness.

3.2 Chart Learning by Mixtures of VAEs

In order to approximate (embedded) manifolds with arbitrary (unknown) topology, we propose to learn several local parameterizations of the manifold instead of a global one. To this end, we propose to use mixture models of VAEs.

An Atlas as Mixture of VAEs. We propose to learn the atlas of an embedded manifold $\mathcal{M}$ by representing it as a mixture model of variational autoencoders with decoders $D_k \colon \mathbb{R}^d \to \mathbb{R}^n$, encoders $E_k \colon \mathbb{R}^n \to \mathbb{R}^d$ and normalizing flows $\mathcal{T}_k$ in the latent space, for $k = 1, \ldots, K$. Then, the inverse of each chart φ_k will be represented by $\varphi_k^{-1} = \mathcal{D}_k := D_k \circ \mathcal{T}_k$. Similarly, the chart φ_k itself is represented by the mapping $\mathcal{E}_k := \mathcal{T}_k^{-1} \circ E_k$ restricted to the manifold. Throughout this chapter, we denote the

parameters of $(D_k, E_k, \mathcal{T}_k)$ by θ_k. Now, let $\tilde{X}_k$, $k = 1, \dots, K$, be the random variable generated by the decoder D_k as in the previous section. Then, we approximate the distribution P_X of the noisy samples from the manifold by the random variable $\tilde{X} \coloneqq \tilde{X}_J$, where J is a discrete random variable on $\{1, \dots, K\}$ with $P(J = k) = \alpha_k$ with mixing weights $\alpha_k > 0$ fulfilling $\sum_{k=1}^{K} \alpha_k = 1$.

3.2.1 Training of Mixtures of VAEs

Loss function. Let $x_1, \dots, x_N$ be the noisy training samples. In order to train mixtures of VAEs, we minimize an upper bound of the negative log likelihood function $-\sum_{i=1}^{N} \log(p_{\tilde{X}}(x_i))$. To this end, we employ the law of total probability and the Jensen inequality and we obtain

$$\log(p_{\tilde{X}}(x_i)) \geq \sum_{k=1}^{K} \beta_{ik} \log(p_{\tilde{X}_k}(x_i)),$$

where $\beta_{ik} \coloneqq P(J = k|\tilde{X} = x_i)$ is the probability that the sample x_i was generated by the k-th random variable $\tilde{X}_k$. Using the definition of conditional probabilities, we observe that

$$\beta_{ik} = P(J = k|\tilde{X} = x_i) = \frac{P(J = k)p_{\tilde{X}_k}(x_i)}{\sum_{j=1}^{K} P(J = j)p_{\tilde{X}_j}(x_i)} = \frac{\alpha_k p_{\tilde{X}_k}(x_i)}{\sum_{j=1}^{K} \alpha_j p_{\tilde{X}_j}(x_i)}. \tag{3.3}$$

As the computation of $p_{\tilde{X}_k}$ is intractable, we replace it by the ELBO (3.2), i.e., we approximate β_{ik} by

$$\tilde{\beta}_{ik} = \frac{\alpha_k \exp(\mathrm{ELBO}(x_i|\theta_k))}{\sum_{j=1}^{K} \alpha_j \exp(\mathrm{ELBO}(x_i|\theta_j))}. \tag{3.4}$$

Then, we obtain

$$\ell(x_i|\Theta) = \sum_{k=1}^{K} \frac{\alpha_k \exp(\mathrm{ELBO}(x_i|\theta_k))}{\sum_{j=1}^{K} \alpha_j \exp(\mathrm{ELBO}(x_i|\theta_j))} \mathrm{ELBO}(x_i|\theta_k),$$

that is an lower bound for $\log(p_{\tilde{X}}(x_i))$. By summing up over all i, we obtain an upper bound of the negative log likelihood function by the loss function

$$\mathcal{L}(\Theta) = -\sum_{i=1}^{N} \ell(x_i|\Theta),$$

in order to train the parameters $\Theta = (\theta_k)_{k=1}^{K}$ of the mixture of VAEs. Finally, this loss function is then optimized with a stochastic gradient based optimizer like Adam [131].

Remark 12 (Lipschitz Regularization)**.** In order to represent the local structure of the manifold and to stabilize the training, we would like to avoid that two points that are close in the latent distribution have too large a distance in the data space. This corresponds

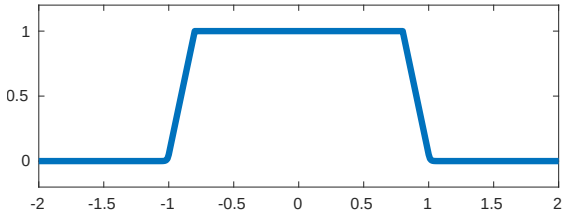


Figure 3.2: Plot of the unnormalized latent density q.

to regularizing the Lipschitz constant of the decoders D_k and of the normalizing flows $\mathcal{T}_k$. More precisely, for some small $\sigma > 0$, we add the regularization term

$$\mathcal{R}(\Theta) \coloneqq \frac{1}{\sigma^2} \sum_{k=1}^{K} \mathbb{E}_{z\sim P_\Xi, \eta\sim\mathcal{N}(0,\sigma^2)} \Big[D_k(\mathcal{T}_k(z)) - D_k(\mathcal{T}_k(z+\eta)) \Big]$$

for the first few epochs of the training. Once the charts roughly capture the local structures of the manifold, we avoid the Lipschitz regularization in order to have the interpretation of the loss function as an approximation of the negative log likelihood of the training points.

Latent Distribution. In order to identify the sets U_k defining the domain of the k-th learned chart, we choose a latent distribution that is mostly concentrated in the rectangle $(-1,1)^d$. Then, we can define the domain U_k of the k-th learned chart as the set $U_k \coloneqq \mathcal{D}_k((-1,1)^d)$. Since the charts are supposed to overlap, the density should become small close to the boundary. To this end, we choose the distribution P_Ξ by using the density $p_\Xi(z) \coloneqq \prod_{i=1}^d q(z_i)$, where the density q is up to a multiplicative constant given by

$$q(z) \propto \begin{cases} 1, & |z| < 0.8, \\ 4.8 - 4.75|z|, & |z| \in [0.8, 1], \\ 0.05 \exp(-100(|z|-1)), & |z| > 1, \end{cases}$$

see Figure 3.2 for a plot.

Due to approximation errors and noise, we will have to deal with points $x \in \mathbb{R}^n$ that are not exactly located in one of the sets U_k. In this case, we cannot be certain to which charts the point x actually belongs. Therefore, we interpret the conditional probability (3.3) as the probability that x_i belongs to the k-th chart. Since we cannot compute the β_{ik} explicitly, we use the approximations $\tilde{\beta}_{ik}$ from (3.4) instead.

Overlapping Charts. Since the sets U_k of an atlas $(U_k, \varphi_k)_{k\in I}$ are an open covering of the manifold, they have to overlap near their boundaries. To this end, we propose the following post-processing heuristic.

By the definition of the loss function $\mathcal{L}$, we have that the k-th generator $\mathcal{D}_k$ is trained such that U_k contains all points x_i from the training set where $\tilde{\beta}_{ik}$ is non-zero. The

following procedure modifies the $\tilde{\beta}_{ik}$ in such a way that the samples x that are close to the boundary of the k-th chart will also be assigned to a second chart.

Since the measure P_Ξ is mostly concentrated in $(-1,1)^d$, the region close to the boundary of the k-th chart can be identified by $D_k(\mathcal{T}_k(z))$ for all z close to the boundary of $(-1,1)^d$. For $c>1$, we define the modified ELBO function

$$\begin{aligned}\mathrm{ELBO}_c(x|\theta_k) \coloneqq \mathbb{E}_{\xi\sim\mathcal{N}(0,I_d)}[&\log(p_\Xi(c\mathcal{T}_k^{-1}(E_k(x)+\sigma_z\xi)))\\ &+\log(|\det(\nabla\mathcal{T}_k^{-1}(E_k(x)+\sigma_z\xi))|)-\tfrac{1}{2\sigma_x^2}\|D_k(E_k(x)+\sigma_z\xi)-x\|^2]\end{aligned}$$

which differs from (3.2) by the additional scaling factor c within the first summand. By construction and by the definiton of p_Ξ, it holds that $\mathrm{ELBO}_c(x,\theta_k)\approx\mathrm{ELBO}(x|\theta_k)$ whenever $c\|\mathcal{T}_k^{-1}(E_k(x))\|_\infty < 0.8$ and $0<\sigma_z\ll 0.1$ is small. Otherwise, we have $\mathrm{ELBO}_c(x|\theta_k)<\mathrm{ELBO}(x|\theta_k)$. In particular, $\mathrm{ELBO}_c(x|\theta_k)$ is close to $\mathrm{ELBO}(x|\theta_k)$ if x belongs to the interior of the k-th chart and is significantly smaller if it is close to the boundary of the k-th chart.

As a variation of $\tilde{\beta}_{ik}$, now we define

$$\hat{\gamma}_{il}^{(l)}=\frac{\alpha_l\exp(\mathrm{ELBO}_c(x_i|\theta_l))}{\alpha_l\exp(\mathrm{ELBO}_c(x_i|\theta_l))+\sum_{j\in\{1,\dots,K\}\setminus\{l\}}\alpha_j\exp(\mathrm{ELBO}(x_i|\theta_j))}$$

and

$$\hat{\gamma}_{ik}^{(l)}=\frac{\alpha_k\exp(\mathrm{ELBO}(x_i|\theta_k))}{\alpha_l\exp(\mathrm{ELBO}_c(x_i|\theta_l))+\sum_{j\in\{1,\dots,K\}\setminus\{l\}}\alpha_j\exp(\mathrm{ELBO}(x_i|\theta_j))}$$

for $k,l\in\{1,\dots,K\}$. Similarly as the $\tilde{\beta}_{ik}$, $\hat{\gamma}_{ik}^{(l)}$ can be viewed as a classification parameter, which assigns each x_i either to a chart $k\neq l$ or to the interior part of the l-th chart. Consequently, points located near the boundary of the l-th chart will also be assigned to another chart. Finally, we combine and normalize the $\hat{\gamma}_{ik}^{(l)}$ by

$$\gamma_{ik}=\frac{\hat{\gamma}_{ik}}{\sum_{k=1}^K\hat{\gamma}_{ik}},\quad\text{where}\quad\hat{\gamma}_{ik}=\max_{l=1,\dots,K}\hat{\gamma}_{ik}^{(l)}. \tag{3.5}$$

Once, the γ_{ik} are computed, we update the mixing weights α by $\alpha_k=\sum_{i=1}^N\gamma_{ik}$ and minimize the loss function

$$\mathcal{L}_{\mathrm{overlap}}(\Theta)=-\sum_{i=1}^N\sum_{k=1}^K\gamma_{ik}\mathrm{ELBO}(x_i|\theta_k) \tag{3.6}$$

using a certain number of epochs of the Adam optimizer [131].

The whole training procedure for a mixture of VAEs representing the charts of an embedded manifold is summarized in Algorithm 1. The hyperparameters M_1, M_2 and M_3 are chosen large enough such that we have approximately approached a local minimum of the corresponding objective function. In our numerical examples, we choose $M_1=50$, $M_2=150$ and $M_3=50$.

Remark 13 (Number of Charts K)**.** The choice of the number of charts K is a trade-off between computational efficiency and flexibility of the model. While each manifold has a minimal number of required charts, it could be easily represented by more charts. Therefore, from a topological viewpoint, there is no upper limit on K. However, since each chart comes with its own pair of decoder and encoder, the number of parameters within the mixture of VAEs, and consequently the training effort, grow with K.

Algorithm 1 Training procedure for mixtures of VAEs.

1. Run the Adam optimizer on $\mathcal{L}(\Theta) + \lambda\mathcal{R}(\Theta)$ for M_1 epochs.
2. Run the Adam optimizer on $\mathcal{L}(\Theta)$ for M_2 epochs.
3. Compute the values γ_{ik}, $i = 1, \dots, N$, $k = 1, \dots, K$ from (3.5).
4. Compute the mixing weights $\alpha_k = \sum_{i=1}^{N} \gamma_{ik}$.
5. Run the Adam optimizer on $\mathcal{L}_{\text{overlap}}(\Theta)$ from (3.6) for M_3 epochs.

3.2.2 Architectures

In this subsection, we focus on the architecture of the VAEs used in the mixture model representing the manifold $\mathcal{M}$. Since $\mathcal{D}_k = D_k \circ \mathcal{T}_k$ represent the inverse of our charts φ_k, the decoders have to be injective. Moreover, since $\mathcal{E}_k = \mathcal{T}_k^{-1} \circ E_k$ represents the chart itself, the condition $\mathcal{E}_k \circ \mathcal{D}_k = \mathrm{Id}$ must be verified. Therefore, we choose the encoder E_k as a left-inverse of the corresponding decoder D_k. More precisely, we use the decoders of the form

$$D_k = T_L \circ A_L \circ \cdots \circ T_1 \circ A_1,$$

where the $T_l \colon \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$ are invertible neural networks and $A_l \colon \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l}$ are fixed injective linear operators for $l = 1, \dots, L$, $d = d_0 < d_1 < \cdots < d_L = n$. As it is a concatenation of injective mappings, we obtain that D_k is injective. Finally, the corresponding encoder is given by

$$E_k = A_1^\dagger \circ T_1^{-1} \circ \cdots \circ A_L^\dagger \circ T_L^{-1}, \quad A^\dagger = (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}}. \tag{3.7}$$

Then, it holds by construction that $\mathcal{E}_k \circ \mathcal{D}_k = \mathrm{Id}$.

Here, we build the invertible neural networks T_l and the normalizing flows $\mathcal{T}_k$ out of coupling blocks as proposed in [22] based on the real NVP architecture [75]. To this end, we split the input $z \in \mathbb{R}^{d_l}$ into two parts $z = (z_1, z_2) \in \mathbb{R}^{d_l^1} \times \mathbb{R}^{d_l^2}$ with $d_l = d_l^1 + d_l^2$ and define $T_l(z) = (x_1, x_2)$ with

$$x_1 = z_1 \, \mathrm{e}^{s_2(z_2)} + t_2(z_2) \quad \text{and} \quad x_2 = z_2 \, \mathrm{e}^{s_1(x_1)} + t_1(x_1),$$

where $s_1, t_1 \colon \mathbb{R}^{d_l^1} \to \mathbb{R}^{d_l^2}$ and $s_2, t_2 \colon \mathbb{R}^{d_l^2} \to \mathbb{R}^{d_l^1}$ are arbitrary subnetworks (depending on l). Then, the inverse $T_l^{-1}(x_1, x_2)$ can analytically be derived as $z = (z_1, z_2)$ with

$$z_2 = (x_2 - t_1(x_1)) \, \mathrm{e}^{-s_1(x_1)} \quad \text{and} \quad z_1 = (x_1 - t_2(z_2)) \, \mathrm{e}^{-s_2(z_2)}.$$

Remark 14 (Projection onto learned charts)**.** Consider a decoder D_k and an encoder E_k as defined above. By construction, the mapping $\pi_k = D_k \circ E_k$ is a (nonlinear) projection onto $\operatorname{range}(D_k) = \operatorname{range}(\pi_k)$, in the sense that $\pi_k \circ \pi_k = \pi_k$ and that $\pi_{k|\operatorname{range}(D_k)}$ is the identity on $\operatorname{range}(D_k)$. Consequently, the mapping π_k is a projection on the range of D_k which represents the k-th chart of $\mathcal{M}$. In particular, there is an open neighborhood $V \coloneqq \pi_k^{-1}(U_k) \subseteq \mathbb{R}^n$ such that $\pi_{k|V}$ is a projection onto U_k.

3.3 Optimization on Learned Manifolds

As motivated in the introduction, we are interested in optimization problems of the form

$$\min_{x \in \mathbb{R}^n} F(x) \quad \text{subject to} \quad x \in \mathcal{M}, \tag{3.8}$$

where $F\colon \mathbb{R}^n \to \mathbb{R}$ is a differentiable function and $\mathcal{M}$ is available only through some data points. In the previous section, we proposed a way to represent the manifold $\mathcal{M}$ by a mixture model $(D_k, E_k, \mathcal{T}_k)$ of VAEs. This section outlines a gradient descent algorithm for the solution of (3.8) once the manifold is learned.

As outlined in the previous section, the inverse charts φ_k^{-1} of the manifold $\mathcal{M}$ are modeled by $\mathcal{D}_k \coloneqq D_k \circ \mathcal{T}_k$. The chart φ_k itself is given by the mapping $\mathcal{E}_k \coloneqq \mathcal{T}_k^{-1} \circ E_k$ restricted to the manifold. For the special case of a VAE with a single generator $\mathcal{D}$, the authors of [12, 56, 77] propose to solve (3.8) in the latent space. More precisely, starting with a latent initialization $z_0 \in \mathbb{R}^d$ they propose to solve

$$\min_{z \in \mathbb{R}^d} F(\mathcal{D}(z))$$

using a gradient descent scheme. However, when using multiple charts, such a gradient descent scheme heavily depends on the current chart. Indeed, the following example shows that the gradient direction can change significantly, if we use a different chart.

Example 10. Consider the two-dimensional manifold $\mathbb{R}^2$ and the two learned charts given by the generators

$$\mathcal{D}_1(z_1, z_2) = (10z_1, z_2), \quad \text{and} \quad \mathcal{D}_2(z_1, z_2) = (z_1, 10z_2).$$

Moreover let $F\colon \mathbb{R}^2 \to \mathbb{R}$ be given by $(x, y) \mapsto x + y$. Now, the point $x^{(0)} = (0, 0)$ corresponds for both charts to $z^{(0)} = (0, 0)$. A gradient descent step with respect to $F \circ \mathcal{D}_k$, $k = 1, 2$, using step size τ yields the latent values

$$\begin{aligned}(z_1^{(1)}, z_2^{(1)}) &= z^{(0)} - \tau \nabla (F \circ \mathcal{D}_1)(z^{(0)}) = -(10\tau, \tau),\\ (\tilde{z}_1^{(1)}, \tilde{z}_2^{(1)}) &= z^{(0)} - \tau \nabla (F \circ \mathcal{D}_2)(z^{(0)}) = -(\tau, 10\tau).\end{aligned}$$

Thus, one gradient steps with respect to $F \circ \mathcal{D}_k$ yields the values

$$x^{(1)} = \mathcal{D}_1(z^{(1)}) = -(100\tau, \tau), \quad \tilde{x}^{(1)} = \mathcal{D}_2(\tilde{z}^{(1)}) = -(\tau, 100\tau).$$

Consequently, the gradient descent steps with respect to two different charts can point into completely different directions, independently of the step size τ.

Therefore, we aim to use a gradient formulation which is independent of the parameterization of the manifold. Here, we use the concept of the Riemannian gradient with respect to the Riemannian metric, which is inherited from the Euclidean space in which the manifold $\mathcal{M}$ is embedded. To this end, we first revisit some basic facts about Riemannian gradients on embedded manifolds which can be found, e.g., in [1]. Afterwards, we consider suitable retractions in order to perform a descent step in the direction of the negative Riemannian gradient. Finally, we use these notions in order to derive a gradient descent procedure on a manifold given by mixtures of VAEs.

3.3.1 Background on Riemannian Optimization

Riemannian Gradients on Embedded Manifolds. Let $x \in \mathcal{M} \subseteq \mathbb{R}^n$ be a point on the manifold, let $\varphi\colon U \to \mathbb{R}^d$ be a chart with $x \in U$ and $\varphi^{-1}\colon \varphi(U) \to U$ be its inverse. Then, the tangent space is given by the set of all directions $\dot\gamma(0)$ of differentiable curves $\gamma\colon (-\varepsilon, \varepsilon) \to \mathcal{M}$ with $\gamma(0) = x$. More precisely, it is given by the linear subspace of $\mathbb{R}^n$ defined as

$$T_x\mathcal{M} = \{Jy : y \in \mathbb{R}^d\}, \quad \text{where } J \coloneqq \nabla\varphi^{-1}(\varphi(x)) \in \mathbb{R}^{n\times d}. \tag{3.9}$$

The tangent space inherits the Riemannian metric from $\mathbb{R}^n$, i.e., we equip the tangent space with the inner product

$$\langle u, v\rangle_x = u^{\mathrm{T}}v, \quad u, v \in T_x\mathcal{M}.$$

A function $f\colon \mathcal{M} \to \mathbb{R}^m$ is called differentiable if for any differentiable curve $\gamma\colon (-\varepsilon, \varepsilon) \to \mathcal{M}$ we have that $f \circ \gamma\colon (-\varepsilon, \varepsilon) \to \mathbb{R}^m$ is differentiable. In this case the differential of f is defined by

$$Df(x)\colon T_x\mathcal{M} \to \mathbb{R}^m, \qquad Df(x)[h] = \frac{\mathrm{d}}{\mathrm{d}t} f(\gamma_h(t))\Big|_{t=0},$$

where $\gamma_h\colon (-\varepsilon, \varepsilon) \to \mathcal{M}$ is a curve with $\gamma_h(0) = x$ and $\dot\gamma_h(0) = h$. Finally, the Riemannian gradient of a differentiable function $f\colon \mathcal{M} \to \mathbb{R}$ is given by the unique element $\nabla_{\mathcal{M}} f \in T_x\mathcal{M}$ which fulfills

$$Df(x)[h] = \langle \nabla_{\mathcal{M}} f, h\rangle_x \quad \text{for all} \quad h \in T_x\mathcal{M}.$$

Remark 15. In the case that f can be extended to a differentiable function on a neighborhood of $\mathcal{M}$, these formulas can be simplified. More precisely, we have that the differential is given by $Df(x)[h] = h^{\mathrm{T}}\nabla f(x)$, where ∇f is the Euclidean gradient of f. In other words, $Df(x)$ is the Fréchet derivative of f at x restricted to $T_x\mathcal{M}$. Moreover, the Riemannian gradient coincides with the orthogonal projection of ∇f on the tangent space, i.e.,

$$\nabla_{\mathcal{M}} f(x) = P_{T_x\mathcal{M}}(\nabla f(x)), \quad P_{T_x\mathcal{M}}(y) = \operatorname*{arg\,min}_{z\in T_x\mathcal{M}} \|y - z\|^2.$$

Here the orthogonal projection can be rewritten as $P_{T_x\mathcal{M}} = J(J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}}$, $J = \nabla\varphi^{-1}(\varphi(x))$ such that the Riemannian gradient is given by $\nabla_{\mathcal{M}} f(x) = J(J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}}\nabla f(x)$.

Retractions. Once the Riemannian gradient is computed, we aim to perform a descent step in the direction of $-\nabla_{\mathcal{M}} f(x)$ on $\mathcal{M}$. To this end, we need the concept of retraction. Roughly speaking, a retraction in x maps a tangent vector ξ to the point that is reached by moving from x in the direction ξ. Formally, it is defined as follows.

Definition 2. A differentiable mapping $R_x \colon V_x \to \mathcal{M}$ for some neighborhood $V_x \subseteq T_x\mathcal{M}$ of 0 is called a retraction in x, if $R_x(0) = x$ and

$$DR_x(0)[h] = h \quad \text{for all} \quad h \in T_0(V_x) = T_x\mathcal{M},$$

where we identified $T_0(V_x)$ with $T_x\mathcal{M}$. Moreover, a differentiable mapping $R = (R_x)_{x\in\mathcal{M}} \colon V \to \mathcal{M}$ on a subset of the tangent bundle $V = (V_x)_{x\in\mathcal{M}} \subseteq T\mathcal{M} = (T_x\mathcal{M})_{x\in\mathcal{M}}$ is a retraction on $\mathcal{M}$, if for all $x \in \mathcal{M}$ we have that R_x is a retraction in x.

Now, let $R \colon V \to \mathcal{M}$ be a retraction on $\mathcal{M}$. Then, the Riemannian gradient descent scheme starting at $x_0 \in \mathcal{M}$ with step size $\tau > 0$ is defined by

$$x_{t+1} = R_{x_t}(-\tau \nabla_{\mathcal{M}} f(x_t)).$$

3.3.2 Retractions for Learned Charts

In order to apply this gradient scheme for a learned manifold given by the learned mappings $(\mathcal{D}_k, \mathcal{E}_k)_{k=1}^K$, we consider two types of retractions. We introduce them and show that they are indeed retractions in the following lemmas. The first one generalizes the idea from [2, Lemma 4, Proposition 5] of moving along the tangent vector in $\mathbb{R}^n$ and reprojecting onto the manifold. However, the results from [2] are based on the orthogonal projection, which is hard or even impossible to compute. Thus, we replace it by some more general projection π. In our applications, π will be chosen as in Remark 14, i.e., we set $\pi(x) = \mathcal{D}_k(\mathcal{E}_k(x))$.

Lemma 11. *Let $x \in \mathcal{M}$, $U_x \subseteq \mathbb{R}^n$ be a neighborhood of x in $\mathbb{R}^n$, $\pi \colon U_x \to \mathcal{M} \cap U_x$ be a differentiable map such that $\pi \circ \pi = \pi$. Set $V_x = \{h \in T_x\mathcal{M} \subseteq \mathbb{R}^n : x + h \in U_x\}$. Then*

$$R_x(h) = \pi(x + h), \qquad h \in V_x,$$

defines a retraction in x.

Proof. The property $R_x(0) = x$ is directly clear from the definition of R_x. Now let $h \in T_x\mathcal{M} \subseteq \mathbb{R}^n$ and $\gamma_h \colon (-\varepsilon, \varepsilon) \to \mathcal{M}$ be a differentiable curve with $\gamma_h(0) = x$ and $\dot{\gamma}_h(0) = h$. As $\pi_{|U}$ is the identity on $\mathcal{M}$, we have by the chain rule that

$$h = \dot{\gamma}_h(t) = \frac{\mathrm{d}}{\mathrm{d}t}\pi(\gamma_h(t))\Big|_{t=0} = \nabla\pi(x)\dot{\gamma}_h(0) = \nabla\pi(x)h,$$

where $\nabla\pi(x)$ is the Euclidean Jacobian matrix of π at x. Similarly,

$$DR_x(0)[h] = \frac{\mathrm{d}}{\mathrm{d}t}R_x(th)\Big|_{t=0} = \frac{\mathrm{d}}{\mathrm{d}t}\pi(x + th)\Big|_{t=0} = \nabla\pi(x)h = h.$$

□

The second retraction uses the idea of changing to local coordinates, moving into the gradient direction by using the local coordinates and then going back to the manifold representation. Note that similar constructions are considered in [1, Section 4.1.3]. However, as we did not find an explicit proof for the lemma, we give it below for the sake of completeness.

Lemma 12. *Let $x \in \mathcal{M}$ and denote by $\varphi\colon U \to \mathbb{R}^d$ a chart with $x \in U \subseteq \mathcal{M}$. Then,*

$$R_x(h) = \varphi^{-1}(\varphi(x) + (J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}}h), \quad J = \nabla\varphi^{-1}(\varphi(x))$$

defines a retraction in x.

Proof. The property $R_x(0) = x$ is directly clear from the definition of R_x. Now let $h \in T_0(T_x\mathcal{M}) = T_x\mathcal{M} \subseteq \mathbb{R}^n$. By (3.9), we have that there exists some $y \in \mathbb{R}^d$ such that $h = Jy$. Then, we have by the chain rule that

$$DR_x(0)[h] = \frac{\mathrm{d}}{\mathrm{d}t}R_x(th)\Big|_{t=0} = (\nabla\varphi^{-1}(\varphi(x)))(J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}}h = J(J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}}Jy = Jy = h.$$

□

3.3.3 Gradient Descent on Learned Manifolds

By Lemma 11 and 12, we obtain that the mappings

$$R_{k,x}(h) = \mathcal{D}_k(\mathcal{E}_k(x+h)) \quad \text{and} \quad \tilde{R}_{k,x}(h) = \mathcal{D}_k(\mathcal{E}_k(x) + (J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}}h) \tag{3.10}$$

with $J = \nabla\mathcal{D}_k(\mathcal{E}_k(x))$ are retractions in all $x \in U_k$. If we define R such that R_x is given by R_k for some k such that $x \in U_k$, then the differentiability of $R = (R_x)_{x\in\mathcal{M}}$ in x might be violated. Moreover, the charts learned by a mixture of VAEs only overlap approximately and not exactly. Therefore, these retractions cannot be extended to a retraction on the whole manifold $\mathcal{M}$ in general. As a remedy, we propose the following gradient descent step on a learned manifold.

Starting from a point x_n, we first compute for $k = 1, \dots, K$ the probability, that x_n belongs to the k-th chart. By (3.3), this probability can be approximated by

$$\beta_k \coloneqq \frac{\alpha_k \exp(\mathrm{ELBO}(x_n|\theta_k))}{\sum_{j=1}^K \alpha_j \exp(\mathrm{ELBO}(x_n|\theta_j))}. \tag{3.11}$$

Afterwards, we project x_n onto the k-th chart by applying $\tilde{x}_{n,k} = \mathcal{D}_k(\mathcal{E}_k(x_n))$ (see Remark 14) and compute the Riemannian gradient $g_{n,k} = \nabla_{\mathcal{M}}F(\tilde{x}_{n_k})$. Then, we apply the retraction $R_{k,\tilde{x}_{n,k}}$ (or $\tilde{R}_{k,\tilde{x}_{n,k}}$) to perform a gradient descent step $x_{n+1,k} = R_{k,\tilde{x}_{n,k}}(-\tau_n g_{n,k})$. Finally, we average the results by $x_{n+1} = \sum_{k=1}^K \beta_k x_{n+1,k}$.

The whole gradient descent step is summarized in Algorithm 2. Finally, we compute the sequence $(x_n)_n$ by applying Algorithm 2 iteratively.

For some applications the evaluation of the derivative of F is computationally costly. Therefore, we aim to take as large step sizes τ_n as possible in Algorithm 2. On the other hand, large step sizes can lead to numerical instabilities and divergence. To this end, we use an adaptive step size selection as outlined in Algorithm 3.

Algorithm 2 One gradient descent step on a learned manifold.

Inputs: Function $F\colon \mathcal{M} \to \mathbb{R}$, point x_n, step size $\tau_n > 0$.
for $k = 1, \dots, K$ **do**
- Approximate the probability that x_n belongs to chart k by computing the β_k from (3.11).
- Project to the k-th chart by $\tilde{x}_{n,k} = \mathcal{D}_k(\mathcal{E}_k(x_n))$.
- Compute the Riemannian gradient $g_{n,k} = \nabla_{\mathcal{M}} F(\tilde{x}_{n,k})$, e.g., by Remark 15.
- Perform a gradient descent with the retraction $R_{k,\tilde{x}_{n,k}}$, i.e., define $x_{n+1,k} = R_{k,\tilde{x}_{n,k}}(-\tau_n g_{n,k})$.

end for
- Average results by computing $x_{n+1} = \sum_{k=1}^{K} \beta_k x_{n+1,k}$.

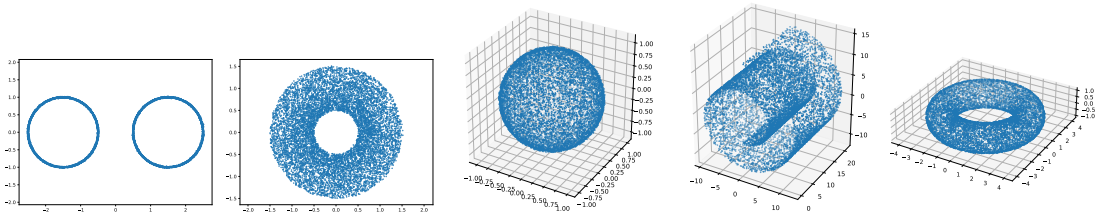
Algorithm 3 Adaptive step size scheme for gradient descent on learned manifolds

Input: Function F, initial point x_0, initial step size τ_0.
for n=0,1,... **do**
  Compute x_{n+1} by Algorithm 2 with step size τ_n.
  while $F(x_{n+1}) > F(x_n)$ **do**
    Update step size by $\tau_n \leftarrow \frac{\tau_n}{2}$.
    Update x_{n+1} by Algorithm 2 with the new step size τ_n.
  end while
  Set step size for the next step $\tau_{n+1} = \frac{3\tau_n}{2}$.
end for

Remark 16 (Descent Algorithm)**.** By construction Algorithm 3 is a descent algorithm. That is, for a sequence $(x_n)_n$ generated by the algorithm it holds that $F(x_{n+1}) \leq F(x_n)$. With the additional assumption that F is bounded from below, we have that $(F(x_n))_n$ is a bounded descending and hence convergent sequence. However, this does neither imply convergence of the iterates $(x_n)_n$ themselves nor optimality of the limit of $(F(x_n))_n$. For more details on the convergence of line-search algorithms on manifolds we refer to [1, Section 4].

3.4 Numerical Examples

Next, we test the numerical performance of the proposed method. In this section, we start with some one- and two-dimensional manifolds embedded in the two- or three-dimensional Euclidean space. We use the architecture from Section 3.2.2 with $L = 1$. That is, for all the manifolds, the decoder is given by $\mathcal{T} \circ A$ where $A\colon \mathbb{R}^d \to \mathbb{R}^n$ is given by $x \mapsto (x, 0)$ if $d = n - 1$ and by $A = \mathrm{Id}$ if $d = n$ and $\mathcal{T}$ is an invertible neural network with 5 invertible coupling blocks where the subnetworks have two hidden layers and 64 neurons in each layer. The normalizing flow modeling the latent space consists of 3 invertible coupling blocks with the same architecture. We train the mixture of VAEs for 200 epochs with



Two circles Ring Sphere Swiss roll Torus

Figure 3.3: Datasets used for the different manifolds.

the Adam optimizer. Afterwards we apply the overlapping procedure for 50 epochs, as in Algorithm 1.

We consider the manifolds "two circles", "ring", "sphere", "swiss roll" and "torus". The (noisy) training data are visualized in Figure 3.3. The number of charts K is given in the following table.

	Two circles	Ring	Sphere	Swiss roll	Torus
Number of charts	4	2	2	4	6

We visualize the learned charts in Figure 3.4. Moreover, additional samples generated by the learned mixture of VAEs are shown in Figure 3.5. We observe that our model covers all considered manifolds and provides a reasonable approximation of different charts. Finally, we test the gradient descent method from Algorithm 2 with some linear and quadratic functions, which often appear as data fidelity terms in inverse problems:

- $F(x) = x_2$ on the manifold "two circles" with initial points $\frac{x_0}{\|x_0\|} \pm (1.5, 0)$ for $x_0 = (\pm 0.2, 1)$;
- $F(x) = \|x - (-1, 0)\|^2$ on the manifold "ring" with initial points $(1, \pm 0.4)$;
- $F(x) = \|x - (0, 0, -2)\|^2$ on the manifold "sphere" with inital points $x_0/\|x_0\|$ for $x_0 \in \{(0.3\cos(\frac{\pi k}{5}), 0.3\sin(\frac{\pi k}{5}), 1) : k = 0, \dots, 9)\}$;
- $F(x) = \|x - (-5, 0, 0)\|^2$ on the manifold "torus", where the inital points are drawn randomly from the training set.

We use the retraction from Lemma 11 with a step size of 0.01. The resulting trajectories are visualized in Figure 3.6. We observe that all the trajectories behave as expected and approach the closest minimum of the objective function, even if this is not in the same chart of the initial point.

Remark 17 (Dimension of the Manifold)**.** For all our numerical experiments, we assume that the dimension d of the data manifold is known. This assumption might be violated for practical applications. However, there exist several methods in the literature to estimate the dimension of a manifold from data, see e.g., [31, 52, 82, 152]. We are aware that dimension estimation of high dimensional datasets is a hard problem which cannot

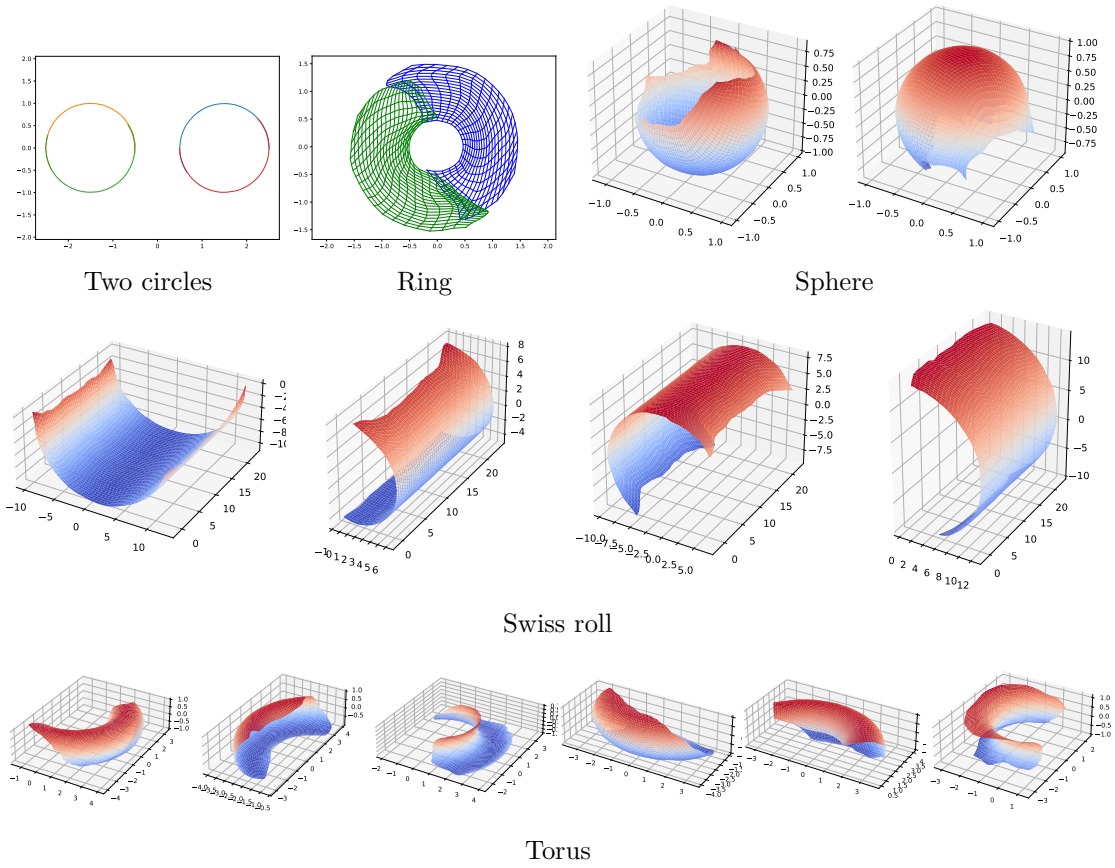


Figure 3.4: Learned charts for the different manifolds. For the manifolds "two circles" and "ring", each color represents one chart. For the manifolds "sphere", "swiss roll" and "torus" we plot each chart in a separate figure.

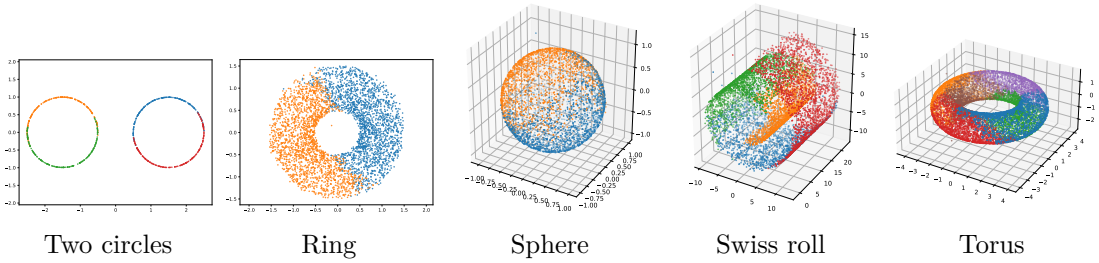
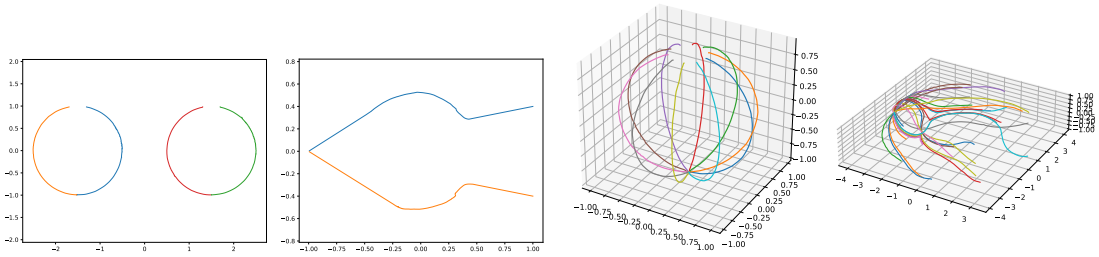


Figure 3.5: Generated samples by the learned mixture of VAEs. The color of a point indicates from which generator the point was sampled.



Two circles Ring Sphere Torus

Figure 3.6: Trajectories of the gradient descent on the learned manifolds.

be considered as completely solved so far. In particular, most of these algorithms make assumptions on the distribution of the given data points. Even though it is an active area of research, it is not the scope of this work to test, benchmark or develop such algorithms. Similarly, combining them with our mixture of VAEs is left for future research.

3.5 Mixture of VAEs for Inverse Problems

In this section we describe how to use mixture of VAEs to solve inverse problems. We consider an inverse problem of the form

$$y = \mathcal{G}(x) + \eta, \tag{3.12}$$

where $\mathcal{G}$ is a possibly nonlinear map between $\mathbb{R}^n$ and $\mathbb{R}^m$, modeling a measurement (forward) operator, $x \in \mathbb{R}^n$ is a quantity to be recovered, $y \in \mathbb{R}^m$ is the noisy data and η represents some noise. In particular, we analyze a linear and a nonlinear inverse problem: a deblurring problem and a parameter identification problem for an elliptic PDE arising in electrical impedance tomography (EIT), respectively.

In many inverse problems, the unknown x can be modeled as an element of a low-dimensional manifold $\mathcal{M}$ in $\mathbb{R}^n$ [12, 24, 44, 122, 176, 204, 172, 5], and this manifold can be represented by the mixture of VAEs as explained in Section 3.2. Thus, the solution of (3.12) can be found by optimizing the function

$$F(x) = \frac{1}{2}\|\mathcal{G}(x) - y\|^2_{\mathbb{R}^m} \quad \text{subject to } x \in \mathcal{M}, \tag{3.13}$$

by using the iterative scheme proposed in Section 3.3.

We would like to emphasize that the main goal of our experiments is not to obtain state-of-the-art results. Instead, we want to highlight the advantages of using multiple generators via a mixture of VAEs. All our experiments are designed in such a way that the manifold property of the data is directly clear. The application to real-world data and the combination with other methods in order to achieve competitive results are not within the scope of this work and are left to future research.

Architecture and Training. Throughout these experiments we consider images of size 128×128 and use the architecture from Section 3.2.2 with $L = 3$. Starting with the latent dimension d, the mapping $A_1 \colon \mathbb{R}^d \to \mathbb{R}^{32^2}$ fills up the input vector x with zeros up to the size 32^2, i.e., we set $A_1(x) = (x, 0)$. The invertible neural network $T_1 \colon \mathbb{R}^{32^2} \to \mathbb{R}^{32^2}$ consists of 3 invertible blocks, where the subnetworks s_i and t_i, $i = 1, 2$ are dense feed-forward networks with two hidden layers and 64 neurons. Afterwards, we reorder the dimensions to obtain an image of size 32×32. The mappings $A_2 \colon \mathbb{R}^{32\times32} \to \mathbb{R}^{32\times32\times4}$ and $A_3 \colon \mathbb{R}^{64\times64} \to \mathbb{R}^{64\times64\times4}$ copy each channel 4 times. Then, the generalized inverses $A_2^\dagger \colon \mathbb{R}^{32\times32\times4} \to \mathbb{R}^{32\times32}$ and $A_3^\dagger \colon \mathbb{R}^{64\times64\times4} \to \mathbb{R}^{64\times64}$ from (3.7) are given by taking the mean of the four channels of the input. The invertible neural networks $T_2 \colon \mathbb{R}^{32\times32\times4} \to \mathbb{R}^{64\times64}$ and $T_3 \colon \mathbb{R}^{64\times64\times4} \to \mathbb{R}^{128\times128}$ consist of 3 invertible blocks, where the subnetworks s_i and t_i, $i = 1, 2$ are convolutional neural networks with one hidden layer and 64 channels. After these three coupling blocks we use an invertible upsampling [80] to obtain the correct output dimension. For the normalizing flow in the latent space, we use an invertible neural network with three blocks, where the subnetworks s_i and t_i, $i = 1, 2$ are dense feed-forward networks with two hidden layers and 64 neurons.

We train all the models for 200 epochs with the Adam optimizer. Afterwards we apply the overlapping procedure for 50 epochs. See Algorithm 1 for the details of the training algorithm.

3.5.1 Deblurring

First, we consider the inverse problem of noisy image deblurring. Here, the forward operator $\mathcal{G}$ in (3.12) is linear and given by the convolution with a 30×30 Gaussian blur kernel with standard deviation 15. In order to obtain outputs y of the same size as the input x, we use constant padding with intensity $1/2$ within the convolution. Moreover, the image is corrupted by white Gaussian noise η with standard deviation 0.1. Given an observation y generated by this degradation process, we aim to reconstruct the unknown ground truth image x.

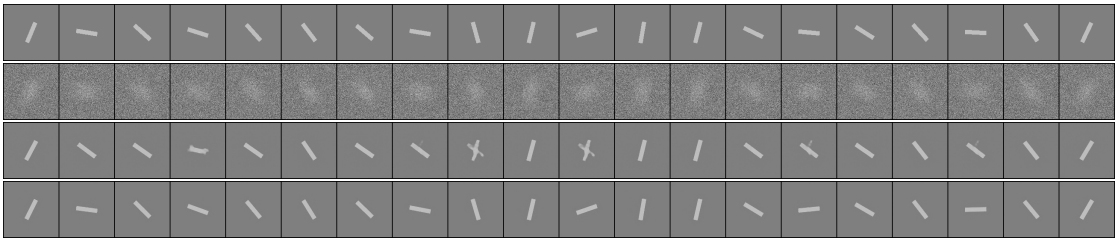
Dataset and Manifold Approximation. Here, we consider the dataset of 128×128 images showing a bright bar with a gray background which is centered and rotated. The intensity of fore- and background as well as the size of the bar are fixed. Some example images from the dataset are given in Figure 3.7a. The dataset forms a one-dimensional manifold parameterized by the rotation of the bar. Therefore, it is homeomorphic to S^1 and does not admit a global parameterization since it contains a hole.

We approximate the data manifold by a mixture model of two VAEs and compare the result with the approximation with a single VAE, where the latent dimension is set to $d = 1$. The learned charts are visualized in Figure 3.7b and 3.7c. We observe that the charts learned with a mixture of two VAEs can generate all possible relations and overlap at their boundaries. On the other hand the chart learned with a single VAE does not cover all rotations but has a gap due to the injectivity of the decoder. This gap is

(a) Samples from the considered dataset for the deblurring example.

(b) Learned chart with one generator. The figure shows the images $D(x)$ for 20 values of x equispaced in $[-1, 1]$.

(c) Learned charts with two generators. The figure shows the images $D_k(x)$ for 20 values of x equispaced in $[-1, 1]$ for $k = 1$ (top) and $k = 2$ (bottom).

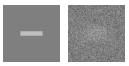


(d) Reconstructions for the deblurring example. From top to bottom: ground truth image, observation, reconstruction with one generator and reconstruction with two generators.

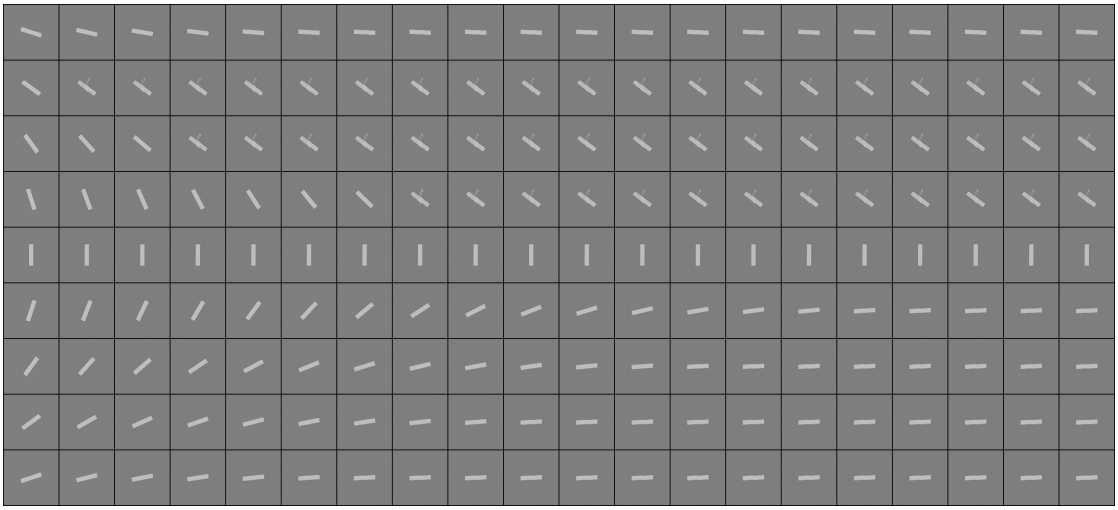
Figure 3.7: Dataset, learned charts and reconstructions for the deblurring example.

also represented in the final test loss of the model, which approximates the negative log likelihood of the test data. It is given by 52.04 for one generator and by 39.84 for two generators. Consequently, the model with two generators fits the data manifold much better.

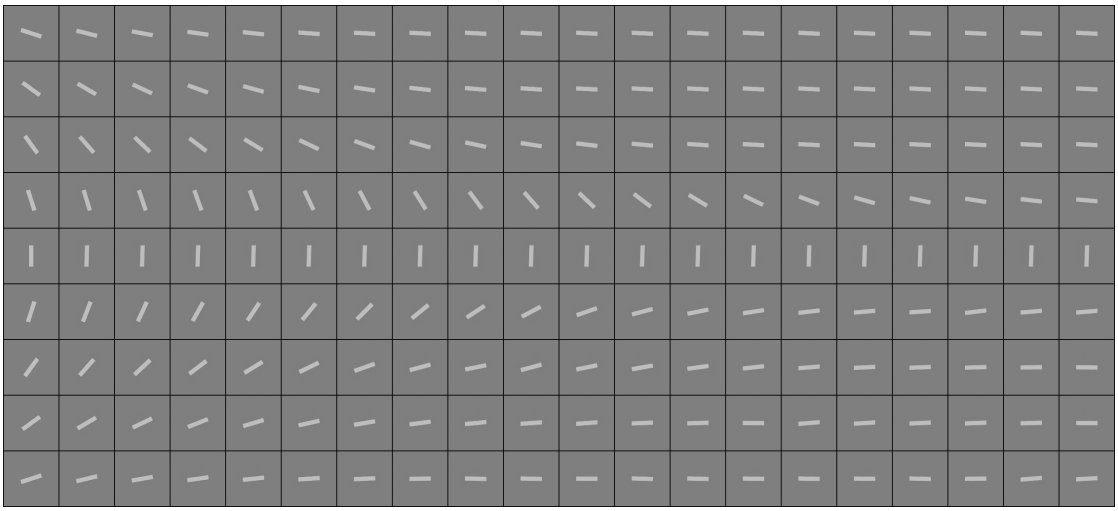
Reconstruction. In order to reconstruct the ground truth image, we use our gradient descent scheme for the function (3.13) as outlined in Algorithm 2 for 500 iterations. Since the function F is defined on the whole $\mathbb{R}^{128\times128}$, we compute the Riemannian gradient $\nabla_{\mathcal{M}} F(x)$ accordingly to Remark 15. More precisely, for $x \in U_k$, we have $\nabla_{\mathcal{M}} F(x) = J(J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}}\nabla F(x)$, where $\nabla F(x)$ is the Euclidean gradient of F and $J = \nabla \mathcal{D}_k(\mathcal{E}_k(x))$ is the Jacobian of the k-th decoder evaluated at $\mathcal{E}_k(x)$. Here, the Euclidean gradient $\nabla F(x)$ and the Jacobian matrix J are computed by algorithmic differentiation. Moreover, we use the retractions $\tilde{R}_{k,x}$ from (3.10). As initialization x_0 of our gradient descent scheme, we use a random sample from the mixture of VAEs. The results are visualized in Figure 3.7d. We observe that the reconstructions with two generators always recover the ground truth images very well. On the other hand, the reconstructions with one generator often are unrealistic and do not match with the ground truth. These unrealistic images may appear when the gradient descent scheme, starting from a random initialization, has to pass through points that are not covered by



(a) Ground truth (left) and observation (right).

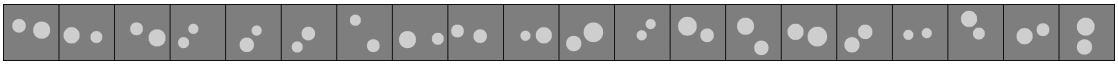


(b) Visualization of the trajectories $(x_n)_n$ for different initializations x_0 with one generator. Left column: initialization, right column: reconstruction x_{250}, columns in between: images x_n for n approximately equispaced between 0 and 250.

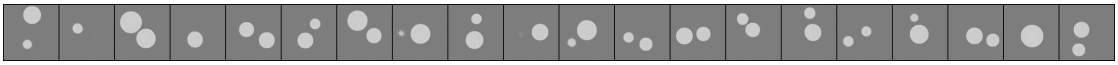


(c) Visualization of the trajectories $(x_n)_n$ for different initializations x_0 with two generators. Left column: initialization, right column: reconstruction x_{250}, columns in between: images x_n for n approximately equispaced between 0 and 250.

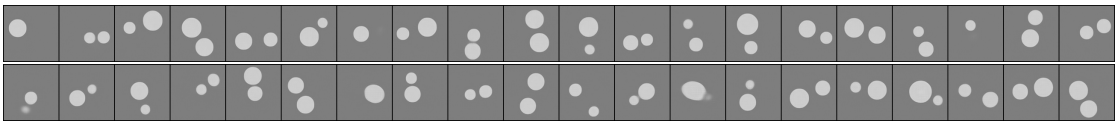
Figure 3.8: Gradient descent for the deblurring example.



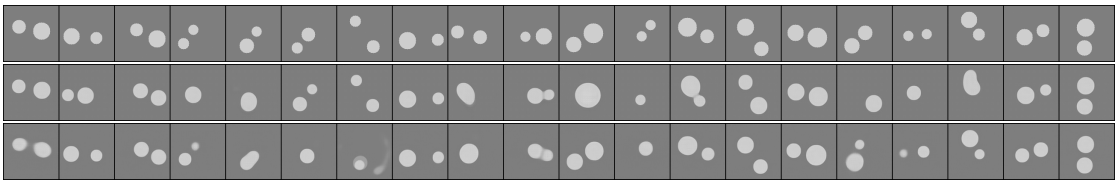
(a) Samples from the considered dataset for the EIT example.



(b) Samples from the VAE with one generator.



(c) Samples from the mixture of two VAEs. Top: first chart, bottom: second chart.



(d) Reconstructions. From top to bottom: ground truth image, reconstruction with one generator, reconstruction with two generators.

Figure 3.9: Dataset, synthesized samples and reconstructions for the EIT example.

the single chart parameterizing the manifold.

In order to better understand why the reconstructions with one generator often fail, we consider the trajectories $(x_n)_n$ generated by Algorithm 2 more in detail. We consider a fixed ground truth image showing a horizontal bar and a corresponding observation as given in Figure 3.8a. Then, we run Algorithm 2 for different initializations. The results are given in Figure 3.8b for one generator and in Figure 3.8c for two generators. The left column shows the initialization x_0, and in the right column, there are the values x_{250} after 250 gradient descent steps. The columns in between show the values x_n for (approximately) equispaced n between 0 and 250. With two generators, the trajectory $(x_n)_n$ are a smooth transition from the initialization to the ground truth. Only when the initialization is a vertical bar (middle row), the images x_n remain similar to the initialization x_0 for all n, since this is a critical point of the $F|_{\mathcal{M}}$ and hence the Riemannian gradient is zero. With one generator, we observe that some of the trajectories get stuck exactly at the gap, where the manifold is not covered by the chart. At this point the latent representation of the corresponding image would have to jump, which is not possible. Therefore, a second generator is required at this point.

3.5.2 Electrical Impedance Tomography

Finally, we consider the highly nonlinear and ill-posed inverse problem of electrical impedance tomography [62], which is also known in the mathematical literature as the Calderón problem [26, 84, 174]. EIT is a non-invasive, radiation-free method to measure the conductivity of a tissue through electrodes placed on the surface of the

body. More precisely, electrical currents patterns are imposed on some of these electrodes and the resulting voltage differences are measured on the remaining ones. Although harmless, the use of this modality in practice is very limited because the standard reconstruction methods provide images with very low spatial resolution. This is an immediate consequence of the severe ill-posedness of the inverse problem [13, 168].

Classical methods for solving this inverse problem include variational-type methods [63], the Lagrangian method [61], the factorization method [46, 137], the D-bar method [209], the enclosure method [124], and the monotonicity method [216]. Similarly as many other inverse problems, deep learning methods have had a big impact on EIT. For example, the authors of [83] propose an end-to-end neural network that learns the forward map $\mathcal{G}$ and its inverse. Moreover, deep learning approaches can be combined with classical methods, e.g, by post processing methods [109, 108] or by variational learning algorithms [204].

Dataset and Manifold Approximation. We consider the manifold consisting of 128×128 images showing two bright non-overlapping balls with a gray background, representing conductivities with special inclusions. The radius and the position of the balls vary, while the fore- and background intensities are fixed. Some exemplary samples of the dataset are given in Figure 3.9a.

Remark 18 (Dimension Topology of the Data Manifold)**.** Since the balls are indistinguishable and not allowed to overlap, an image can be uniquely described by the angle between the two balls, the midpoint between the both balls, their distance and the two radii. Hence, the data manifold is homeomorphic to $\mathbb{S}^1 \times (0,1)^2 \times (0,1) \times (0,1)^2 = \mathbb{S}^1 \times (0,1)^5$. In particular, it contains a hole and does not admit a global parameterization.

A slightly more general version of this manifold was considered in [5], where Lipschitz stability is proven for a related inverse boundary value problem restricted to the manifold. Other types of inclusions (with unknown locations), notably polygonal and polyhedral inclusions, have been considered in the literature ([38, 36, 25]). The case of small inclusions is discussed in [18].

We approximate the data manifold by a mixture of two VAEs and compare the results with the approximation with a single VAE. The latent dimension is set to the manifold dimension, i.e., $d = 6$. Some samples of the learned charts are given in Figure 3.9b and 3.9c. As in the previous example, both models produce mostly realistic samples. The test loss is given by 365.21 for one generator and by 229.99 for two generators. Since the test loss approximates the negative log likelihood value of the test data, this indicates that the two generators are needed in order to cover the whole data manifold.

The Forward Operator and its Derivative. From a mathematical viewpoint, EIT considers the following PDE with Neumann boundary conditions

$$\begin{cases} -\nabla \cdot (\gamma \, \nabla u_g) = 0 & \text{in } \Omega, \\ \gamma \, \partial_\nu u_g = g & \text{on } \partial\Omega, \end{cases} \tag{3.14}$$

where $\Omega \subseteq \mathbb{R}^2$ is a bounded domain, $\gamma \in L^\infty(\Omega)$ is such that $\gamma(x) \geq \lambda > 0$ and $u_g \in H^1(\Omega)$ is the unique weak solution with zero boundary mean of (3.14) with Neumann boundary data $g \in H_\diamond^{-\frac{1}{2}}(\partial\Omega)$, with $H_\diamond^s(\partial\Omega) = \{f \in H^s(\partial\Omega) : \int_{\partial\Omega} f ds = 0\}$. From the physical point of view, g represents the electric current applied on $\partial\Omega$ (through electrodes placed on the boundary of the body), u_g is the electric potential and γ is the conductivity of the body in the whole domain Ω. The inverse problem consists in the reconstruction of γ from the knowledge of all pairs of boundary measurements $(g, u_g|_{\partial\Omega})$, namely, of all injected currents together with the corresponding electric voltages generated at the boundary. In a compact form, the measurements may be modelled by the Neumann-to-Dirichlet map

$$\begin{aligned} \mathcal{G}(\gamma)\colon H_\diamond^{-\frac{1}{2}}(\partial\Omega) &\to H_\diamond^{\frac{1}{2}}(\partial\Omega) \\ g &\mapsto u_g\big|_{\partial\Omega}. \end{aligned}$$

Since the PDE (3.14) is linear, the map $\mathcal{G}(\gamma)$ is linear. However, the forward map $\gamma \mapsto \mathcal{G}(\gamma)$ is nonlinear in γ, and so is the corresponding inverse problem. The map $\mathcal{G}$ is continuously differentiable, and its Fréchet derivative (see [113]) is given by $[\nabla\mathcal{G}(\gamma)](\sigma)(g) = w_g|_{\partial\Omega}$, where $w_g \in H^1(\Omega)$ is the unique weak solution with zero boundary mean of

$$\begin{cases} -\nabla \cdot (\gamma\, \nabla w_g) = \nabla \cdot (\sigma\, \nabla u_g) & \text{in } \Omega, \\ -\gamma\, \partial_\nu w_g = \sigma\, \partial_\nu u_g & \text{on } \partial\Omega, \end{cases}$$

where $u_g \in H^1(\Omega)$ is the unique weak solution with zero boundary mean that solves (3.14). We included the expression of this derivative in the continuous setting for completeness, but, as a matter of fact, we will need only its semi-discrete counterpart given below.

Discretization and Objective Function. In our implementations, we discretize the linear mappings $G(\gamma)$ by restricting them to a finite dimensional subspace spanned by a-priori fixed boundary functions $g_1, \ldots, g_N \in H_\diamond^{-\frac{1}{2}}(\partial\Omega)$. Then, following [39, eqt. (2.2)], we reconstruct the conductivity by minimizing the semi-discrete functional

$$F(\gamma) = \frac{1}{2} \sum_{n=1}^{N} \int_{\partial\Omega} |u_{g_n}(s) - (u_{\text{true}})_{g_n}(s)|^2 ds, \tag{3.15}$$

where $(u_{\text{true}})_{g_n}$ is the observed data. In our discrete setting, we represent the conductivity γ by a piecewise constant function $\gamma = \sum_{m=1}^{M} \gamma_m \mathbb{1}_{T_m}$ on a triangulation $(T_m)_{m=1,\ldots,M}$. Then, following [39, eqt. (2.20)], the derivative of (3.15) with respect to γ is given by

$$\frac{dF}{d\gamma_m}(\gamma) = \sum_{n=1}^{N} \int_{T_m} \nabla u_{g_n}(x) \cdot \nabla z_{g_n}(x) dx, \tag{3.16}$$

where z_{g_n} solves

$$\begin{cases} -\nabla \cdot (\gamma\, \nabla z_{g_n}) = 0 & \text{in } \Omega, \\ \gamma\, \partial_\nu z_{g_n} = (u_{\text{true}})_{g_n} - u_{g_n} & \text{on } \partial\Omega, \end{cases} \tag{3.17}$$

with the normalization $\int_{\partial\Omega} z_{g_n}(s) ds = \int_{\partial\Omega} (u_{\text{true}})_{g_n}(s) ds$.

Implementation Details. In our experiments the domain Ω is given by the unit square $[0,1]^2$. For solving the PDEs (3.14) and (3.17), we use a finite element solver from the DOLFIN library [159]. We employ meshes that are coarser in the middle of Ω and finer close to the boundary. To simulate the approximation error of the meshes, and to avoid inverse crimes, we use a fine mesh to generate the observation and a coarser one for the reconstructions. We use $N = 15$ boundary functions, which are chosen as follows. We divide each of the four edges of the unit square $[0,1]^2$ into 4 segments and denote by $b_1, \dots, b_{16}$ the functions that are equal to 1 on one of these segments and 0 otherwise. Then, we define the boundary functions as $g_n = \sum_{i=1}^{16} a_{n,i} b_i$, where the matrix $A = (a_{n,i})_{n=1,\dots,15, i=1,\dots,16}$ is the 16×16 Haar matrix without the first row. More precisely, the rows of A are given by the rows of the matrices $2^{-k/2}(\mathrm{Id}_{2^{4-k}} \otimes (1,-1) \otimes e_{2^{k-1}}^{\mathrm{T}})$ for $k = 1, \dots, 4$, where $\otimes$ is the Kronecker product and $e_j \in \mathbb{R}^j$ is the vector where all entries are 1.

Results. We reconstruct the ground truth images from the observations by minimizing the functional F from (3.15) subject to $\gamma \in \mathcal{M}$. To this end, we apply the gradient descent scheme from Algorithm 2 for 100 steps. Since the evaluation of the forward operator and its derivative include the numerical solution of a PDE, it is computationally very costly. Hence, we aim to use as few iterations of Algorithm 2 as possible. To this end, we apply the adaptive step size scheme from Algorithm 3. As retractions we use $\tilde{R}_{k,x}$ from (3.10). The initialization γ_0 of the gradient descent scheme is given by a random sample from the mixture of VAEs.

Since F is defined on the whole $\mathbb{R}_+^{128\times128}$, we use again Remark 15 for the evaluation of the Riemannian gradient. More precisely, for $\gamma \in U_k$, we have that $\nabla_{\mathcal{M}} F(\gamma) = J(J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}}\nabla F(\gamma)$, where $\nabla F(\gamma)$ is the Euclidean gradient of F and $J = \nabla \mathcal{D}_k(\mathcal{E}_k(\gamma))$. Here, we compute $\nabla F(\gamma)$ by (3.16) and J by algorithmic differentiation.

The reconstructions for 20 different ground truths are visualized in Figure 3.9d. We observe that both models capture the ground truth structure in most cases, but also fail sometimes. Nevertheless, the reconstructions with the mixture of two VAEs recover the correct structure more often and more accurately than the single VAE which can be explained by the better coverage of the data manifold. To quantify the difference more in detail, we rerun the experiment with 200 different ground truth images and compare the results with one and two generators using the PSNR and SSIM. As an additional evaluation metric, we run the segmentation algorithm proposed by [178] on the ground truth and reconstruction image and compare the resulting segmentation masks using the SSIM. The resulting values are given in the following table.

	One generator	Two generators
PSNR	23.64 ± 3.91	24.76 ± 3.79
SSIM	0.8951 ± 0.0377	0.9111 ± 0.0368
segment+SSIM	0.8498 ± 0.0626	0.8667 ± 0.0614

Consequently, the reconstructions with two generators are significantly better than those with one generator for all evaluation metrics.

Chapter 4

Learning a Gaussian Mixture for Sparsity Regularization in Inverse Problems

A signal is said to be *sparse* if it can be represented as a linear combination of a small number of vectors in a known family (e.g., a basis or a frame). Sparsity has played a major role in the last decades in signal processing and statistics as a way to identify key quantities and find low dimensional representations of many families of signals, including natural images [54, 76, 166]. In inverse problems, sparse optimization has had a significant impact in many applications, notably in accelerated magnetic resonance imaging via compressed sensing [163].

The key ingredient needed in sparse optimization is the knowledge of a suitable dictionary that can well represent the unknown quantities with as few elements as possible. If these quantities cannot be described analytically and are measured from experiments, given a sufficient number of samples, machine learning techniques can be used to infer the dictionary [149]. Dictionary learning in the context of sparse optimization, i.e. sparse coding, can be seen as the problem of finding the best change of basis that makes the data as sparse as possible. Thus, it is a special case of the problem of learning a regularization functional [162, 203, 7] and the more general framework of learning operators between function spaces (see [142] and references therein).

In this chapter, we propose a new approach for dictionary learning for sparse optimization motivated by the problem of learning a regularization functional for solving inverse problems. We consider a linear inverse problem in a finite-dimensional setting:

$$y = Ax + \varepsilon, \tag{4.1}$$

where $x \in \mathbb{R}^n$, $\varepsilon \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$. We assume a statistical perspective: namely, x and ε are realizations of the random variables X and E on $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. Let ρ_X and ρ_E denote their probability distributions.

Given some partial knowledge about the probability distributions of X and E, we aim to recover an estimator $R\colon \mathbb{R}^m \to \mathbb{R}^n$ that has statistical guarantees and encodes

information on the sparsity of X. While sparsity is well established in a deterministic setting, there is no clear consensus on how to define a probability distribution of sparse vectors. One option comes from the Bayesian interpretation of ℓ^1 minimization: it can be seen as a maximum a posteriori (MAP) estimator under a Laplacian prior in the presence of additive white Gaussian noise. However, this is unsatisfactory if the aim is to generate truly sparse signals. Recently, it has been shown that specific hierarchical Bayesian models can be used as priors for sparse signals [51, 50].

Here, instead, we consider a mixture of degenerate Gaussians as a statistical model for a distribution of sparse vectors X. The dimensions of the degenerate support of each Gaussian naturally represent the sparsity levels of the signals. We also assume that the noise is Gaussian.

To find the estimator R, we consider a statistical learning framework. We employ the mean squared error (MSE, also referred to as expected risk), namely:

$$L(R) = \mathbb{E}_{X\sim\rho_X, E\sim\rho_E}\|R(AX+E) - X\|^2_{\mathbb{R}^m} = \mathbb{E}_{(X,Y)\sim\rho}\|R(Y) - X\|^2_{\mathbb{R}^m}, \tag{4.2}$$

where ρ is the joint probability distribution of (X, Y) on $\mathbb{R}^n \times \mathbb{R}^m$. By minimizing L among all possible measurable functions R, we get the so-called Minimum MSE (MMSE) estimator, or Bayes estimator.

The Bayes estimator cannot be directly employed as a solution to the statistical inverse problem because it requires the knowledge of the full probability distribution ρ. Supervised learning techniques can be used to find a good approximation when a finite amount of data is available. However, a suitable class of functions (hypothesis space) must be chosen to obtain a good approximation.

Our main contribution is the development of supervised and unsupervised learning schemes to approximate the Bayes estimator. This is done thanks to an explicit expression for the MMSE, which is usually not available for general probability distributions underlying the data and the noise. The proposed training schemes are based on the observation that the explicit expression of the MMSE can be seen as a two-layer neural network, which can be easily trained using stochastic optimization and back-propagation. In particular, we find a strong similarity of our network with a single self-attention layer, a building block of the well-known transformer architecture [222]: our results provide a novel statistical interpretation of the attention mechanism.

After the training, our network/algorithm can be used as an alternative to classical sparse optimization approaches. We perform extensive numerical comparisons of our approach with well-known sparsity promoting algorithms: LASSO, group LASSO, iterative hard thresholding (IHT), and sparse coding/dictionary learning. The tests are done by performing denoising and deblurring on several 1D datasets. In all experiments, our method outperforms the others, in the MSE. Our method can be seen as a new paradigm for both sparse optimization and dictionary learning for sparse data.

The main limitation of our strategy is represented by the number of parameters that need to be estimated to identify the optimal regularizer. It is easy to show that this number grows as Ln^2, being L the number of components of the Gaussian mixture model. If the unknown X is simply assumed to be s-sparse, namely, that its support is contained

in any possible $s-$dimensional coordinate subspace in $\mathbb{R}^n$, we should consider $L = \binom{n}{s}$, which is prohibitive in most applications. This is why our method is best suited for cases of *structured* sparsity, in which the support of X consists of a restricted number of subspaces, which is the case of group sparsity [231], for instance.

The chapter is organized as follows. In Section 4.1, we provide an overview of the concept of Bayes estimator and recall an explicit formula in the case of linear inverse problems with Gaussian mixture prior. Afterwards, Section 4.2 is dedicated to the interpretation of this estimator as a neural network, highlighting its potential utility for sparse recovery. We introduce a supervised and an unsupervised algorithm for learning the neural network representation of the Bayes estimator in Section 4.3. In Section 4.4, we describe some baseline sparsity-promoting algorithms, against which our methods are evaluated. This is done in Section 4.5, where we conduct numerical comparisons on denoising and deblurring problems across various 1D datasets.

4.1 Statistical learning and Bayes estimator for inverse problems

In this section, we introduce our approach and motivation, collecting some results that are already known about inverse problems, statistical estimation, and mixtures of Gaussian random variables. The main elements of novelty are reported in the next section.

We recall the inverse problem introduced in (4.1), which we can also interpret as a realization of the following equation involving the random variables X on $\mathbb{R}^n$ and E, Y on $\mathbb{R}^m$:

$$Y = AX + E. \tag{4.3}$$

Throughout the chapter, we make the following assumptions on the random variables X and E.

Assumption 13. We assume that:

- The distribution of the noise, ρ_E, is known and zero-mean, i.e. $\mathbb{E}[E] = \mathbf{0}$. The covariance Σ_E is invertible.
- The random variables X and E are independent.

Our goal is to identify a function $R\colon \mathbb{R}^m \to \mathbb{R}^n$ (which we will refer to as a regularizer or an estimator) such that the reconstructions $R(Ax + \varepsilon)$ are close to the corresponding x, when x, ε are sampled from X, E and the squared error $\|R(Ax + \varepsilon) - R(x)\|^2$ is used as a metric.

This problem is in general different from the deterministic formulation of inverse problems: the recovery of a single x from $y = Ax + \varepsilon$. The task we are considering also differs from the one of Bayesian inverse problems. In that scenario, indeed, the goal is not to retrieve a function R (or a regularized solution $R(Y)$), but rather to determine a probability distribution, in particular the conditional distribution of $X|Y$, and to ensure

that, as the noise E converges to $\mathbf{0}$ in some suitable sense, this (posterior) distribution converges to the distribution of X.

Our perspective on the inverse problem is instead rather related to the statistical theory of estimation, or statistical inference. Indeed, based on some partial knowledge about the probability distributions of X and E, we aim to recover an estimate $R(Y)$ of X close to X w.r.t. the MSE. In particular, we set our discussion at the intersection of two areas of statistical inference, namely the Minimum Mean Square Error estimation and parametric estimation.

The first field is determined by the choice of (4.2) as the metric according to which the random variables $R(Y)$ and X are considered close to each other or not. The minimizer of L among all possible measurable functions R is defined as the MMSE estimator, or Bayes estimator:

$$R^\star \in \arg\min\{L(R) : \ R\colon \mathbb{R}^m \to \mathbb{R}^n, R \text{ measurable}\}.$$

As it is easy to show (see [66]), the solution to such a problem is the conditional mean of X given Y, so that

$$R^\star(y) = \mathbb{E}[X|Y = y] = \int_{\mathbb{R}^n} x\, p(x|y)\, dx.$$

The Bayes estimator $R^\star$ is not always the most straightforward choice. Indeed, its exact computation requires the knowledge of the joint probability distribution ρ, which depends on the distributions of the noise E and of X. Notice that the distribution of X is not a user-crafted prior but should encode the full statistical model of the ground truth X, which may not be fully accessible when solving the inverse problem.

To overcome this issue, one usually assumes to have access to a rather large set of training data, namely of pairs (x_j, y_j) sampled from the joint distribution ρ. It is then possible to substitute the integrals appearing in the definition of L and of $R^\star$ by means of Monte-Carlo quadrature rules. This approach is usually referred to as *supervised statistical learning*. In order to avoid overfitting and preserve stability, one typically restricts the choice of the possible estimators R to a specific class of functions $\mathcal{H}$, which introduces an implicit bias on the estimator and possibly a regularizing effect. This leads us to the framework of parametric estimation.

In [7], for example, the hypothesis class $\mathcal{H}$ consists of a family of affine functions in y, solutions of suitably parametrized quadratic minimization problems. Minimizing over such a class (which corresponds to the task of learning the optimal generalized Tikhonov regularizer) is motivated by theoretical reasons: if the distributions of X and E are Gaussian, then $R^\star$ belongs to $\mathcal{H}$. However, in general the minimizer of L within $\mathcal{H}$ would deviate from $R^\star$, and such a bias could be in some cases undesired. On top of that, a significant outcome of the analysis in [7] is a closed-form expression of the optimal regularizer (also in infinite dimension), which in particular can be computed if the mean and the covariance of X are known. This paves the way to an *unsupervised statistical learning* approach: namely, to approximate such an estimator one would not need a training set $(x_j, y_j)_j$ sampled from the probability distribution ρ, but only a set $\{x_j\}$ sampled from ρ_X.

This raises the question if there exist other possible prior distributions X that may lead to a simple choice of a hypothesis class $\mathcal{H}$ containing the corresponding Bayes estimators, possibly endowed with an unsupervised technique to approximate them. If we consider a variational approach, in which $\mathcal{H}$ is a set of solution maps of suitable minimization problems, [98] showed that the conditional mean $R^\star$, i.e. the MMSE estimator, can always be represented as the minimizer of a functional Φ_{MMSE}, related to the probability distribution p_X. Nevertheless, in most cases, such a functional is not convex, which makes it critical to define $\mathcal{H}$ and to solve a minimization problem in it.

In this chapter, without considering a variational point of view, we propose a strategy through which it is possible to associate a hypothesis class to (rather general) prior distributions of X. The most prominent outcome of this approach is the thorough treatment of the case in which X is a mixture of Gaussian random variables, which may be employed as a model of sparsity.

Our approach is based on the idea of parametric estimation in statistical inference: the distribution of the exact solution X is unknown, but belongs to a known class of distributions, parametrized by a suitable set of parameters:

$$\{\rho_X(\theta) : \ \theta \in \Theta\},$$

for some $\Theta \subseteq \mathbb{R}^p$. For example, we can assume that X is a Gaussian random variable of unknown mean μ and covariance Σ, the pair (μ, Σ) being the parameter θ. Analogously, we can describe more complicated probability distributions, possibly employing a larger set of parameters.

In this context, it is easy to construct a hypothesis class $\mathcal{H}$ that contains the Bayes estimators $R^\star$ of every possible prior in the parametric class, namely,

$$\mathcal{H} = \{R_\theta = \mathbb{E}_{\rho(\theta)}[X|Y = \cdot] : \theta \in \Theta\},$$

where $\rho(\theta)$ is the joint probability distribution for (X, Y) when $X \sim \rho_X(\theta)$, $Y = AX + E$ and $E \sim \rho_E$. Recall that, by Assumption 13, the distribution of the noise E is known, and so it is considered fixed.

Notice that this same strategy can also be employed for nonlinear inverse problems: we formulated it in this narrower context for the ease of notation, since the content of the next sections is only related to linear problems.

Despite this approach being rather general, it is useful only if it is possible to provide a closed-form expression of R_θ, which might entail a learning approach for its approximation. This can be easily done when X belongs to fairly simple classes, such as Gaussian random variables or, as we show in this section, mixtures of Gaussians.

4.1.1 Bayes estimator for linear inverse problems with a Gaussian Mixture prior

We first recall some basic definitions on mixtures of random variables. Then, in Theorem 14 we show the formula of the Bayes estimator for linear inverse problems with a Gaussian mixture prior.

Definition 3. A random variable X in $\mathbb{R}^n$ is a *mixture of random variables* if it can be written as

$$X = \sum_{i=1}^{L} X_i \mathbb{1}_{\{i\}}(I),$$

where X_i are random variables in $\mathbb{R}^n$, I is a random variable on $\{1, ..., L\}$ independent of X_i and $L \in \mathbb{N}^+$ is the number of elements in the mixture. The indicator function $\mathbb{1}_{\{i\}}(I)$ is equal to 1 when $I = i$ and 0 otherwise: as a consequence, the role of the discrete random variable I is selecting the random variable X_i, therefore $w_i := \mathbb{P}(I = i)$ are informally called the *weights of the mixture*.

Definition 4. A random variable X in $\mathbb{R}^n$ is a *Gaussian mixture* if it is a mixture, as defined in Definition 3, and $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ are Gaussian random variables in $\mathbb{R}^n$.

Theorem 14. *Let X be a Gaussian mixture in $\mathbb{R}^n$, as in Definition 4, and $E \sim \mathcal{N}(\mathbf{0}, \Sigma_E)$ be independent of X_i for every i, and of I. Let $A \in \mathbb{R}^{m\times n}$ be such that $A\Sigma_i A^T + \Sigma_E$ is invertible for every $i = 1, \dots, L$. Set $Y = AX + E$, as in (4.3). The corresponding Bayes estimator is*

$$R^\star(y) = \mathbb{E}[X|Y = y] = \sum_{i=1}^{L} \frac{c_i}{\sum_{j=1}^{L} c_j}(\mu_i + \Sigma_i A^T (A\Sigma_i A^T + \Sigma_E)^{-1}(y - A\mu_i)), \tag{4.4}$$

where

$$c_i = \frac{w_i}{\sqrt{(2\pi)^n |A\Sigma_i A^T + \Sigma_E|}} \exp\Big(-\frac{1}{2}\|(A\Sigma_i A^T + \Sigma_E)^{-\frac{1}{2}}(y - A\mu_i)\|_2^2\Big), \tag{4.5}$$

with the notation $|B| = \det B$.

A proof of Theorem 14 can be found in [143]. For completeness, we provide a similar proof below.The proof follows easily using the next two lemmas. The first lemma provides the formula of the Bayes estimator for the denoising problem with a random variable that comes from a general mixture distribution. The second one establishes an explicit formula in the case of Gaussian mixture distributions.

Lemma 15. *Suppose that Assumption 13 holds true. Consider the statistical linear inverse problem (4.3), where x is sampled from a mixture of random variables, as in Definition 3, and $A \in \mathbb{R}^{m\times n}$. Let $Y_i = AX_i + E$, for $i = 1, \dots, L$. Then*

1. *the density of Y is $p_Y(y) = \sum_{i=1}^{L} w_i p_{Y_i}(y)$, where $w_i = \mathbb{P}(I = i)$;*

2. $\mathbb{E}[X|Y = y] = \sum_{i=1}^{L} \frac{w_i p_{Y_i}(y)}{p_Y(y)} \mathbb{E}[X_i|Y_i = y]$.

Proof. We prove the two parts separately.

Proof of 1. The random variable Y can be written as

$$Y = A\Big(\sum_{i=1}^{L} X_i \mathbb{1}_{\{i\}}(I)\Big) + E = \Big(\sum_{i=1}^{L} AX_i \mathbb{1}_{\{i\}}(I)\Big) + \Big(\sum_{i=1}^{L} E\mathbb{1}_{\{i\}}(I)\Big) = \sum_{i=1}^{L} Y_i \mathbb{1}_{\{i\}}(I).$$

Since $Y_i = AX_i + E$ and $X_i, E \perp I$ for every $i = 1, ..., L$, then $Y_i \perp I$ for every $i = 1, ..., L$. Therefore, the density of Y becomes

$$p_Y(y) = \sum_{i=1}^{L} w_i p_{Y_i}(y),$$

recalling that $w_i = \mathbb{P}(I = i)$.

Proof of 2. Since X_i and E are independent, their joint density is $p_{X_i,E}(x, \varepsilon) = p_{X_i}(x)p_E(\varepsilon)$. Then, using the change of variable $\Phi(X_i, E) = (X_i, AX_i + E) = (X_i, Y_i)$, we have

$$\begin{aligned} p_{X_i,Y_i}(x, y) &= p_{\Phi(X_i,E)}(\Phi(x, \varepsilon)) \\ &= p_{X_i,E}(x, \varepsilon)|J_{\Phi^{-1}}| \\ &= p_{X_i,E}(x, \varepsilon) \\ &= p_{X_i}(x)p_E(y - Ax), \end{aligned}$$

since $\Phi^{-1}(X_i, Y_i) = (X_i, Y_i - AX_i)$ and

$$J_{\Phi^{-1}} = \begin{bmatrix} I & \mathbf{0} \\ -A & I \end{bmatrix}.$$

The same argument holds using X instead of X_i, then

$$p_{X,Y}(x, y) = p_X(x)p_E(y - Ax).$$

Moreover, since $X_i \perp I$ for every $i = 1, ..., L$, $p_X(x) = \sum_{i=1}^{L} w_i p_{X_i}(x)$ with $w_i = \mathbb{P}(I = i)$. Then

$$\begin{aligned} p_{X|Y}(x|y) &= \frac{p_{X,Y}(x, y)}{p_Y(y)} \\ &= \sum_{i=1}^{L} \frac{w_i p_{X_i}(x)p_E(y - Ax)}{p_Y(y)} \\ &= \sum_{i=1}^{L} \frac{w_i p_{Y_i}(y)}{p_Y(y)} \frac{p_{X_i}(x)p_E(y - Ax)}{p_{Y_i}(y)} \\ &= \sum_{i=1}^{L} \frac{w_i p_{Y_i}(y)}{p_Y(y)} \frac{p_{X_i,Y_i}(x, y)}{p_{Y_i}(y)} \\ &= \sum_{i=1}^{L} \frac{w_i p_{Y_i}(y)}{p_Y(y)} p_{X_i|Y_i}(x|y). \end{aligned}$$

Integrating in x, we obtain the result. □

Lemma 16. *Suppose that Assumption 13 holds true. Consider the statistical linear inverse problem (4.3), where x is sampled from a Gaussian mixture, the noise ε is sampled from $E \sim \mathcal{N}(\mathbf{0}, \Sigma_E)$, independent of X_i and I, and $A \in \mathbb{R}^{m\times n}$. Let $Y = AX + E$. Then*

1. *the density of Y_i is*

$$p_{Y_i}(y) = \frac{1}{\sqrt{(2\pi)^n |A\Sigma_i A^T + \Sigma_E|}} \exp\Big(-\frac{1}{2} \|(A\Sigma_i A^T + \Sigma_E)^{-\frac{1}{2}}(y - A\mu_i)\|_2^2 \Big);$$

2. *and $\mathbb{E}[X_i|Y_i = y] = \mu_i + \Sigma_i A^T (A\Sigma_i A^T + \Sigma_E)^{-1}(y - A\mu_i)$.*

Proof. We prove the two parts separately.

Proof of 1: Since $X_i \perp E$, $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ and $E \sim \mathcal{N}(\mathbf{0}, \Sigma_E)$, we have

$$Y_i = AX_i + E \sim \mathcal{N}(A\mu_i, A\Sigma_i A^T + \Sigma_E).$$

The expression for the density of Y_i immediately follows.

Proof of 2: Since $Y_i = AX_i + E$ and $X_i \perp E$, we have

$$\begin{aligned}(X_i, Y_i) &\sim \mathcal{N}\left(\begin{bmatrix} \mathbb{E}[X_i] \\ \mathbb{E}[Y_i] \end{bmatrix}, \begin{bmatrix} \mathrm{Var}[X_i] & \mathrm{Cov}[X_i, Y_i] \\ \mathrm{Cov}[Y_i, X_i] & \mathrm{Var}[Y_i] \end{bmatrix} \right) \\ &= \mathcal{N}\left(\begin{bmatrix} \mu_i \\ A\mu_i \end{bmatrix}, \begin{bmatrix} \Sigma_i & \Sigma_i A^T \\ A\Sigma_i & A\Sigma_i A^T + \Sigma_E \end{bmatrix} \right).\end{aligned}$$

Then, using [205, Theorem 2, Section 13], the conditional distribution $(X_i|Y_i = y)$ is Gaussian and its expectation is given by

$$\mathbb{E}[X_i|Y_i = y] = \mu_i + \Sigma_i A^T (A\Sigma_i A^T + \Sigma_E)^{-1}(y - A\mu_i). \qquad \square$$

4.2 A neural network for sparse recovery

In this section, we provide a novel interpretation both of the resulting formula (4.4) and of the Gaussian mixture model itself, in view of an application to sparsity-promoting learned regularization. We start by showing that the expression derived in (4.4) can be understood as a neural network, whose architecture has strong connections with the well-known attention mechanism used in transformers [222]. Then, we describe how the Gaussian mixture model assumption can be used to encode a (group) sparsity prior on the unknown random variable X.

4.2.1 Bayes estimator as a neural network

We wish to interpret the expression of the Bayes estimator (4.4) for Gaussian mixture models as a neural network from $\mathbb{R}^m$ to $\mathbb{R}^n$.

We start by identifying the parameters: in particular, we define

$$\theta = (\{w_i\}_{i=1}^L, \{\mu_i\}_{i=1}^L, \{\Sigma_i\}_{i=1}^L), \tag{4.6}$$

collecting all the weights, means, and covariances of the mixture. We can now denote the function defined in (4.4) as R_θ, parametrized according to (4.6). Note that the forward map A and the covariance of the noise Σ_E are considered known and fixed, and we assume that Σ_E is invertible, so that $A\Sigma_i A^T + \Sigma_E$ is invertible for every i. The resulting hypothesis class is

$$\mathcal{H} = \{R_\theta : \ \theta \in \Theta\}; \quad \Theta \subseteq [0,1]^L \times (\mathbb{R}^n)^L \times (\mathbb{R}^{n\times n})^L. \tag{4.7}$$

In order for R_θ to be well defined and for the parameters to have a precise statistical interpretation, we impose that

$$\Theta = \{\theta \text{ defined in (4.6)} : \sum_{i=1}^L w_i = 1 \text{ and } \Sigma_i \succcurlyeq 0 \text{ symmetric for } i = 1, ..., L\},$$

where $\Sigma_i \succcurlyeq 0$ denotes that Σ_i is positive semidefinite. Notice that the set $\mathcal{H}$ is the collection of the Bayes estimators of all possible Gaussian mixture models of L components in $\mathbb{R}^n$.

We now focus on describing formula (4.4) in terms of a neural network's architecture (see Figure 4.1):

$$R_\theta(y) = \sum_{i=1}^L W_i t_i, \tag{4.8}$$

where

$$W_i = \text{softmax}(z)_i := \frac{e^{z_i}}{\sum_{j=1}^L e^{z_j}},$$

with

$$z_i = f_i(y) := \log\left(\frac{w_i}{\sqrt{(2\pi)^n |A\Sigma_i A^T + \Sigma_E|}}\right) - \frac{1}{2}\|(A\Sigma_i A^T + \Sigma_E)^{-\frac{1}{2}}(y - A\mu_i)\|_2^2, \tag{4.9}$$

and

$$t_i = g_i(y) := \mu_i + \Sigma_i A^T (A\Sigma_i A^T + \Sigma_E)^{-1}(y - A\mu_i). \tag{4.10}$$

Note that c_i in (4.5) is $e^{f_i(y)}$ and that $g_i \colon \mathbb{R}^m \to \mathbb{R}^n$ is an affine map in y, whereas $f_i \colon \mathbb{R}^m \to \mathbb{R}$ is a *generalized quadratic function* [169], namely

$$f_i(y) = \gamma_i + b_i^T y + y^T A_i y$$

for some $\gamma_i \in \mathbb{R}$, $b_i \in \mathbb{R}^m$, $A_i \in \mathbb{R}^{m\times m}$. Second-order functions have already been used as neural network layers [89].

Adopting the vocabulary of machine learning practitioners, R_θ is a two-layer feed-forward neural network, with a single hidden layer that involves non-standard operations on the input variables. Such a layer has some connections with an attention mechanism with L channels, as discussed in the next paragraph.

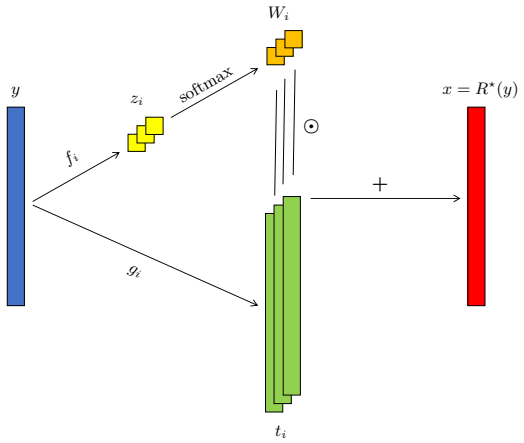


Figure 4.1: Architecture of the neural network representing the Bayes estimator for denoising with $L = 3$.

Connections with the attention mechanism.

The transformer architecture, based on the attention mechanism [222], has revolutionized the field of machine learning, especially natural language processing (NLP), offering a powerful solution for capturing and utilizing relevant information within complex data. Inspired by human attention, this mechanism enables models to selectively focus on the most important elements of the input, significantly enhancing their performance across a wide range of tasks, from NLP to computer vision.

Traditionally, machine learning models treat all input elements equally, neglecting the varying importance of different parts of the data. Attention addresses this limitation by introducing a mechanism that assigns weights or attention scores to different elements. By doing so, models can dynamically allocate their resources to the most relevant components, so their interpretability and performance increase.

More precisely, we consider the self-attention mechanism, which consists of three main components: queries, keys, and values. The queries represent the elements that require attention, while the keys are compared to the queries to determine their relevance. The values correspond to the associated information or representations of the input elements. Attention scores are computed by measuring compatibility or similarity between the queries and the keys, often employing techniques like dot product, additive, or multiplicative attention. Softmax normalization is then used to obtain attention weights. Mathematically, an input $\eta \in \mathbb{R}^{L\times m}$ is linearly transformed into $Q = \eta U_Q$, $K = \eta U_K$ and $V = \eta U_V$, where $U_Q, U_K, U_V \in \mathbb{R}^{m\times m}$ are linear maps. The elements $Q, K, V \in \mathbb{R}^{L\times m}$ represent the L queries, keys, and values of dimension m, respectively. The attention mechanism is then

$$\mathrm{Att}(\eta) = \mathrm{Att}(Q, K, V) := \mathrm{softmax}\left(\frac{QK^T}{\sqrt{L}}\right)V \in \mathbb{R}^{L\times m},$$

where the softmax acts column-wise, namely

$$\text{softmax}(M)_{i,j} = \frac{e^{M_{i,j}}}{\sum_{l=1}^{N_c} e^{M_{i,l}}}, \quad M \in \mathbb{R}^{N_r \times N_c}.$$

Considering as input $\eta \in \mathbb{R}^{L \times m}$ with rows $\eta_i := y - A\mu_i \in \mathbb{R}^m$, $i = 1, ..., L$, we are able to define three affine maps (not linear as in the classical attention mechanism) modeling the queries, the keys and the values. More precisely, the rows of Q and K are, respectively,

$$Q_i = \frac{l_i^{\frac{1}{2}}}{\sqrt{m}} \mathbf{1}_m + \frac{1}{\sqrt{2}} M_i \eta_i \in \mathbb{R}^m$$

and

$$K_i = \frac{l_i^{\frac{1}{2}}}{\sqrt{m}} \mathbf{1}_m - \frac{1}{\sqrt{2}} M_i \eta_i \in \mathbb{R}^m,$$

where $l_i = \log\left(\frac{w_i}{\sqrt{(2\pi)^n |A\Sigma_i A^T + \Sigma_E|}}\right)$, $M_i = (A\Sigma_i A^T + \Sigma_E)^{-\frac{1}{2}}$ and $\mathbf{1}_m = (1, \ldots, 1) \in \mathbb{R}^m$. Then $z_i \in \mathbb{R}$, $i = 1, ..., L$, defined in (4.9) can be rewritten as

$$z = (Q \odot K)\mathbf{1}_m^T \in \mathbb{R}^{L \times 1},$$

where $\odot$ represents the Hadamard product (or element-wise product).

Next, by defining the rows of $V \in \mathbb{R}^{L \times n}$ where n is the dimension of the linear inverse problem unknowns (4.1) using the affine transformation of η given in (4.10), namely

$$V_i = \mu_i + \Sigma_i A^T M_i^2 \eta_i \in \mathbb{R}^n,$$

we obtain

$$R^\star(y) = \overline{\text{Att}}(\eta) = \overline{\text{Att}}(Q, K, V) := \text{softmax}\left((Q \odot K)\mathbf{1}_m^T\right)^T V \in \mathbb{R}^n.$$

Therefore, we have found the queries, the keys and the values to view our network as an alternative version of the attention mechanism. The difference from the classical attention mechanism is the fact that the three transformations for finding Q, K, V are affine, and not linear, and that we use the element-wise product of matrices, instead of the matrix product, which leads to a different size of the network output.

4.2.2 (Degenerate) Gaussian mixture as sparsity prior

The Gaussian mixture prior can be viewed as a sparsity prior by taking X_i in Definition 4 as a degenerate Gaussian. This means that X_i is a Gaussian variable whose support is contained in an s-dimensional subspace of $\mathbb{R}^n$, with $s \ll n$ denoting the sparsity level. In other words, the covariance matrix Σ_i is degenerate, with $\dim(\ker \Sigma_i)^\perp \leq s$. Note that this setting, in which the covariances are not full rank, is compatible with the model

considered so far, and with the corresponding estimator written as a neural network, because Σ_E is invertible.

Let us briefly discuss why this setting corresponds to a sparsity prior. Take $\mu_i = \mathbf{0}$. Write Σ_i with respect to its eigenvectors $\{\varphi_k^i\}_k$ and eigenvalues $\{\sigma_k^i\}_k$:

$$\Sigma_i = \sum_{k=1}^{s} \sigma_k^i \, \varphi_k^i \otimes \varphi_k^i.$$

This allows us to expand X_i as

$$X_i = \sum_{k=1}^{s} a_k^i \varphi_k^i,$$

where $a_k^i \sim \mathcal{N}(0, (\sigma_k^i)^2)$. Therefore, with probability w_i, we have $X = X_i$, and X_i is a random linear combination of $\varphi_1^i, \ldots, \varphi_s^i$, and so is an s-sparse vector.

In the case when the number of elements in the mixture, L, is equal to the number of all possible subsets of cardinality s of $\{1, \ldots, n\}$, $\binom{n}{s}$, and the eigenvectors φ_k^i are all chosen from a fixed orthonormal basis $\mathcal{B}$ of $\mathbb{R}^n$, the mixture generates all vectors that are s-sparse with respect to $\mathcal{B}$. However, $\binom{n}{s}$ grows very fast in s and n, and so this becomes unfeasible even with relatively small values of s and n. Therefore, we are led to take $L \ll \binom{n}{s}$, which corresponds to selecting a priori a subset of all possible s-dimensional subspaces of $\mathbb{R}^n$, a setting that is commonly referred to as group sparsity [231]. On the other hand, we have additional flexibility in the choice of the eigenvectors φ_k^i, which need not be chosen from a fixed basis.

While Gaussian distributions work well to represent smoothness priors, they are not well-adapted to model sparsity. Here, we propose to use (degenerate) Gaussian mixture models to represent (group) sparsity prior. An alternative approach using hierarchical Bayesian models is considered in [51, 50]. We also note that our unsupervised approach (introduced below) shares a similar clustering step with the dictionary learning strategy presented in [43] within a hierarchical model framework.

4.3 Proposed algorithms: Supervised and Unsupervised approaches

In this section, we propose two different algorithms to learn the neural network representation (4.8) of the Bayes estimator (4.4) to solve the statistical linear inverse problem of retrieving X from Y in (4.3). We assume that the random variables X and E satisfy Assumption 13, and write X as a mixture of L random variables

$$X = \sum_{i=1}^{L} X_i \mathbb{1}_{\{i\}}(I),$$

where the weights $w_{X_i} = \mathbb{P}(I = i)$, the means $\mu_{X_i} = \mathbb{E}[X_i]$ and the covariances $\Sigma_{X_i} = \mathrm{Cov}[X_i]$ are in general unknown.

We propose two possible regularizers of the form R_θ (cfr. (4.6) and (4.7)), for two ideal choices of parameters θ. The first one, which we will call *supervised*, is

$$\theta^* \in \operatorname{argmin} \{L(R_\theta) : \theta \in \Theta, \|\theta\|_\infty \leq \varrho\}, \tag{4.11}$$

for some $\varrho > 0$, being $L(R_\theta)$ defined as in (4.2) and $\|\theta\|_\infty$ the ℓ^∞ norm of the vectorized θ. Notice that, since the minimization problem (4.11) is restricted to a closed subset of a compact ball and $L(R_\theta)$ is continuous in θ, the minimum θ^* exists.

The second parameter choice, which corresponds to an *unsupervised* approach, is instead simply

$$\theta_X = (\{w_{X_i}\}_{i=1}^L, \{\mu_{X_i}\}_{i=1}^L, \{\Sigma_{X_i}\}_{i=1}^L),$$

where the involved quantities are defined above.

We immediately remark that both θ^* and θ_X depend on the probability distribution of X, which is in general unknown. In Section 4.3.1, we show how to approximate θ^* by taking advantage of a training set of the form $\{(x_j, y_j)\}_{j=1}^N$ sampled from (X, Y). This qualifies the strategy as a supervised learning algorithm. Instead, in Section 4.3.2, we show how to approximate θ_X by means of a training set of the form $\{x_j\}_{j=1}^N$, i.e., in an unsupervised way.

We also observe that, if we further assume that E is a Gaussian random variable and that X is a mixture of Gaussian random variables, then the outcomes of the two strategies coincide, provided that $\|\theta_X\|_\infty \leq \varrho$. Indeed, by Theorem 14, we know that R_{θ_X} is the Bayes estimator R^*, and so

$$R^* = R_{\theta_X} = R_{\theta^*}.$$

Nevertheless, the trained networks R_{θ^*} and R_{θ_X} could also be good estimators even in contexts that slightly deviate from the setting of Gaussian random variables. In particular, we are interested in testing them in cases in which X is group-sparse, but we are unsure if it is distributed as a Gaussian mixture model. If this is the case, the two parameters are in general distinct - and also different from R^*, thus they are not theoretically guaranteed to be good estimators - and the two approximation strategies might achieve different levels of performance.

Notice also that, despite we suppose to deal with the sparsity prior provided by the degenerate Gaussian mixture model, the proposed strategies hold for a general mixture.

4.3.1 Supervised approach

Let $\{(x_j, y_j)\}_{j=1}^N$ be a training set, where $x_j \sim X$ are i.i.d., $\varepsilon_j \sim E$ are i.i.d. and $y_j = Ax_j + \varepsilon_j$. The parameter $\theta = (w_i, \mu_i, \Sigma_i)_{i=1}^L$ of the neural network defined in Section 4.2.1 can be learned by minimizing the empirical risk

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^N \|x_j - R_\theta(y_j)\|_2^2,$$

which is the squared MSE between the original signals, x_j, and the reconstructions provided by the neural network, $R_\theta(y_j)$. In order to enforce sparsity, it is possible to add a second term to the empirical risk, namely to consider

$$\mathcal{L}_s(\theta) = \mathcal{L}(\theta) + \lambda \mathcal{J}(\theta),$$

where λ is a regularization parameter, $\mathcal{J}(\theta) = \sum_{i=1}^{L} \|\Sigma_i\|_*$, and $\|\cdot\|_*$ is the nuclear norm. Such a regularization term can be equivalently defined as the ℓ^1 norm of the singular values of Σ_i, hence it promotes low-rank covariances without requiring the knowledge of their eigenvectors.

The evaluation of the nuclear norm is computationally expensive and should be done L times for each step of the minimization process. Another option is taking $\mathcal{J}(\theta) = \sum_{i=1}^{L} \|\Sigma_i\|_{\mathrm{F}}^2$, where $\|\cdot\|_{\mathrm{F}}$ represents the Frobenius norm. This choice is computationally convenient, but does not promote any kind of sparsity on the covariances. However, in the numerical simulations, we do not observe significant differences when using the nuclear norm, the Frobenius norm or no regularization term. Therefore, in the numerical results of Section 4.5 we did not make use of any regularization term.

4.3.2 Unsupervised approach

Suppose to be in an unsupervised setting, i.e. to have a training set $\{x_j\}_{j=1}^N$, where x_j are sampled i.i.d. from the mixture $X = \sum_{i=1}^{L} X_i \mathbb{1}_{\{i\}}(I)$. We now wish to find an approximation for the means μ_{X_i}, the covariances Σ_{X_i} and the weights w_{X_i} of the mixture. We propose the following two-step procedure.

1. **Subspace clustering**. The elements of the training set $\{x_j\}_{j=1}^N$ sampled from different degenerate distributions X_i belong to different subspaces of $\mathbb{R}^n$. We propose to cluster these points by using subspace clustering and, in particular, the technique provided in [160]. As a result, the training set is partitioned into $\widehat{L}$ subsets. Ideally, we should have $\widehat{L} = L$, and each subset should correspond to one Gaussian of the mixture.

2. **The parameters**. We compute the empirical means $\{\widehat{\mu}_i\}_{i=1}^{\widehat{L}}$ and the empirical covariances $\{\widehat{\Sigma}_i\}_{i=1}^{\widehat{L}}$ of each cluster, and we estimate the weights $\{\widehat{w}_i\}_{i=1}^{\widehat{L}}$ of the mixture by using the number of elements in the clusters.

Once this is done, we can use the neural network defined in Section 4.2.1 with these parameters. More precisely, this is the neural network $R_{\widehat{\theta}_X}$, where

$$\widehat{\theta}_X = (\{\widehat{w}_i\}_{i=1}^{\widehat{L}}, \{\widehat{\mu}_i\}_{i=1}^{\widehat{L}}, \{\widehat{\Sigma}_i\}_{i=1}^{\widehat{L}}).$$

4.4 Baseline algorithms

In this section we describe some of the most popular regularization techniques used to solve linear inverse problems with a sparsity prior. These include Least Absolute

Shrinkage and Selection Operator (LASSO), Group LASSO, Iterative Hard Thresholding (IHT) and Dictionary Learning. In the numerical experiments of Section 4.5, we will compare our two proposed approaches against the methods presented in this section.

4.4.1 LASSO

Least Absolute Shrinkage and Selection Operator, LASSO [219], is one of the most acknowledged techniques for sparsity promotion, also in the context of inverse problems. It involves the minimization of a functional composed by the sum of a data fidelity term and a regularization term that promotes sparsity with respect to a given basis. In particular, the regularization term is the ℓ^1 norm of the components of the unknown in the sparsifying basis.

More precisely, we suppose that the unknown x in (4.1) is sparse with respect to an orthonormal basis $\mathcal{B}$, and we denote by $M \in \mathbb{R}^{n\times n}$ the (orthogonal) matrix representing the change of basis from $\mathcal{B}$ to the canonical one, also known as the synthesis operator. Then, the functional to be minimized is

$$\mathcal{F}(\beta) = \frac{1}{2}\|y - AM\beta\|_2^2 + \lambda\|\beta\|_1, \tag{4.12}$$

where $\lambda > 0$ is a regularization parameter. The minimization of $\mathcal{F}(\beta)$ over $\mathbb{R}^n$, also known as synthesis formulation of LASSO, can be performed employing the Iterative Soft Thresholding Algorithm (ISTA) [70, 107], which is a proximal-gradient descent method involving the computation of the proximal operator for the convex and non-smooth term of the functional (the ℓ^1-norm), and the gradient descent step for the smooth term (the data fidelity). The proximal operator of the ℓ^1-norm is the soft thresholding operator, from which ISTA takes its name. Mathematically, the $(k+1)$-th iteration is

$$\beta^{k+1} = S_{t\lambda}(\beta^k - tM^TA^T(AM\beta^k - y)), \tag{4.13}$$

where $t > 0$ is a stepsize and $S_\lambda \colon \mathbb{R}^n \to \mathbb{R}^n$ is the soft thresholding operator defined componentwise as

$$S_\lambda(\beta)_i = \max\{|\beta_i| - \lambda, 0\}\,\mathrm{sign}(\beta_i),$$

where i indicates the component.

For the denoising problem ($A = I$), setting $t = 1$, algorithm (4.13) achieves convergence in a single step. Therefore, the solution $\bar{\beta}$ is

$$\bar{\beta} = S_\lambda(M^TA^Ty).$$

The main drawbacks of this method are the choice of the regularization parameter λ, and the required prior knowledge of the synthesis operator M (i.e., of the basis $\mathcal{B}$ with respect to which the unknown is sparse). In Section 4.4.4, we show how to combine dictionary learning techniques to fill this gap. Other alternatives, explored in the numerical experiments in Section 4.5, are to leverage prior knowledge on M, or to infer it via the singular value decomposition (SVD).

4.4.2 Group LASSO

An extension of LASSO is Group LASSO [87], which encodes the idea that the features of the possible unknowns can be organized into groups. This is achieved by introducing L different coordinate systems, each represented by an orthogonal synthesis matrix $M_i \in \mathbb{R}^{n\times n}$, and by minimizing the following functional:

$$\mathcal{F}(\beta) = \frac{1}{2}\left\| y - A\sum_{i=1}^{L} M_i\beta_i \right\|_2^2 + \lambda \sum_{i=1}^{L} \|\beta_i\|_{K_i} \tag{4.14}$$

where β_i is the coefficient vector corresponding to the i-th group, $\beta \in \mathbb{R}^{nL}$ is the concatenation of the vectors $\beta_1, \dots, \beta_L$, and $\lambda > 0$ is a regularization parameter. The second term of (4.14) is a regularization functional encouraging entire groups of features to be either included or excluded from the model. The weighted norm $\|\beta_i\|_2$ with $\|\beta_i\|_{K_i} := (\beta_i^T K_i \beta_i)^{\frac{1}{2}}$ is employed to promote prior information regarding the sparse representation of each group. The minimization of (4.14) can be performed via the iterative algorithm proposed in [231, Proposition 1], showing strong connections with the proximal gradient descent method, which reads

$$\beta^{k+1} = \text{prox}_{t\lambda f}(\beta^k - t\tilde{M}^T A^T(A\tilde{M}\beta^k - y)),$$

where $f(\beta) = \sum_{i=1}^{L} \|\beta_i\|_{K_i}$, $\tilde{M} = [M_1|M_2|...|M_L]$, and $t > 0$ is a stepsize. The map $\text{prox}_{t\lambda f}$ satisfies (see [32, Theorem 6.6, Chapter 6])

$$\text{prox}_{t\lambda f}(\beta) = (\text{prox}_{t\lambda f_i}(\beta_i))_{i=1}^{L},$$

being $f_i(\beta_i) = \|\beta_i\|_{K_i}$, and the proximal operator of the ℓ^2 weighted norm is (see [32, Lemma 6.68, Chapter 6])

$$\text{prox}_{t\lambda f_i}(\beta_i) = \begin{cases} \beta_i - K_i^T(K_iK_i^T)^{-1}K_i\beta_i & \|(K_iK_i^T)^{-1}K_i\beta_i\|_2 \le t\lambda \\ \beta_i - K_i^T(K_iK_i^T + \alpha^* I)^{-1}K_i\beta_i & \|(K_iK_i^T)^{-1}K_i\beta_i\|_2 > t\lambda \end{cases},$$

where α^* is the unique positive root of the non decreasing function $g(\alpha) := \|(K_iK_i^T + \alpha I)^{-1}K_i\beta_i\|_2^2 - (t\lambda)^2$. Since the computation of the proximal operator involves the root-finding problem, the whole process is computationally very expensive. As for LASSO, also for Group LASSO one has to choose the regularization parameter λ, the stepsize t, and to know a priori the group bases $\mathcal{B}_i$ for $i = 1, ..., L$.

4.4.3 IHT

The Iterative Hard Thresholding algorithm, as presented in [42], can be employed as a reconstruction method for inverse problems in which the unknown belongs to the union of L subspaces. This can be seen as a model of sparsity. Once an orthogonal synthesis matrix $M \in \mathbb{R}^{n\times n}$ is introduced, the coefficients β such that $x = M\beta$ are supposed to satisfy $\beta \in \mathcal{S} = \cup_{i=1}^{L}\mathcal{S}_i$, where each $\mathcal{S}_i$ is a coordinate subspace in $\mathbb{R}^n$ of dimension s or

smaller, i.e., the span of a subset of cardinality at most s of the canonical basis. IHT then involves the minimization of

$$\mathcal{F}(\beta) = \frac{1}{2}\|y - AM\beta\|_2^2 + \chi_{\mathcal{S}}(\beta), \tag{4.15}$$

where

$$\chi_{\mathcal{S}}(\beta) = \begin{cases} 0 & \beta \in \mathcal{S} \\ +\infty & \text{otherwise} \end{cases}$$

forces the unknown to belong to $\mathcal{S}$.

The iterative algorithm to minimize (4.15) consists in the alternation of a gradient descent step in the direction given by the MSE and a projection onto $\mathcal{S}$, namely the $(k+1)$-th iteration is

$$\beta^{k+1} = P_{\mathcal{S}}(\beta^k - tM^T A^T(AM\beta^k - y)), \tag{4.16}$$

where $P_{\mathcal{S}}(\beta) = P_{\mathcal{S}_{\bar{i}}}(\beta)$, $\bar{i} = \arg\min_i\{\|P_{\mathcal{S}_i}(\beta) - x\|_2\}$, $P_{\mathcal{S}_i}$ is the orthogonal projection onto $\mathcal{S}_i$ and $t > 0$ is a stepsize. The projection onto coordinate hyperplanes is performed by simply setting to 0 all the non-active coordinates. The overall projection $P_{\mathcal{S}}(\beta)$ is also very efficient, since it consists only of computing L projections onto the different subspaces $\mathcal{S}_i$ and selecting the optimal one. Moreover, if $\mathcal{S}$ is the union of all coordinate hyperplanes of dimension s, $P_{\mathcal{S}}(\beta)$ simply reduces to selecting the s largest components of β.

For the denoising problem ($A = I$) the IHT algorithm (4.16) with $t = 1$ converges to the solution in a single step. Therefore, using the same definitions as before, the solution is

$$\bar{\beta} = P_{\mathcal{S}_{\bar{i}}}(M^T y).$$

As for LASSO and Group LASSO, the subspaces $\mathcal{S}_i$, and especially the basis given by M, should be known a priori or should be inferred. In this setting, the sparsity level s can play the role of a regularization parameter (such as λ of LASSO and Group LASSO) and is tuned by suitable heuristic methods.

4.4.4 Dictionary Learning

The baseline algorithms proposed in the previous sections need the a priori knowledge of the basis with respect to which the unknown is sparse. A possible approach to infer it is *sparse dictionary learning* (also known as sparse coding), where a dictionary with respect to which the unknown is sparse is learned from the data [165]. Note that this dictionary does not necessarily have to be a basis. More precisely, given a training set $\{x_j\}_{j=1}^N$ with $x_j \in \mathbb{R}^n$, a sparsifying dictionary $D \in \mathbb{R}^{n\times d}$, where d is the number of elements of the dictionary (the columns of the matrix D), can be found as

$$\min_{D\in\mathbb{R}^{n\times d},\, \beta\in\mathbb{R}^{d\times N}} \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{2}\|x_j - D\beta_j\|_2^2 + \lambda\|\beta_j\|_1\right), \tag{4.17}$$

where $\lambda > 0$ is a regularization parameter and $\beta_j \in \mathbb{R}^d$ is the sparse representation of x_j with respect to the dictionary D. If the dictionary is fixed, the functional in (4.17) resembles the LASSO functional (4.12), except for the fact that the dictionary may not be a basis and that we are summing over the elements of the training set. However, in this case, the goal is mainly to learn D. In order to solve the minimization problem in (4.17), we rely on the online method used in [165], which iteratively minimizes over one variable keeping the others fixed. We update the sparse representation β by using the least angle regression (LARS) algorithm and the dictionary D by using block coordinate descent.

We assume that $D \in \mathbb{R}^{n\times d}$ is a full-rank matrix. In general, the number of atoms d is allowed to be larger than the original dimension of the signal n. Such a redundancy may arise with frames or with the union of multiple bases. Nevertheless, here we consider the non-redundant setting, with $d \leq n$. This is the case for all the numerical experiments reported in this work.

Once the dictionary is learned, the solution to the linear inverse problem (4.3) can be computed by combining the baseline algorithms of the previous sections. Under the assumptions previously introduced, the matrix D is injective, and the pseudo-inverse $D^+ = (D^TD)^{-1}D^T$ is its left inverse. We thus tackle the minimization of the following functional:

$$\mathcal{F}(x) = \frac{1}{2}\|y - Ax\|^2 + \lambda G(x), \qquad G(x) = \chi_{\mathrm{Im}(D)}(x) + \|D^+x\|_1, \tag{4.18}$$

where $\chi_{\mathrm{Im}(D)}(x)$ is the characteristic function of the range $\mathrm{Im}(D)$. We minimize the functional $\mathcal{F}$ by means of a proximal-gradient scheme, whose iterations read as

$$x^{k+1} = \mathrm{prox}_{\lambda t G}(x^k - tA^T(Ax^k - y)), \tag{4.19}$$

where $t > 0$ is a stepsize and

$$\begin{aligned}\mathrm{prox}_G(z) &= \underset{x\in\mathrm{Im}(D)}{\arg\min}\left\{\frac{1}{2}\|x - z\|^2 + \|D^+x\|_1\right\} \\ &= D\,\underset{\beta\in\mathbb{R}^d}{\arg\min}\left\{\frac{1}{2}\|D\beta - z\|^2 + \|\beta\|_1\right\}.\end{aligned}$$

Notice that, under the injectivity assumption on the matrix D, the minimization of $\mathcal{F}$ in (4.18) is equivalent to a LASSO problem in synthesis formulation, using D as a synthesis operator. Despite the minimizers of those two problems are the same, the iterates approximating them by proximal-gradient schemes do not coincide. For numerical reasons, we prefer the formulation (4.18), leading to the iterations (4.19). Indeed, as it will be discussed in Section 4.5, the implementation of prox_G can be done efficiently by means of the same routines employed for dictionary learning. For the denoising problem ($A = I$) the algorithm (4.19) with $t = 1$ converges to the solution $\bar{x}$ in a single step, namely

$$\bar{x} = D\,\mathrm{prox}_{\lambda G}(y). \tag{4.20}$$

Since the degenerate Gaussian mixture prior represents a group sparsity prior, it is useful to compare our algorithms with a "group" version of sparse dictionary learning. This has been explored with the name "Block-sparse" or "Group-sparse" dictionary learning [232, 155].

Those techniques, however, do not share the same perspective on group sparsity as the one inspiring our proposed algorithms. The common feature behind all of them is the assumption that, when the signals are represented in a suitable basis or dictionary, there exists groups, or blocks, of coordinates that are simultaneously activated. In our degenerate Gaussian mixture approach, though, the signals can be clustered according to which group of coordinates they activate: each signal is thus associated with a single group. In block-sparse dictionary learning (and, similarly, in the Group LASSO approach previously exposed), the active components of each signal can also belong to (a small number of) different groups.

For this reason, we propose an alternative dictionary learning technique that is closer to our setup and can be employed for more direct comparisons. We simply call it "Group Dictionary Learning": after doing a subspace clustering of the training set as explained in Section 4.3.2, it is possible to learn a dictionary D_i for each group obtained with the clustering. Assuming that each $D_i \in \mathbb{R}^{n\times d_i}$ is injective for $i = 1, ..., L$, and adopting the same approach as in (4.18), we tackle the minimization of the following functional:

$$\begin{aligned} \mathcal{F}(x) &= \frac{1}{2}\|y - Ax\|^2 + \lambda \hat{G}(x), \\ \hat{G}(x) = \min_{i=1,\dots,L} G_i(x), &\qquad G_i(x) = \chi_{\mathrm{Im}(D_i)}(x) + \|D_i^+ x\|_1. \end{aligned} \tag{4.21}$$

Again, we minimize $\mathcal{F}$ by means of a proximal-gradient scheme as in (4.19), replacing prox_G by $\mathrm{prox}_{\hat{G}}$, which can be easily computed as

$$\mathrm{prox}_{\hat{G}}(z) = \mathrm{prox}_{G_I}(z), \quad I = \arg\min_{i=1,\dots,L} \left\{ \frac{1}{2}\|z - \mathrm{prox}_{G_i}(z)\|^2 + G_i\big(\mathrm{prox}_{G_i}(z)\big) \right\}.$$

Remark 19. Differently from dictionary learning, the approaches proposed in Section 4.3 do not focus on the reconstruction of a dictionary D. However, such dictionary might be obtained as a by-product, e.g. by computing the singular vectors of each covariance Σ_i and by collecting them in a single matrix, but the theoretical properties of such an object are out of the scope of the present work. Nevertheless, we wish to underline that, as in the problem of dictionary learning, our algorithms do not require as an input the knowledge of the dictionary D . It is worth noting that the parameters $\theta = (w_i, \mu_i, \Sigma_i)_{i=1}^L$ of our network are $O(n^2 \times L)$ while the dictionary has only $n \times d$ parameters, where d is usually chosen as $O(n)$. Therefore, one of the reasons our methods seem more effective may be that they employ more parameters, a well-know (but possibly not fully understood) phenomenon related to overparametrization in deep learning.

4.5 Numerical results

In this section, we compare the algorithms proposed in Section 4.3 with the baseline methods discussed in Section 4.4 for 1D denoising and deblurring problems with three datasets. Our experiments primarily aim to compare our algorithms with classical sparsity-promoting methods on simplified signal classes, rather than striving for state-of-the-art results. The exploration of more complex inverse problems and the incorporation of real-world data fall outside the scope of this chapter and are designated for future research.

4.5.1 Datasets

We consider three datasets with increasing complexity.

1. *Gaussian Mixture Model*: This dataset contains samples from a degenerate Gaussian mixture variable (4) where I is a uniform variable (all the Gaussians of the mixture have the weight $w_i = \frac{1}{L}$). Each X_i is a Gaussian random variable in $\mathbb{R}^n$, having mean $\mu_i = \mathbf{0}$ and whose covariance matrix Σ_i has zero entries everywhere, except from s elements on the diagonal which are set to 1. This indicates that we are considering a mixture of degenerate variables, each of which has support on an s-dimensional coordinate hyperplane of $\mathbb{R}^n$ and whose components are independent standard Gaussian random variables. In our experiments we consider $n = 1000$, $s = \operatorname{rank} \Sigma_i = 20$ and $L = 10$. Therefore, this dataset consists of very sparse random variables in a rather large ambient space. The sparsity is nevertheless very structured, since only $L = 10$ combinations of s active coefficients are considered.

2. *Sinusoidal functions with one discontinuity*: This dataset contains functions with support in $[0, 4\pi]$ of the type

$$x(\tau) = \begin{cases} A\sin(\omega\tau) + B & 0 \leq \tau \leq \tau_i \\ A\sin(\omega\tau) + B + C & \tau_i < \tau \leq 4\pi \end{cases}$$

where the amplitude $A \sim \mathcal{U}(0.05, 0.1)$, the angular frequency $\omega \sim \mathcal{U}(1, 2)$ and the vertical translations $B \sim \mathcal{U}(\frac{1}{2}, 3)$, $C \sim \mathcal{N}(0, 0.2^2)$. The discontinuity points τ_i with $i = 1, ..., L$ and $L = 10$ correspond to different subspaces and are equispaced points in $[0, 4\pi]$. Each signal x is discretized on 1000 equispaced points in $[0, 4\pi]$. These functions can be approximately seen as samples from a degenerate Gaussian mixture in a wavelet basis. While the coarse scale coefficients are expected to be nonzero for all the signals, the wavelet coefficients at the finer scales will be relevant only in the locations of the discontinuity, and negligible in the smooth regions. Therefore, each discontinuity determines a coordinate subspace with respect to the wavelet basis. Considering that the amplitude of the jump C follows a Gaussian distribution, we posit that the fine scale coefficients can be effectively modeled as degenerate Gaussian random vectors. We can also infer an estimate of the sparsity level in terms of the size of the support of the mother wavelet, and of the considered resolution scales.

3. *Truncated Fourier series with two discontinuities*: This dataset contains functions with support in $[0, 4\pi]$ of the form

$$x(\tau) = \begin{cases} \sum_{d=1}^{4} a_d \cos(2\pi d\tau) + b_d \sin(2\pi d\tau) & 0 \leq \tau \leq \tau_i(1) \\ \sum_{d=1}^{4} a_d \cos(2\pi d\tau) + b_d \sin(2\pi d\tau) + C_1 & \tau_i(1) < \tau \leq \tau_i(2) \\ \sum_{d=1}^{4} a_d \cos(2\pi d\tau) + b_d \sin(2\pi d\tau) + C_2 & \tau_i(2) < \tau \leq 4\pi \end{cases}$$

where the Fourier coefficients $a_d, b_d \sim \mathcal{N}(0.1, 0.1^2)$ for $d = 1, ..., 4$ and the vertical translations $C_1, C_2 \sim \mathcal{N}(0, 0.2^2)$. As for dataset 2, the discontinuity points are 10 equispaced points in $[0, 4\pi]$ and τ is discretized with 1000 equispaced points in $[0, 4\pi]$. However, each function has either one or two discontinuities. More precisely, if $\tau_i(1) = \tau_i(2)$ the function has one discontinuity, otherwise it has two discontinuities. Therefore the subspaces are the ones representing functions with 2 discontinuities (all the possible combinations of 10 elements in groups of 2) and the ones representing functions with 1 discontinuity (10 subspaces), so the total number of subspaces is $L = \binom{10}{2} + 10 = 55$. For the same reasons discussed above for dataset 2, these functions can be seen as approximate samples from a degenerate Gaussian mixture distribution in the wavelet domain. The level of sparsity can be estimated as in the case of the previous dataset, updating it to the presence of two distinct singularities.

4.5.2 Methods

In the following experiments, we test our supervised and unsupervised algorithms described in Sections 4.3.1 and 4.3.2 and compare them with the baseline algorithms described in Section 4.4. More precisely, we consider the following methods.

(A) *Supervised* (§4.3.1).

(B) *Unsupervised*(§4.3.2).

(C) *Dictionary learning* (§4.4.4). We learn a sparsifying dictionary D for the training set, using the Lasso LARS algorithm to solve (4.17); then, we employ the learned D to denoise the test signal by minimising (4.18). In particular, we rely on the dictionary learning algorithm provided in the *scikit-learn* library [183]. The routines provided there also allow to implement (4.20): in particular, the computation of prox_G can be done through the *transform* method of the learned dictionary. We choose the number of elements of the dictionary as $d = \frac{n}{2}$, where n is the signal length, in order to balance expressivity and numerical efficiency.

(D) *Group dictionary learning* (§4.4.4). After dividing the training set into L clusters and learning a sparsifying dictionary D_i for each group, we minimise (4.21). In this case, we choose $d = \frac{n}{2L}$.

(E) *IHT with SVD basis* (§4.4.3). We infer the basis with respect to which the unknown is sparse by computing the SVD of the empirical covariance matrix of the training

set and considering the orthonormal basis composed by the eigenvectors. Then, we choose $\mathcal{S}$ as the union of all the s-dimensional coordinate subspaces in $\mathbb{R}^n$ w.r.t. the inferred sparsity basis. Here we choose the degree of sparsity s by minimizing the relative MSE (computed w.r.t. the norm of the original signals) over the training set.

(F) *IHT with SVD bases of groups* (§4.4.3). We divide the training set into L clusters as explained in Section 4.3.2. Then, for each group provided by the clustering, we infer the basis with respect to which that group is sparse by computing the SVD of its empirical covariance matrix and considering the eigenvectors corresponding to the s largest eigenvalues, where the degree of sparsity s is chosen as for (E). Finally, we choose $\mathcal{S}$ as the union of the s-dimensional subspaces spanned by the bases inferred.

(G) *IHT with known basis* (§4.4.3). We suppose to know the basis with respect to which the unknown is sparse. In particular, for Dataset 1 we consider the canonical basis, while Datasets 2 and 3 require a more complicated treatment. Indeed, as discussed in Section 4.5.1, their representation in a suitable wavelet basis is sparse at fine scales. For this reason, we preprocess the datasets by applying the wavelet transform, we keep the low-scale coefficients fixed, and apply IHT only to the fine scales. We choose the wavelet basis generated by the Daubechies wavelet with 6 vanishing moments.

(H) *LASSO with SVD basis* (§4.4.1). We infer the basis with respect to which the unknown is sparse as in (E). Then, we minimize (4.12).

(I) *Group LASSO with SVD bases* (§4.4.2). We divide the training set into L clusters as explained in Section 4.3.2. Then, for each group provided by the clustering we infer the basis with respect to which that group is sparse as explained in (F). Here, however, we consider the complete bases provided by the SVDs of the empirical covariance matrices of the groups. In order to minimize the functional (4.14), we could in principle take advantage of the algorithm proposed in Section 4.4.2, which is nevertheless computationally expensive because it must be applied on each vector y_j separately. The resulting algorithm is computationally rather expensive: indeed, the iterative computation of the proximal operator, involving the root-finding problem, must be applied on each vector y_j separately. To ease the computational burden, in our experiments we instead minimize (4.14) through the ADAM optimization scheme implemented in the *pytorch* library, which can process multiple signals in parallel. We choose the power $-\frac{1}{2}$ of the empirical covariances matrices of each group as penalty matrices K_i.

(J) *LASSO with known basis* (§4.4.1). We incorporate in LASSO the knowledge of the basis with respect to which the unknown is sparse, as already explained for (G). This implies that, for Dataset 1, we solve LASSO with the canonical basis, whereas for Datasets 2 and 3 we first compute the wavelet transform and then solve LASSO only on the high-scale coefficients.

Whenever needed (methods (C), (D), (H), (I), and (J)), we always choose the optimal regularization parameter λ, namely, the one minimizing the relative MSE over the training set. Further, whenever needed (methods (A), (B), (D), (E), and (I)), we always suppose to know the number of Gaussians L in the mixture.

Methods (B), (D), (E), and (I) rely on clustering. The subspace clustering procedure described in Section 4.3.2 is applied to the signals for Dataset 1 and to their derivatives computed as finite difference approximations for Datasets 2 and 3. Indeed, for the latest datasets, the derivative of the signals highlights the discontinuity points and allows for a more efficient clustering.

It is worth noting that methods (G) and (J) take advantage of the knowledge of the basis with respect to which the unknown is sparse. For this reason, they do not constitute a fair comparison for all the other algorithms, which do not use this piece of information. However, we are interested to see if they can outperform our methods.

4.5.3 Denoising

In this section we focus on the denoising problem with 10% of noise, namely $y = x + \varepsilon$, where ε is sampled from $\mathcal{N}(\mathbf{0}, \sigma^2 I)$ where σ is the 10% of the maximum of the amplitudes of the training set signals, i.e. $\max_f(\max f - \min f)$ where f is a training sample and the maximum is taken for each dataset. We compare the performances of the methods reported in Section 4.5.2.

In Table 4.1 we show the mean over 2000 signals of the test set of the relative MSE between the original signals and the different reconstructions. We observe that our unsupervised algorithm (B) provides the best reconstructions for Datasets 1 and 3, and for Dataset 2 it is only outperformed by group dictionary learning (D). However, dictionary learning is computationally much more expensive than our unsupervised method, and the gain in terms of reconstruction is minimal (as can be visually seen in Figure 4.2). From Table 4.1 it seems clear that the best baseline algorithms to which we can compare our approaches are dictionary learning (C), among the methods that do not use clustering, and group dictionary learning (D), among the methods using the groups obtained by the clustering of the training set. For this reason, in Figure 4.2 we show a qualitative comparison between the reconstructions provided by the methods described in 4.5.2. We notice that for Datasets 2 and 3 the algorithms learn all the discontinuities and try to remove those not present in the signal.

Considering that accurately clustering the training set is challenging (see e.g. Dataset 3), we question whether precise clustering is truly necessary for employing the unsupervised algorithm proposed in Section 4.3.2. For this purpose, in Table 4.2 we show the mean over 2000 signals of the test set of the relative MSE between the original signals and the reconstructions provided by the unsupervised approach with exact clustering and with random clustering for the denoising problem with 10% of noise. We observe that correct clustering is important for Dataset 1, but not so important for Datasets 2 and 3. We believe that this is due to the fact that in Datasets 2 and 3 the energy of the signals is shared between the smooth part, which is independent of the clustering, and the discontinuities. In Dataset 1, the absence of the (non-zero) smooth part makes the

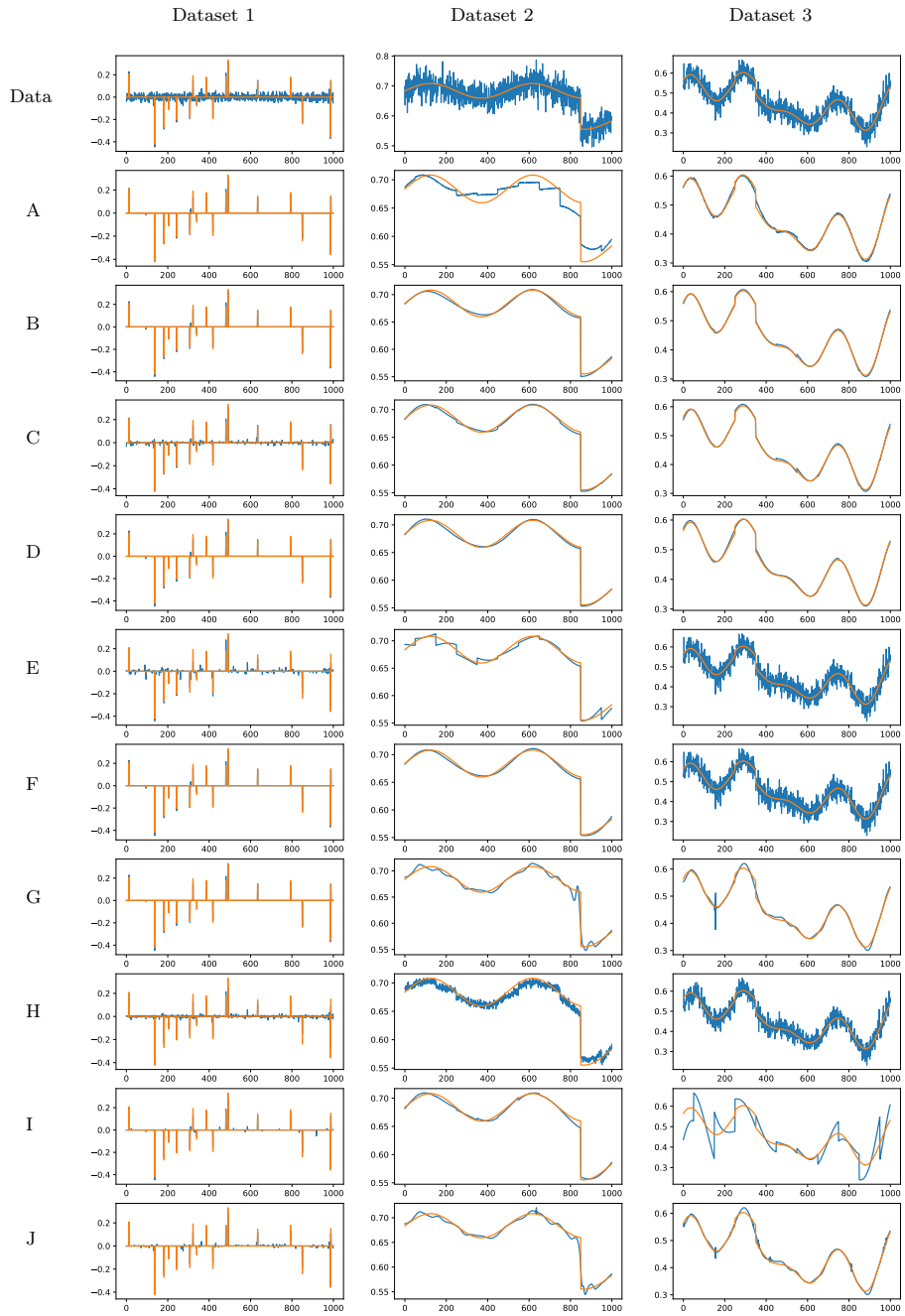


Figure 4.2: Qualitative comparisons for the denoising problem with 10% noise. In each column we show a signal of the test set from Datasets 1, 2 and 3, respectively. In each row we report the original data in orange and the noisy data, the reconstructions obtained with the methods supervised (A), unsupervised (B), dictionary learning (C), group dictionary learning (D), IHT with SVD basis (E), IHT with SVD bases of groups (F), IHT with known basis (G), LASSO with SVD basis (H), Group LASSO with SVD bases (I), and LASSO with known basis (J), respectively, in blue.

Table 4.1: Relative MSE values for the denoising problem with 10% noise.

	Dataset 1	Dataset 2	Dataset 3
Supervised (A)	1.97%	$1.70\ 10^{-2}$%	$2.71\ 10^{-2}$%
Unsupervised (B)	**0.98**%	$1.80\ 10^{-3}$%	$\mathbf{6.32\ 10^{-3}}$%
Dictionary learning (C)	4.18%	$3.43\ 10^{-3}$%	$7.13\ 10^{-3}$%
Group dictionary learning (D)	0.99%	$\mathbf{1.70\ 10^{-3}}$%	$2.42\ 10^{-2}$%
IHT with SVD basis (E)	9.93%	$1.25\ 10^{-2}$%	$3.97\ 10^{-1}$%
IHT with SVD bases of groups (F)	0.99%	$2.04\ 10^{-3}$%	$3.97\ 10^{-1}$%
IHT with known basis (G)	2.78%	$1.22\ 10^{-2}$%	$2.46\ 10^{-2}$%
LASSO with SVD basis (H)	9.24%	$1.55\ 10^{-2}$%	$3.07\ 10^{-1}$%
Group LASSO with SVD bases (I)	3.01%	$3.71\ 10^{-3}$%	$8.31\ 10^{-1}$%
LASSO with known basis (J)	4.69%	$1.00\ 10^{-2}$%	$2.15\ 10^{-2}$%

Table 4.2: Relative MSE values for the denoising problem with 10% noise with the unsupervised approach (B) when using exact vs. random clustering.

	Dataset 1	Dataset 2	Dataset 3
Exact	0.97%	$1.66\ 10^{-3}$%	$3.37\ 10^{-3}$%
Learned	0.98%	$1.80\ 10^{-3}$%	$6.32\ 10^{-3}$%
Random	8.25 ± 0.04%	$(3.46 \pm 0.25)\ 10^{-3}$%	$(5.70 \pm 0.55)\ 10^{-3}$%

dependence on the clustering stronger.

4.5.4 Deblurring

In this section we focus on a deblurring problem with 10% of noise, namely we consider the problem $y = q * x + \varepsilon$, where $*$ represents the discrete convolution, the noise ε is sampled from $\mathcal{N}(\mathbf{0}, \sigma^2 I)$ and σ is the 10% of the maximum of the amplitudes of the training set signals. We choose the filter q to be Gaussian, i.e. the entries of q are a finite discretization of the density of $\mathcal{N}(0, \sigma_b^2)$ in a neighborhood of 0. For Dataset 1 we choose $\sigma_b = 1$, while for Datasets 2 and 3 we set $\sigma_b = 30$ and $\sigma_b = 20$, respectively. The latter values of σ_b are larger because the effect of the convolution on piecewise smooth signals (Datasets 2 and 3) is less visible than on Dirac deltas (Dataset 1). Since for the denoising problem our unsupervised method provides better results than the supervised one, we only show the results for the deblurring problem using the unsupervised technique and we compare it with the most significant baseline algorithms. In particular, we consider the following methods: Unsupervised (B), Dictionary learning (C), Group dictionary learning (D), IHT with SVD bases of groups (F) and Group LASSO with SVD bases (I). We exclude from the comparisons the methods that do not employ clustering, except for dictionary learning, as they performed significantly worse in the denoising experiments.

In Table 4.3 we show the mean over 2000 signals of the test set of the relative MSE between the original signals and the different reconstructions. As for the denoising

Table 4.3: Relative MSE values for the deblurring problem with 10% noise.

	Dataset 1	Dataset 2	Dataset 3
Unsupervised (B)	**3.68**%	**2.65 10^{-3}**%	**1.01 10^{-2}**%
Dictionary learning (C)	14.32%	6.61 10^{-3}%	1.28 10^{-2}%
Group dictionary learning (D)	13.51%	4.62 10^{-3}%	3.41 10^{-2}%
IHT with SVD bases of groups (F)	3.80%	5.54 10^{-3}%	9.48 10^{-1}%
Group LASSO with SVD bases (I)	11.48%	1.34 10^{-2}%	9.11 10^{-1}%

problem, our unsupervised method outperforms the others.

In Figure 4.3 we show a qualitative comparison between the reconstructions provided by some the methods described in 4.5.2.

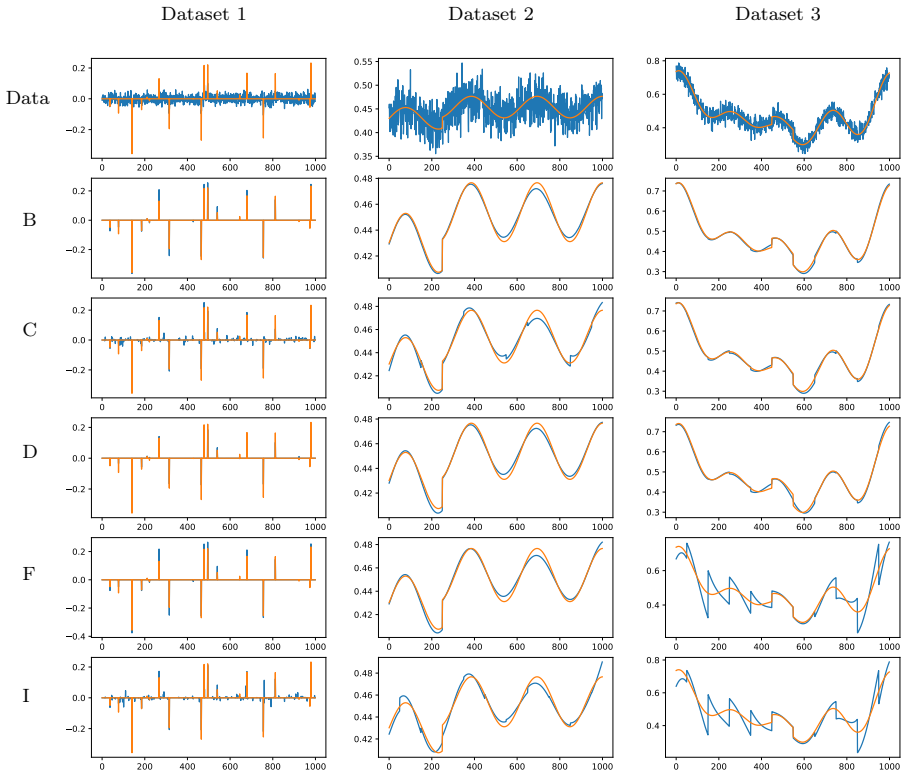


Figure 4.3: Qualitative comparisons for the deblurring problem with 10% noise. In each column we show a signal of the test set from Datasets 1, 2 and 3, respectively. In each row we report the original data in orange and the noisy data, the reconstructions obtained with the methods unsupervised (B), dictionary learning (C), group dictionary learning (D), IHT with SVD bases of groups (F), and Group LASSO with SVD bases (I), respectively, in blue.

Conclusions

In Chapter 2 of this thesis, we have introduced CGNNs, a family of generative models in the continuous, infinite-dimensional, setting, generalizing popular architectures such as DCGANs [189]. We have shown that, under natural conditions on the weights of the networks and on the nonlinearity, a CGNN is globally injective. This allowed us to obtain a Lipschitz stability result for (possibly nonlinear) ill-posed inverse problems, with unknowns belonging to the manifold generated by a CGNN.

The main mathematical tool used is wavelet analysis and, in particular, a multi-resolution analysis of $L^2(\mathbb{R})$. While wavelets yield the simplest multi-scale analysis, they are suboptimal when dealing with images. So, it would be interesting to consider CGNNs with other systems, such as curvelets [53] or shearlets [144], more suited to higher-dimensional signals, or, more generally, CGNNs made of an injective discrete neural network composed with a synthesis operator not necessarily associated to a multi-scale structure.

For simplicity, we considered only the case of a smooth nonlinearity σ: we leave the investigation of Lipschitz σ's to future work. This would allow for including many commonly used activation functions. Some simple illustrative numerical examples are included in this work, which was mainly focused on the theoretical properties of CGNNs. It would be interesting to perform more extensive numerical simulations in order to better evaluate the performance of CGNNs, also with nonlinear inverse problems, such as electrical impedance tomography.

The stability estimate obtained in Theorem 8 is based on a more general result that holds also with manifolds that are not described by one single generator [5]. Therefore in Chapter 3 we extend our work combining multiple generators to learn several charts of a manifold for the finite-dimensional case.

The output of a CGNN is always in a certain space V_j of a MRA and, as such, is as smooth as the scaling function. It would be very interesting to design and study architectures that can generate more irregular, e.g. discontinuous, signals in functions spaces. This would probably require exploiting the full depth of the wavelet MRA, in order to capture arbitrarily fine scales, and keeping at the same time a low-dimensional latent space.

In Chapter 3 of this thesis, we have introduced mixture models of VAEs for learning manifolds of arbitrary topology. The corresponding decoders and encoders of the VAEs provide analytic access to the resulting charts and are learned by a loss function that

approximates the negative log-likelihood function. For minimizing functions F defined on the learned manifold we have proposed a Riemannian gradient descent scheme. In the case of inverse problems, F is chosen as a data-fidelity term. Finally, we have demonstrated the advantages of using several generators on numerical examples.

This work can be extended in several directions. First, gradient descent methods converge only locally and are not necessarily fast. Therefore, it would be interesting to extend the minimization of the functional F in Section 3.8 to higher-order methods or incorporate momentum parameters. Moreover, a careful choice of the initialization could improve the convergence behavior. Further, our reconstruction method could be extended to Bayesian inverse problems. Since the mixture model of VAEs provides us with a probability distribution and an (approximate) density, stochastic sampling methods like the Langevin dynamics could be used for quantifying uncertainties within our reconstructions. Indeed, Langevin dynamics on Riemannian manifolds are still an active area of research. Further, for large numbers K of charts the mixture of VAEs might have a considerable number of parameters. As a remedy, we could incorporate the selection of the chart as conditioning parameter in one conditional decoder-encoder pair, see [213] as a reference for conditional VAEs. Finally, recent papers show that diffusion models provide an implicit representation of the data manifold [31, 194]. It would be interesting to investigate optimization models on such manifolds in order to apply them to inverse problems.

In Chapter 4 of this thesis, we introduced an innovative approach to sparse optimization for inverse problems, leveraging an explicit formula for the MMSE estimator in the context of a mixture of degenerate Gaussian random variables. This methodology, rooted in statistical learning theory, follows a different approach to sparsity promotion compared to the deterministic optimization paradigm (e.g., Lasso) and the Bayesian inverse problem framework. Our reconstruction formula exhibits a notable connection to the self-attention mechanism underlying the transformer architecture, offering an efficient training process: this is useful whenever the mixture model, corresponding to the sparsity properties of the signals of interest, is unknown. This training can be conducted in both supervised and unsupervised modes.

To validate our approach, we conducted numerical implementations (both supervised and unsupervised) and compared them against established baseline algorithms for sparse optimization, such as Lasso, IHT, and their group variants. Additionally, we incorporated a dictionary learning strategy for fair comparisons. The experiments focused on three 1D datasets featuring sparse or compressible signals, addressing denoising and deblurring tasks.

Our findings indicate that the unsupervised method consistently outperforms baseline algorithms in nearly all experiments, demonstrating superior performance with lower computational costs. However, a notable limitation of our work lies in its numerical scalability, particularly concerning larger and higher-dimensional datasets. Addressing this limitation will be a key focus in future research endeavors.

Bibliography

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. Optimization algorithms on matrix manifolds. In *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[2] P.-A. Absil and J. Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012.

[3] J. Adler and O. Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

[4] J. Adler and O. Öktem. Learned primal-dual reconstruction. *IEEE transactions on medical imaging*, 37(6):1322–1332, 2018.

[5] G. S. Alberti, Á. Arroyo, and M. Santacesaria. Inverse problems on low-dimensional manifolds. *Nonlinearity*, 36(1):734, 2022.

[6] G. S. Alberti, P. Campodonico, and M. Santacesaria. Compressed sensing photoacoustic tomography reduces to compressed sensing for undersampled fourier measurements. *SIAM Journal on Imaging Sciences*, 14(3):1039–1077, 2021.

[7] G. S. Alberti, E. De Vito, M. Lassas, L. Ratti, and M. Santacesaria. Learning the optimal tikhonov regularizer for inverse problems. *Advances in Neural Information Processing Systems*, 34:25205–25216, 2021.

[8] G. S. Alberti, J. Hertrich, M. Santacesaria, and S. Sciutto. Manifold learning by mixture models of VAEs for inverse problems. *arXiv preprint arXiv:2303.15244*, 2023.

[9] G. S. Alberti, L. Ratti, M. Santacesaria, and S. Sciutto. Learning a Gaussian mixture for sparsity regularization in inverse problems. *arXiv preprint arXiv:2401.16612*, 2024.

[10] G. S. Alberti and M. Santacesaria. Calderón's inverse problem with a finite number of measurements. In *Forum of Mathematics, Sigma*, volume 7. Cambridge University Press, 2019.

[11] G. S. Alberti and M. Santacesaria. Infinite-dimensional inverse problems with finite measurements. *Archive for Rational Mechanics and Analysis*, 243(1):1–31, 2022.

[12] G. S. Alberti, M. Santacesaria, and S. Sciutto. Continuous generative neural networks. *arXiv preprint arXiv:2205.14627*, 2022.

[13] G. Alessandrini. Stable determination of conductivity by boundary measurements. *Applicable Analysis*, 27(1-3):153–172, 1988.

[14] G. Alessandrini, V. Maarten, R. Gaburro, and E. Sincich. Lipschitz stability for the electrostatic inverse boundary value problem with piecewise linear conductivities. *Journal de Mathématiques Pures et Appliquées*, 107(5):638–664, 2017.

[15] G. Alessandrini and S. Vessella. Lipschitz stability for the inverse conductivity problem. *Advances in Applied Mathematics*, 35(2):207–241, 2005.

[16] D. Allman, A. Reiter, and M. A. L. Bell. Photoacoustic source detection and reflection artifact removal enabled by deep learning. *IEEE transactions on medical imaging*, 37(6):1464–1477, 2018.

[17] F. Altekrüger, A. Denker, P. Hagemann, J. Hertrich, P. Maass, and G. Steidl. Patchnr: Learning from small data by patch normalizing flow regularization. *arXiv preprint arXiv:2205.12021*, 2022.

[18] H. Ammari and H. Kang. *Reconstruction of small inhomogeneities from boundary measurements*, volume 1846 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2004.

[19] B. Amos, L. Xu, and J. Z. Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

[20] J. Andén and S. Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

[21] T. Angles and S. Mallat. Generative networks as inverse problems with scattering transforms. *arXiv preprint arXiv:1805.06621*, 2018.

[22] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

[23] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019.

[24] M. Asim, F. Shamshad, and A. Ahmed. Blind image deconvolution using deep generative priors. *IEEE Transactions on Computational Imaging*, 6:1493–1506, 2020.

[25] A. Aspri, E. Beretta, E. Francini, and S. Vessella. Lipschitz stable determination of polyhedral conductivity inclusions from local boundary measurements. *SIAM J. Math. Anal.*, 54(5):5182–5222, 2022.

[26] K. Astala and L. Päivärinta. Calderón's inverse conductivity problem in the plane. *Annals of Mathematics*, pages 265–299, 2006.

[27] Y. Bai, W. Chen, J. Chen, and W. Guo. Deep learning methods for solving linear inverse problems: Research directions and paradigms. *Signal Processing*, 177:107729, 2020.

[28] D. L. Bailey, M. N. Maisey, D. W. Townsend, and P. E. Valk. *Positron emission tomography*, volume 2. Springer, 2005.

[29] E. Banijamali, A. Ghodsi, and P. Popuart. Generative mixture of networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3753–3760. IEEE, 2017.

[30] F. Bartolucci, E. de Bezenac, B. Raonic, R. Molinaro, S. Mishra, and R. Alaifari. Representation equivalent neural operators: a framework for alias-free operator learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[31] G. Batzolis, J. Stanczuk, and C.-B. Schönlieb. Your diffusion model secretly knows the dimension of the data manifold. *arXiv preprint arXiv:2212.12611*, 2022.

[32] A. Beck. *First-order methods in optimization*. SIAM, 2017.

[33] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019.

[34] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[35] E. Beretta, M. V. de Hoop, E. Francini, S. Vessella, and J. Zhai. Uniqueness and Lipschitz stability of an inverse boundary value problem for time-harmonic elastic waves. *Inverse Problems*, 33(3):035013, 2017.

[36] E. Beretta and E. Francini. Global Lipschitz stability estimates for polygonal conductivity inclusions from boundary measurements. *Appl. Anal.*, 101(10):3536–3549, 2022.

[37] E. Beretta, E. Francini, A. Morassi, E. Rosset, and S. Vessella. Lipschitz continuous dependence of piecewise constant lamé coefficients from boundary data: the case of non-flat interfaces. *Inverse Problems*, 30(12):125005, 2014.

[38] E. Beretta, E. Francini, and S. Vessella. Lipschitz stable determination of polygonal conductivity inclusions in a two-dimensional layered medium from the Dirichlet-to-Neumann map. *SIAM Journal on Mathematical Analysis*, 53(4):4303–4327, 2021.

[39] E. Beretta, S. Micheletti, S. Perotto, and M. Santacesaria. Reconstruction of a piecewise constant conductivity on a polygonal partition via shape optimization in eit. *Journal of Computational Physics*, 353, 2018.

[40] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, 2000.

[41] M. Bertero, P. Boccacci, and C. De Mol. *Introduction to inverse problems in imaging*. CRC press, 2021.

[42] T. Blumensath. Compressed sensing with nonlinear observations and related nonlinear optimization problems. *IEEE Transactions on Information Theory*, 59(6):3466–3474, 2013.

[43] A. Bocchinfuso, D. Calvetti, and E. Somersalo. Bayesian sparsity and class sparsity priors for dictionary learning and coding. *arXiv preprint arXiv:2309.00999*, 2023.

[44] A. Bora, A. Jalal, E. Price, and A. G. Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546. PMLR, 2017.

[45] M. Brand. Charting a manifold. *Advances in Neural Information Processing Systems*, 15, 2002.

[46] M. Brühl and M. Hanke. Numerical implementation of two noniterative methods for locating inclusions by impedance tomography. *Inverse Problems*, 16(4):1029, 2000.

[47] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[48] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale modeling & simulation*, 4(2):490–530, 2005.

[49] T. M. Buzug. Computed tomography. In *Springer handbook of medical technology*, pages 311–342. Springer, 2011.

[50] D. Calvetti and E. Somersalo. *Hierarchical Models and Bayesian Sparsity*, pages 183–210. Springer International Publishing, Cham, 2023.

[51] D. Calvetti, E. Somersalo, and A. Strang. Hierachical Bayesian models and sparsity: ℓ_2-magic. *Inverse Problems*, 35(3):035003, 26, 2019.

[52] F. Camastra and A. Staiano. Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41, 2016.

[53] E. J. Candès and D. L. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise c^2 singularities. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(2):219–266, 2004.

[54] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.

[55] H. Chen, Y. Zhang, W. Zhang, P. Liao, K. Li, J. Zhou, and G. Wang. Low-dose ct via convolutional neural network. *Biomedical optics express*, 8(2):679–694, 2017.

[56] N. Chen, A. Klushyn, F. Ferroni, J. Bayer, and P. Van Der Smagt. Learning flat latent manifolds with VAEs. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2020.

[57] R. T. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019.

[58] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[59] Y. Chen, S. Liu, and X. Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021.

[60] Y. Chen, C.-B. Schönlieb, P. Lio, T. Leiner, P. L. Dragotti, G. Wang, D. Rueckert, D. Firmin, and G. Yang. Ai-based reconstruction for fast mri—a systematic review and meta-analysis. *Proceedings of the IEEE*, 110(2):224–245, 2022.

[61] Z. Chen and J. Zou. An augmented lagrangian method for identifying discontinuous parameters in elliptic systems. *SIAM Journal on Control and Optimization*, 37(3):892–910, 1999.

[62] M. Cheney, D. Isaacson, and J. C. Newell. Electrical impedance tomography. *SIAM review*, 41(1):85–101, 1999.

[63] M. Cheney, D. Isaacson, J. C. Newell, S. Simske, and J. Goble. Noser: An algorithm for solving the inverse conductivity problem. *International Journal of Imaging systems and technology*, 2(2):66–75, 1990.

[64] X. Cheng, X. Chen, and S. Mallat. Deep Haar scattering networks. *Information and Inference: A Journal of the IMA*, 5(2):105–133, 04 2016.

[65] T. Cohn, N. Devraj, and O. C. Jenkins. Topologically-informed atlas learning. *arXiv preprint arXiv:2110.00429*, 2021.

[66] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.

[67] C. O. da Costa-Luis and A. J. Reader. Deep learning for suppression of resolution-recovery artefacts in mlem pet image reconstruction. In *2017 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, pages 1–3. IEEE, 2017.

[68] B. Dai and D. Wipf. Diagnosing and enhancing VAE models. In *International Conference on Learning Representations*, 2019.

[69] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, USA, 1992.

[70] 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*, 57(11):1413–1457, 2004.

[71] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak. Hyperspherical variational auto-encoders. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 856–865. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

[72] M. V. De Hoop, L. Qiu, and O. Scherzer. Local analysis of inverse problems: Hölder stability and iterative reconstruction. *Inverse Problems*, 28(4):045001, 2012.

[73] L. de Oliveira, M. Paganini, and B. Nachman. Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. *Computing and Software for Big Science*, 1(1):1–24, 2017.

[74] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8599–8603. IEEE, 2013.

[75] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2016.

[76] D. L. Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[77] M. Duff, N. D. Campbell, and M. J. Ehrhardt. Regularising inverse problems with generative machine learning models. *arXiv preprint arXiv:2107.11191*, 2021.

[78] E. Dupont, Y. W. Teh, and A. Doucet. Generative models as distributions of functions. *arXiv preprint arXiv:2102.04776*, 2021.

[79] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[80] C. Etmann, R. Ke, and C.-B. Schönlieb. iUNets: learnable invertible up-and downsampling for large-scale inverse problems. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2020.

[81] F. Falck, H. Zhang, M. Willetts, G. Nicholson, C. Yau, and C. C. Holmes. Multi-facet clustering variational autoencoders. *Advances in Neural Information Processing Systems*, 34:8676–8690, 2021.

[82] M. Fan, N. Gu, H. Qiao, and B. Zhang. Intrinsic dimension estimation of data by principal component analysis. *arXiv preprint arXiv:1002.2050*, 2010.

[83] Y. Fan and L. Ying. Solving electrical impedance tomography with deep learning. *Journal of Computational Physics*, 404:109119, 2020.

[84] J. Feldman, M. Salo, and G. Uhlmann. The calderón problem—an introduction to inverse problems. *Preliminary notes on the book in preparation*, 2019.

[85] D. Floryan and M. D. Graham. Charts and atlases for nonlinear data-driven models of dynamics on manifolds. *arXiv preprint arXiv:2108.05928*, 2021.

[86] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer graphics and Applications*, 22(2):56–65, 2002.

[87] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[88] M. Genzel, J. Macdonald, and M. März. Solving inverse problems with deep neural networks–robustness included? *IEEE transactions on pattern analysis and machine intelligence*, 45(1):1119–1134, 2022.

[89] C. L. Giles and T. Maxwell. Learning, invariance, and generalization in high-order neural networks. *Applied optics*, 26(23):4972–4978, 1987.

[90] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. In *Handbook for Automatic Computation: Volume II: Linear Algebra*, pages 134–151. Springer, 1971.

[91] M. González, A. Almansa, and P. Tan. Solving inverse problems by joint posterior maximization with autoencoding prior. *SIAM Journal on Imaging Sciences*, 15(2):822–859, 2022.

[92] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[93] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[94] W. B. Gordon. On the diffeomorphisms of Euclidean space. *The American Mathematical Monthly*, 79(7):755–759, 1972.

[95] A. Goujon, S. Neumayer, P. Bohra, S. Ducotterd, and M. Unser. A neural-network-based convex regularizer for image reconstruction. *arXiv preprint arXiv:2211.12461*, 2022.

[96] L. Grafakos. *Classical fourier analysis*, volume 2. Springer, 2008.

[97] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2018.

[98] R. Gribonval. Should penalized least squares regression be interpreted as maximum a posteriori estimation? *IEEE Transactions on Signal Processing*, 59(5):2405–2410, 2011.

[99] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[100] D. Güera and E. J. Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2018.

[101] A. Habring and M. Holler. A generative variational model for inverse problems in imaging. *SIAM Journal on Mathematics of Data Science*, 4(1):306–335, 2022.

[102] A. Habring and M. Holler. A note on the regularity of images generated by convolutional neural networks. *arXiv preprint arXiv:2204.10588*, 2022.

[103] J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, pages 49–52, 1902.

[104] P. Hagemann, J. Hertrich, and G. Steidl. Stochastic normalizing flows for inverse problems: a markov chains viewpoint. *SIAM/ASA Journal on Uncertainty Quantification*, 10(3):1162–1190, 2022.

[105] P. Hagemann and S. Neumayer. Stabilizing invertible neural networks using mixture models. *Inverse Problems*, 37(8):Paper No. 085002, 23, 2021.

[106] P. L. Hagemann, J. Hertrich, and G. Steidl. *Generalized normalizing flows via Markov chains*. Cambridge University Press, 2023.

[107] E. T. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing. *CAAM TR07-07, Rice University*, 43:44, 2007.

[108] S. J. Hamilton, A. Hänninen, A. Hauptmann, and V. Kolehmainen. Beltrami-net: domain-independent deep d-bar learning for absolute imaging with electrical impedance tomography (a-eit). *Physiological measurement*, 40(7):074002, 2019.

[109] S. J. Hamilton and A. Hauptmann. Deep d-bar: Real-time electrical impedance tomography imaging with deep neural networks. *IEEE transactions on medical imaging*, 37(10):2367–2377, 2018.

[110] C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama. GAN-based synthetic brain MR image generation. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 734–738. IEEE, 2018.

[111] Y. Han and J. C. Ye. Framing u-net via deep convolutional framelets: Application to sparse-view ct. *IEEE transactions on medical imaging*, 37(6):1418–1429, 2018.

[112] P. C. Hansen, J. G. Nagy, and D. P. O'leary. *Deblurring images: matrices, spectra, and filtering*. SIAM, 2006.

[113] B. Harrach. Uniqueness and lipschitz stability in electrical impedance tomography with finitely many electrodes. *Inverse Problems*, 35(2):024005, 2019.

[114] E. Hernandez and G. Weiss. *A First Course on Wavelets*. Studies in Advanced Mathematics. CRC Press, 1996.

[115] J. Hertrich. Proximal residual flows for bayesian inverse problems. *arXiv preprint arXiv:2211.17158*, 2022.

[116] J. Hertrich, A. Houdard, and C. Redenbach. Wasserstein patch prior for image superresolution. *IEEE Transactions on Computational Imaging*, 8:693–704, 2022.

[117] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[118] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung. Multi-generator generative adversarial nets. *arXiv preprint arXiv:1708.02556*, 2017.

[119] A. Holub and P. Perona. A discriminative framework for modelling object classes. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 664–671. IEEE, 2005.

[120] X. Hong, Y. Zan, F. Weng, W. Tao, Q. Peng, and Q. Huang. Enhancing the image quality via transferred deep residual learning of coarse pet sinograms. *IEEE transactions on medical imaging*, 37(10):2322–2332, 2018.

[121] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR, 2018.

[122] C. M. Hyun, S. H. Baek, M. Lee, S. M. Lee, and J. K. Seo. Deep learning-based solvability of underdetermined inverse problems in medical imaging. *Medical Image Analysis*, 69:101967, 2021.

[123] 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*, 63(13):135007, jun 2018.

[124] M. Ikehata and S. Siltanen. Numerical method for finding the convex hull of an inclusion in conductivity from boundary measurements. *Inverse Problems*, 16(4):1043, 2000.

[125] A. J. Izenman. Introduction to manifold learning. *WIREs Computational Statistics*, 4(5):439–446, 2012.

[126] T. Jebara. *Machine learning: discriminative and generative*, volume 755. Springer Science & Business Media, 2012.

[127] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2017.

[128] 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*, 26(9):4509–4522, 2017.

[129] U. S. Kamilov, H. Mansour, and B. Wohlberg. A plug-and-play priors approach for solving nonlinear imaging inverse problems. *IEEE Signal Processing Letters*, 24(12):1872–1876, 2017.

[130] E. Kang, J. Min, and J. C. Ye. A deep convolutional neural network using directional wavelets for low-dose x-ray ct reconstruction. *Medical physics*, 44(10):e360–e375, 2017.

[131] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[132] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[133] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[134] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[135] D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[136] A. Kirsch et al. *An introduction to the mathematical theory of inverse problems*, volume 120. Springer, 2011.

[137] A. Kirsch and N. Grinberg. *The factorization method for inverse problems*, volume 36. OUP Oxford, 2007.

[138] B. Kivva, G. Rajendran, P. Ravikumar, and B. Aragam. Identifiability of deep generative models without auxiliary information. *Advances in Neural Information Processing Systems*, 35:15687–15701, 2022.

[139] E. O. Korman. Autoencoding topology. *arXiv preprint arXiv:1803.00156*, 2018.

[140] E. O. Korman. Atlas based representation and metric learning on manifolds. *arXiv preprint arXiv:2106.07062*, 2021.

[141] K. Kothari, A. Khorashadizadeh, M. de Hoop, and I. Dokmanić. Trumpets: Injective flows for inference and inverse problems. In *Uncertainty in Artificial Intelligence*, pages 1269–1278. PMLR, 2021.

[142] N. B. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. M. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, 24(89):1–97, 2023.

[143] A. Kundu, S. Chatterjee, A. S. Murthy, and T. Sreenivas. Gmm based bayesian approach to speech enhancement in signal/transform domain. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4893–4896. IEEE, 2008.

[144] D. Labate, W.-Q. Lim, G. Kutyniok, and G. Weiss. Sparse multidimensional representation using shearlets. In *Wavelets XI*, volume 5914, page 59140U. International Society for Optics and Photonics, 2005.

[145] J. A. Lasserre, C. M. Bishop, and T. P. Minka. Principled hybrids of generative and discriminative models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 87–94. IEEE, 2006.

[146] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[147] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[148] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[149] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. *Advances in neural information processing systems*, 19, 2006.

[150] H. Lee, J. Lee, H. Kim, B. Cho, and S. Cho. Deep-neural-network-based sinogram synthesis for sparse-view ct image reconstruction. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 3(2):109–119, 2018.

[151] Q. Lei, A. Jalal, I. S. Dhillon, and A. G. Dimakis. Inverting deep generative models, one layer at a time. *Advances in neural information processing systems*, 32, 2019.

[152] E. Levina and P. Bickel. Maximum likelihood estimation of intrinsic dimension. *Advances in neural information processing systems*, 17, 2004.

[153] H. Li, J. Schwab, S. Antholzer, and M. Haltmeier. Nett: Solving inverse problems with deep neural networks. *Inverse Problems*, 36(6):065005, 2020.

[154] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson. Deep learning for hyperspectral image classification: An overview. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):6690–6709, 2019.

[155] S. Li, H. Yin, and L. Fang. Group-sparse representation with dictionary learning for medical image denoising and fusion. *IEEE Transactions on biomedical engineering*, 59(12):3450–3459, 2012.

[156] Y. Li, Q. Pan, S. Wang, T. Yang, and E. Cambria. A generative model for category text generation. *Information Sciences*, 450:301–315, 2018.

[157] Z.-P. Liang and P. C. Lauterbur. *Principles of magnetic resonance imaging*. SPIE Optical Engineering Press Belllingham, WA, 2000.

[158] F. Locatello, D. Vincent, I. Tolstikhin, G. Rätsch, S. Gelly, and B. Schölkopf. Competitive training of mixtures of independent deep generative models. *arXiv preprint arXiv:1804.11130*, 2018.

[159] A. Logg and G. N. Wells. Dolfin: Automated finite element computing. *ACM Transactions on Mathematical Software (TOMS)*, 37(2):1–28, 2010.

[160] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, and S. Yan. Robust and efficient subspace segmentation via least squares regression. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VII 12*, pages 347–360. Springer, 2012.

[161] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[162] S. Lunz, O. Öktem, and C.-B. Schönlieb. Adversarial regularizers in inverse problems. *Advances in neural information processing systems*, 31, 2018.

[163] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly. Compressed sensing MRI. *IEEE signal processing magazine*, 25(2):72–82, 2008.

[164] Y. Ma and Y. Fu. *Manifold learning theory and applications*. CRC press, 2011.

[165] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696, 2009.

[166] S. Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Academic Press, Inc., USA, 3rd edition, 2008.

[167] S. Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[168] N. Mandache. Exponential instability in an inverse problem for the Schrödinger equation. *Inverse Problems*, 17(5):1435, 2001.

[169] P. Mantini and S. K. Shah. Cqnn: Convolutional quadratic neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9819–9826. IEEE, 2021.

[170] M. Mardani, E. Gong, J. Y. Cheng, S. S. Vasanawala, G. Zaharchuk, L. Xing, and J. M. Pauly. Deep generative adversarial neural networks for compressive sensing MRI. *IEEE Transactions on Medical Imaging*, 38(1):167–179, 2019.

[171] M. Mardani, Q. Sun, D. Donoho, V. Papyan, H. Monajemi, S. Vasanawala, and J. Pauly. Neural proximal gradient descent for compressive imaging. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[172] P. Massa, S. Garbarino, and F. Benvenuto. Approximation of discontinuous inverse operators with neural networks. *Inverse Problems*, 38(10):Paper No. 105001, 16, 2022.

[173] E. Mathieu, C. Le Lan, C. J. Maddison, R. Tomioka, and Y. W. Teh. Continuous hierarchical representations with Poincaré variational auto-encoders. *Advances in neural information processing systems*, 32, 2019.

[174] J. L. Mueller and S. Siltanen. *Linear and nonlinear inverse problems with practical applications*. SIAM, 2012.

[175] S. Mukherjee, S. Dittmer, Z. Shumaylov, S. Lunz, O. Öktem, and C.-B. Schönlieb. Learned convex regularizers for inverse problems. *arXiv preprint arXiv:2008.02839*, 2020.

[176] 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*, 1(1):39–56, 2020.

[177] D. Onken, S. W. Fung, X. Li, and L. Ruthotto. OT-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9223–9232, 2021.

[178] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[179] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri, and R. Verheyen. Event generation and statistical sampling for physics with deep generative models and a density information buffer. *Nature communications*, 12(1):1–16, 2021.

[180] A. Oussidi and A. Elhassouny. Deep generative models: Survey. In *2018 International conference on intelligent systems and computer vision (ISCV)*, pages 1–8. IEEE, 2018.

[181] N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

[182] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[183] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[184] E. Pineau and M. Lelarge. InfoCatVAE: representation learning with categorical variational autoencoders. *arXiv preprint arXiv:1806.08240*, 2018.

[185] N. Pitelis, C. Russell, and L. Agapito. Learning a manifold as an atlas. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1642–1649, 2013.

[186] B. Pourciau. Global invertibility of nonsmooth mappings. *Journal of mathematical analysis and applications*, 131(1):170–179, 1988.

[187] M. Puthawala, K. Kothari, M. Lassas, I. Dokmanić, and M. de Hoop. Globally injective ReLU networks. *arXiv e-prints*, pages arXiv–2006, 2020.

[188] M. Puthawala, M. Lassas, I. Dokmanić, and M. de Hoop. Universal joint approximation of manifolds and densities by simple injective flows. *arXiv preprint arXiv:2110.04227*, 2021.

[189] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[190] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[191] L. A. P. Rey, V. Menkovski, and J. W. Portegies. Diffusion variational autoencoders. In *29th International Joint Conference on Artificial Intelligence-17th Pacific Rim International Conference on Artificial Intelligence.*, 2020.

[192] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[193] Y. Romano, M. Elad, and P. Milanfar. The little engine that could: Regularization by denoising (red). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017.

[194] B. L. Ross, G. Loaiza-Ganem, A. L. Caterini, and J. C. Cresswell. Neural implicit manifold learning for topology-aware generative modelling. *arXiv preprint arXiv:2206.11267*, 2022.

[195] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[196] L. Ruthotto and E. Haber. An introduction to deep generative modeling. *GAMM-Mitteilungen*, 44(2):e202100008, 2021.

[197] R. Salakhutdinov. Learning deep generative models. *Annual Review of Statistics and Its Application*, 2:361–385, 2015.

[198] J. Scarlett, R. Heckel, M. R. Rodrigues, P. Hand, and Y. C. Eldar. Theoretical perspectives on deep learning methods in inverse problems. *IEEE journal on selected areas in information theory*, 3(3):433–453, 2022.

[199] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer, 2017.

[200] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price, and D. Rueckert. A deep cascade of convolutional neural networks for dynamic mr image reconstruction. *IEEE transactions on Medical Imaging*, 37(2):491–503, 2017.

[201] S. Schonsheck, J. Chen, and R. Lai. Chart auto-encoders for manifold structured data. *arXiv preprint arXiv:1912.10094*, 2019.

[202] T. Schuster, B. Kaltenbacher, B. Hofmann, and K. S. Kazimierski. Regularization methods in banach spaces. In *Regularization Methods in Banach Spaces*. de Gruyter, 2012.

[203] J. Schwab, S. Antholzer, and M. Haltmeier. Deep null space learning for inverse problems: convergence analysis and rates. *Inverse Problems*, 35(2):025008, 2019.

[204] J. K. Seo, K. C. Kim, A. Jargal, K. Lee, and B. Harrach. A learning-based method for solving ill-posed nonlinear inverse problems: a simulation study of lung EIT. *SIAM journal on Imaging Sciences*, 12(3):1275–1295, 2019.

[205] A. N. Shiryaev. *Probability-1*, volume 95. Springer, 2016.

[206] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis. Model-based deep learning. *Proceedings of the IEEE*, 2023.

[207] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[208] S. Sidheekh, C. B. Dock, T. Jain, R. Balan, and M. K. Singh. VQ-Flows: Vector quantized local normalizing flows. *arXiv preprint arXiv:2203.11556*, 2022.

[209] S. Siltanen, J. Mueller, and D. Isaacson. An implementation of the reconstruction algorithm of a nachman for the 2d inverse conductivity problem. *Inverse Problems*, 16(3):681, 2000.

[210] N. K. Singh and K. Raza. Medical image generation using generative adversarial networks: a review. *Health Informatics: A Computational Perspective in Healthcare*, pages 77–96, 2021.

[211] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[212] I. Skorokhodov, S. Ignatyev, and M. Elhoseiny. Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10753–10764, 2021.

[213] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing Systems*, 28, 2015.

[214] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[215] J. Stolberg-Larsen and S. Sommer. Atlas generative models and geodesic interpolation. *Image and Vision Computing*, page 104433, 2022.

[216] A. Tamburrino and G. Rubinacci. A new non-iterative inversion method for electrical resistance tomography. *Inverse Problems*, 18(6):1809, 2002.

[217] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[218] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin. Deep learning on image denoising: An overview. *Neural Networks*, 131:251–275, 2020.

[219] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[220] A. N. Tikhonov. On the solution of ill-posed problems and the method of regularization. In *Doklady akademii nauk*, volume 151, pages 501–504. Russian Academy of Sciences, 1963.

[221] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[222] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[223] W. P. Walters and R. Barzilay. Applications of deep learning in molecule generation and molecular property prediction. *Accounts of chemical research*, 54(2):263–270, 2020.

[224] S. Wang, Z. Su, L. Ying, X. Peng, S. Zhu, F. Liang, D. Feng, and D. Liang. Accelerating magnetic resonance imaging via deep learning. In *2016 IEEE 13th international symposium on biomedical imaging (ISBI)*, pages 514–517. IEEE, 2016.

[225] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[226] Y. Xu, K. Lin, S. Wang, L. Wang, C. Cai, C. Song, L. Lai, and J. Pei. Deep learning for molecular generation. *Future medicinal chemistry*, 11(6):567–597, 2019.

[227] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui, and M.-H. Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

[228] Y. Yao, S. M. Kang, W. Jigang, P.-X. Yang, et al. A regularized gradient projection method for the minimization problem. *Journal of Applied Mathematics*, 2012, 2012.

[229] F. Ye and A. G. Bors. Deep mixture generative autoencoders. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[230] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018.

[231] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[232] L. Zelnik-Manor, K. Rosenblum, and Y. C. Eldar. Dictionary optimization for block-sparse representations. *IEEE Transactions on Signal Processing*, 60(5):2386–2395, 2012.

[233] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[234] K. Zhang, W. Ren, W. Luo, W.-S. Lai, B. Stenger, M.-H. Yang, and H. Li. Deep image deblurring: A survey. *International Journal of Computer Vision*, 130(9):2103–2130, 2022.

[235] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1):1–38, 2019.

[236] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[237] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487–492, 2018.