Alma Mater Studiorum–Università di Bologna

DOTTORATO DI RICERCA IN MATEMATICA
Ciclo XXXVI

Settore Concorsuale: 01/A5 - ANALISI NUMERICA
Settore Scientifico Disciplinare: MAT/08 - ANALISI NUMERICA

Regularization meets GreenAI: a new framework for image reconstruction in life sciences applications

Presentata da: Davide Evangelista

Coordinatore Dottorato:
Prof.ssa Valeria Simoncini

Supervisore:
Prof.ssa Elena Loli Piccolomini

Esame finale anno accademico 2023/2024

Alla Benni, mia compagna di vita,
Ai miei genitori, mio fratello,
Ai miei amici, Vito, Andre, Mello...

ABSTRACT

Ill-conditioned inverse problems frequently arise in life sciences, particularly in the context of image deblurring and medical image reconstruction. These problems have been addressed through iterative variational algorithms, which regularize the reconstruction by adding prior knowledge about the problem's solution. Despite the theoretical reliability of these methods, their practical utility is constrained by the time required to converge. Recently, the advent of neural networks allowed the development of reconstruction algorithms that can compute highly accurate solutions with minimal time demands. Regrettably, it is well-known that neural networks are sensitive to unexpected noise, and the quality of their reconstructions quickly deteriorates when the input is slightly perturbed. Modern efforts to address this challenge have led to the creation of massive neural network architectures, but this approach is unsustainable from both ecological and economic standpoints. The recently introduced GreenAI paradigm argues that developing sustainable neural network models is essential for practical applications.

In this thesis, we aim to bridge the gap between theory and practice by introducing a novel framework that combines the reliability of model-based iterative algorithms with the speed and accuracy of end-to-end neural networks. Additionally, we demonstrate that our framework yields results comparable to state-of-the-art methods while using relatively small, sustainable models.

In the first part of this thesis, we discuss the proposed framework from a theoretical perspective. We provide an extension of classical regularization theory, applicable in scenarios where neural networks are employed to solve inverse problems, and we show there exists a trade-off between accuracy and stability. Furthermore, we demonstrate the effectiveness of our methods in common life science-related scenarios.

In the second part of the thesis, we initiate an exploration extending the proposed method into the probabilistic domain. We analyze some properties of deep generative models, revealing their potential applicability in addressing ill-posed inverse problems.

Keywords: inverse problems, model-based methods, convolutional neural networks, greenAI, computed tomography, image deblurring, life sciences, neural network stability, accuracy stability trade-off.

Contents

List of Acronyms

ADMM	Alternating Direction Method of Multipliers
AI	Artificial Intelligence
BCCB	Block Circulant with Circulant Block
CG	Conjugate Gradient
CGLS	Conjugate Gradient method for Least Squares
CNN	Convolutional Neural Network
CP	Chambolle-Pock
CT	Computed Tomography
DDIM	Diffusion Denoising Implicit Model
DDPM	Diffusion Denoising Probabilistic Model
DGM	Deep Generative Model
DGP	Deep Generative Prior
DLVM	Deep Latent Variable Model
DPC	Discrete Picard Condition
ELBO	Evidence Lower Bound
FBP	Filtered Back Projection
GAN	Generative Adversarial Network
LPP	Learnt Post-Processing
MAP	Maximum a-posteriori
MLE	Maximum likelihood estimation
MRI	Magnetic Resonance Imaging
MSE	Mean Squared Error
NF	Normalizing Flow
NN	Neural Network
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
SGD	Stochastic Gradient Descent
SparseCT	Sparse Computed Tomography
SSIM	Structural Similarity Index Measure
$\mathrm{T}_p\mathrm{V}$	Total p-Variation
TV	Total Variation
VAE	Variational AutoEncoder

Introduction

Our ability to comprehend reality is confined to the things we can *measure*. To illustrate this point, let's imagine a scenario similar to Plato's Cave Allegory. In this imaginary setup, there's a group of people confined in a cave, facing a blank wall. Behind them, a source of light casts shadows of various objects onto the wall in front. It's quite evident that they won't be able to gain a complete understanding of the actual objects being projected. For instance, they can't determine the object's material or the texture of its surface because this information isn't included in the projected shadow. Their comprehension is limited to what they can deduce from the information contained in the shadows they observe, along with their pre-existing knowledge of reality. In particular, the problem of inferring properties of an object of interest by *observable* quantities, obtained by *experiments*, is usually referred to as *inverse problem*.

Inverse problems consistently appear in life sciences. For example in X-ray computed tomography, where an image representing the interior of a body is inferred by a collection of the intensities of X-ray beams that have been passed through the body, or in Magnetic Resonance Imaging (MRI), where the resonance frequency of a strong magnetic field applied to the body of a patient is used to estimate the composition of the tissues, or in microscopy and astronomy, where the available information are inevitably blurred due to physical limitation of the measurement system, and clear, sharp details has to be recovered by inference. Practical solutions of inverse problems require advanced mathematical techniques, mainly based on a formalization of the measurement process and on the prior knowledge on the solution. Recently, the introduction of powerful data-driven methods such as neural networks, opened the chance of developing algorithms able to infer important properties of the object of interest with very few measurements. However, a clear mathematical understanding of data-driven methods is still lacking in the literature, with the consequence that those algorithms are unreliable and, practically, unused. Moreover, the exponential growth of neural network architecture required to improve their prediction ability introduces sustainability problems, both from an ecological point of view (due to the carbon dioxide emission caused by the computation devices) and from an economical point of view, which limits the applicability of the most advanced methods. The newborn field of GreenAI, discussed in greater detail in the following, aims to shed light on these challenges, focusing on the sustainability of the neural network-based algorithm rather than their performance.

In this thesis, we focus on two case studies, namely image deblurring and sparse computed tomography. We begin with classical regularization methods that are addressed using optimization techniques. Subsequently, we explore the integration of these methods with data-driven approaches, with the objective of leveraging the strengths of both variational and deep learning methods. To this aim, we analyze the reliability of modern data-driven methods solving the associated inverse problem and we propose some methods aiming to solve these challenges, guided by the paradigm of ecological and economical sustainability proposed by the GreenAI literature.

In the following, we will describe the problems considered in this thesis, with the intent of analyzing their importance in the field of life sciences, and we describe the advantages and disadvantages of modern algorithms employed to solve these problems. Finally, we will introduce the methods proposed in this thesis which will be developed in detail in the next chapters.

Image Deblurring

As already mentioned, image deblurring is an important inverse problem appearing in many fields of life sciences, such as cell microscopy, astronomy, and other related topics such as medical imaging. Talking about microscopy, when the object of interest is too small to be measured directly, it is necessary to acquire it by systems such as the light microscope. However, a portion of information gets lost during the acquisition process due to several reasons, including:

- **Diffraction Limit**: due to the wave nature of light, when the measured object has a resolution that is smaller than the wavelength of the light used by the optical system, the acquired object appears blurred;
- **Low Light Levels**: microscopical objects are often acquired with inadequate illumination levels, resulting in noisy and blurred acquisition;
- **Movement of the object**: when the optical microscope is used to measure alive cells, the micro-movement of the object causes motion blur artifact in the acquisition.

Whatever the reason, the acquired image often appears blurred, and advanced mathematical tools are required to recover the information on the true, sharp image, to be able to see the details of the object of interest.

Astronomical images acquired by telescopes shows similar blurring artifact, mainly due to:

- **Atmospheric Turbulence**: as the light coming from astronomical objects passes through the Earth's atmosphere, it encounters variations in air density and temperature, causing the light to refract and scatter. This turbulence results in a blurring effect corrupting the acquisition;

- **Movement of the object**: due to the Earth's rotation and movement of the celestial object, the acquisition from telescopes often shows motion blur artifacts that reduce the amount of details observable by the image.

To understand the impact of the blurring effect on the measured data, Figure 1 shows the real acquisition of a microscopical and an astronomical image. Both the acquisitions lack important details, that have to be reconstructed by advanced mathematical techniques.

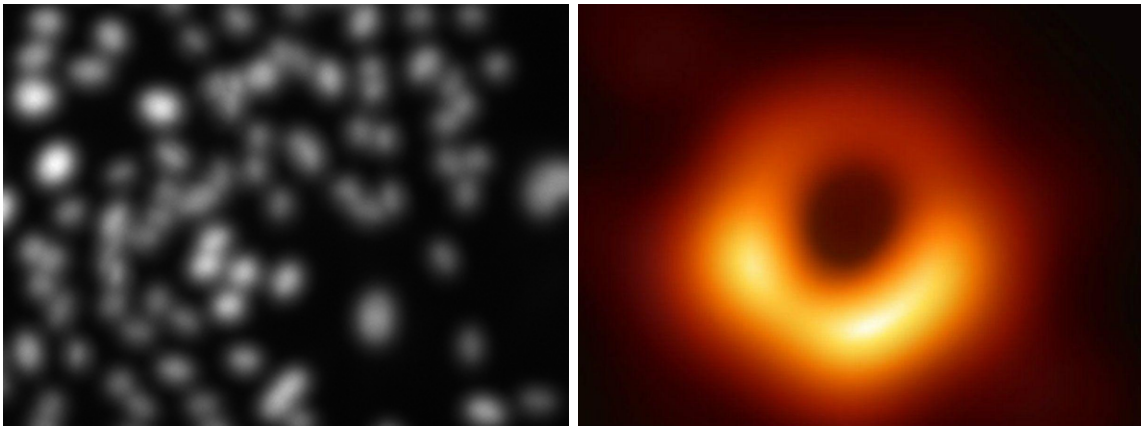


Figure 1: An example of blurred acquisition in microscopy and astronomy, taken from the web. *Left.* The acquisition of a group of cells by a light microscope. *Right.* The first acquisition of a black hole. The blurring effect is clearly visible in both the images.

Computed Tomography

The importance of X-ray computed tomography in life sciences is undoubted. Since its invention at the beginning of 1900, it contributed to saving an uncountable number of human lives, due to its application in medical diagnosis. The way it works is very simple: an X-ray emitting source is positioned on one side of a patient, in such a way that the radiation, emitted at a known intensity, passes through the body and gets collected by a detector, placed on the opposite side of the patient itself. Since the tissues absorb a portion of the emitted photons in a number proportional to their density, the amount of radiation measured by the detector is smaller than it was when emitted. By acquiring multiple emissions at different angles, it is possible to accurately reconstruct the density map of the interior of the body in analysis and, consequently, to spot some alterations such as tumors or other unexpected objects.

However, the radiation passing through the cells of the body can cause illness after multiple acquisitions. For this reason, it is common in modern medical protocols to try to reduce as much as possible the amount of radiation emitted on the patient, to a point that the amount of information collected is not sufficient to accurately reconstruct the structure of the body, resulting in an acquisition that shows severe streaking artifacts. In this setup, known

as sparse computed tomography (SparseCT in the following), it is of primary importance to develop mathematical methods to improve the quality of the reconstructed images, with the intent of simplifying the identification of the diagnosis by the doctor.

The reconstruction task is made even harder by the limitations imposed by the medical protocols. In particular, the algorithm must be:

- **Efficient**: the method has to be applied on very high-resolution images, usually containing tenths of millions of pixels. For this reason, it has to be efficient enough to scale with the dimensionality. This is a difficult challenge since, most of the time, accurate reconstruction algorithms require high computational time to be applied.

- **Fast**: this is especially true when it has to be employed for real-time reconstruction, such as when the tomographic acquisition is used to assist the doctor in surgery.

- **Explainable:** to use a reconstruction algorithm on real medical applications, it has to be certificated. This is done by a procedure involving a large number of experts who have to understand how the method works and what are its weaknesses. The introduction of a new reconstruction technique thus requires it to be simple to explain and understand.

- **Stable**: since the computed tomography acquisitions naturally present irregularities, the method should be stable to modifications of the physical structure of the patient, and to unexpected noise and blur that can appear due to problems with the device and/or movement of the patient.

- **Sustainable**: since the number of daily acquisitions in an hospital is usually large, the reconstruction method has to be sustainable, both in terms of carbon emissions and economic requirements.

The listed requirements clarify that the mere introduction of advanced neural network techniques is not sufficient to guarantee the applicability of the method to the real medical framework. For this reason, we remark on the importance of the perspective of the GreenAI and ExplainableAI fields to develop powerful methods satisfying the requirements imposed by real medical applications. For example, Figure 2 shows the reconstruction of a computed tomography acquisition in a SparseCT setup, performed by a very fast but inaccurate method known as Filtered Back-Projection (FBP), and by a slow but very accurate iterative algorithm. Note that, even if the details are better visible in the image on the right, modern medical protocols still utilize the FBP method, since it better fits the medical constraint listed above.

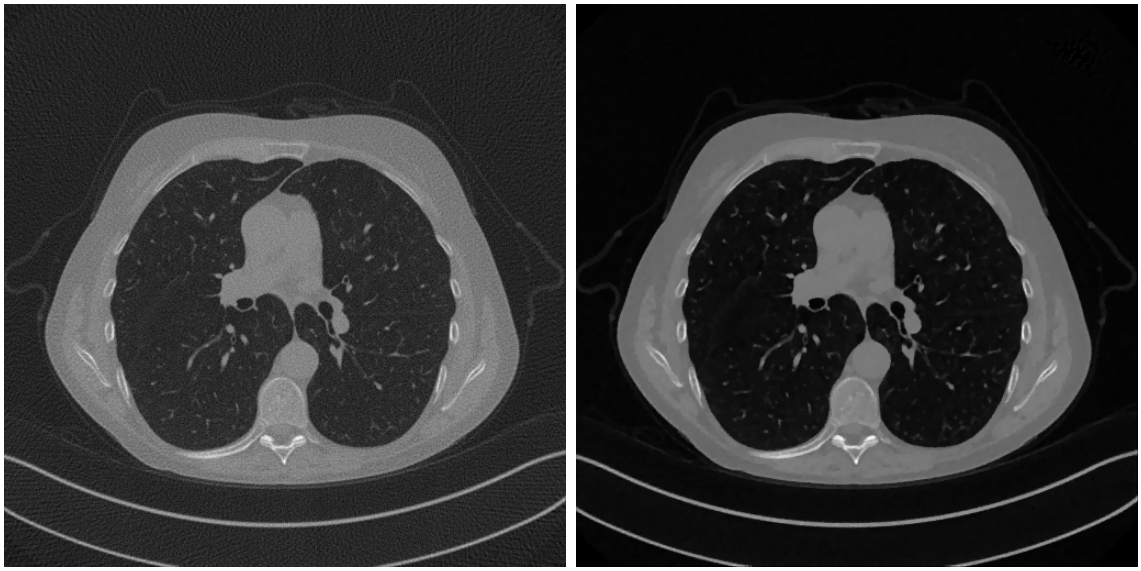


Figure 2: An example of a SparseCT reconstruction with *Left.* the fast but inaccurate FBP algorithm, and *Right.* a slow but accurate iterative algorithm.

Mathematical tools to solve inverse problems

Developing a good algorithm to solve inverse problems is probably even more important than introducing new techniques to acquire measurements. To this aim, advanced mathematical tools have been developed over the decades, leading to an extensive set of techniques, each with its unique properties. Among all the methods, we can distinguish a category that excels on the others for the accuracy of the reconstruction and the stability achieved: the class of model-based algorithms. Briefly, a model-based algorithm considers a mathematical model imitating the measurement process that maps the object of interest to the observable data and returns an estimation of the object by inverting this process. For example, both the Filtered Back-Projection (FBP) and the iterative methods described above are examples of model-based algorithms. The main difference between the two is that the latter assumes knowledge on the solution to interpolate the information contained in the data and generate better solutions. Such kinds of algorithms, known as regularized variational methods and described in detail in the following chapters, despite the great accuracy and stability that they show in practical application, are rarely used in practice. This happens because of two main reasons: first, the quality of the solution is strictly related to the mathematical approximation of the measurement process and the considered prior knowledge of the solution and second, the reconstruction involves highly computationally expensive algorithms that are of limited use for real-time applications such as computed tomography.

Recently, the introduction of neural networks in the field of inverse problems allowed the development of very fast and accurate algorithms to solve reconstruction problem, that does not require any mathematical model for the measurement process. We will refer to this class of methods as data-driven algorithms since they make use of large datasets to be tuned for

the specific application of interest. However, their use is still limited in practice, because of multiple challenges that are still unsolved. In particular, since they are purely trained on data and no knowledge on the physical measurement process is taken into consideration, they are prone to unwanted bias present in the collected data, and the details of how they work are still unexplainable from a mathematical point of view. Moreover, it is known that they suffer from severe stability problems, mainly due to the way they are trained and on properties of the measurement operator.

A modern tentative solution to the challenges described above is to make the model more and more complex, to be able to learn from a bigger amount of data, and, consequently, to reduce the impact of the described issues. Consider for example the plot represented in Figure 3, taken from [1], that shows the number of free parameters (a quantity that is proportional to the computational complexity of the model) of state-of-the-art neural networks over the last decade, in logarithm scale. Note that the number of parameters has grown consistently over the years, with an exceptional value of more than 10^{11} for the modern GPT-3 algorithm. In [2], the authors showed that the computational complexity of modern neural networks reflects in an unsustainable growth of the carbon dioxide emission required to give electricity to the devices dedicated to the training of the models. The exponential increase in the number of parameters also reflects in an increase in the economical requirements to train the models, cutting off everyone who is not able to sustain the expenses, researchers included. For this reason, the authors in [2] propose a new paradigm for neural networks, where the computational complexity is taken into consideration together with their accuracy, named *GreenAI*.

Contributions

Aware of the limitations described above, in this thesis we propose a new framework to solve inverse problems related to life sciences. In particular, we consider a hybrid method, that exploits the stability offered by model-based iterative algorithms, with the reconstruction quality typical of end-to-end neural networks. By taking the physics of the model into consideration as in regularized iterative methods, the resulting model exhibits increased consistency and stability to unexpected noise in the data. Moreover, the reconstruction problem is simplified by the model-based method, and the neural network requires fewer parameters to obtain the same accuracy as classical end-to-end models. This makes the framework more sustainable, as described in GreenAI literature, and reliable, as required for life sciences applications, with minimal to no accuracy loss.

In particular, the framework is presented in Chapter 2 where, after a discussion on the limitations of the classical regularization theory described in [3] when applied to end-to-end neural networks, we propose an extension that allows to theoretically describe the stability and accuracy performance of data-driven solvers to inverse problems. Moreover, in the same

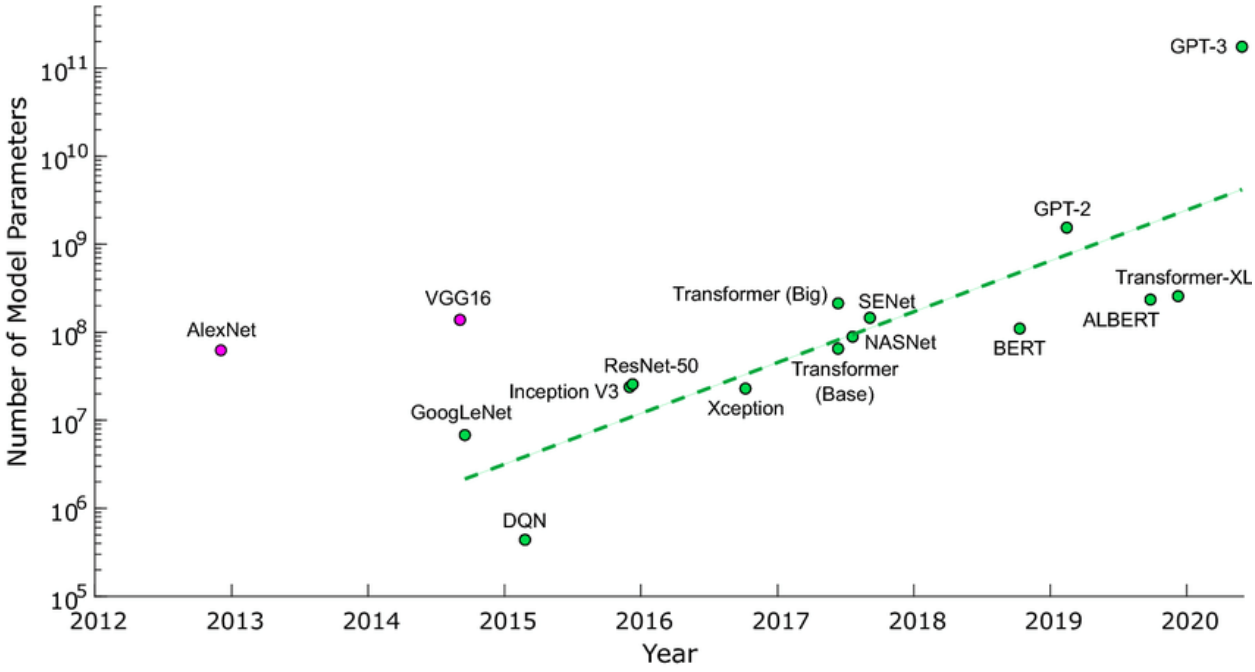


Figure 3: A plot representing the total number of parameters in state-of-the-art neural network models over the year, showing that the performance gain correlates with the increase in dimension of the model. The y-axis is represented in logarithmic scale. Image taken from [1].

Chapter, we analyze the proposed framework from a theoretical perspective and we perform extended empirical experiments on the image deblurring inverse problem. This analysis suggests that the framework is stable and widely applicable to multiple realistic life science-related inverse problems. Moreover, the algorithm can be executed on a very basic device, such as a personal computer.

The method is then tested on a SparseCT inverse problem in Chapter 3, where the framework shows its potential to recover accurate solutions to complex medical protocols.

In Chapter 4, we extend the method to non-convex reconstruction problems, such as Total p-Variation regularization.

Finally, in the last part of the thesis, we begin an analysis of the sustainability of probabilistic neural network-based methods, named *generative models*, that aims to set the basis for an extension of the hybrid technique described above to stochastic models, that are known to perform better in real applications and shows superior explainability. An extended discussion about this point is left to the end of the thesis.

List of Publications

[4] *A. Asperti, D. Evangelista, E. Loli Piccolomini*, **A survey on Variational Autoencoders from a GreenAI perspective**, SN Computer Science, 1st March 2021;

[5] *A. Asperti, D. Evangelista, M. Marzolla*, **Dissecting FLOPs along input dimensions for GreenAI cost estimations**, International Conference on Machine Learning, Optimization, and Data Science, 2021;

[6] *E. Morotti, D. Evangelista, E. Loli Piccolomini*, **A green prospective for learned post-processing in sparse-view tomographic reconstruction**, Journal of Imaging, 2021;

[7] *E. Morotti, D. Evangelista, E. Loli Piccolomini*, **RISING a new framework for few-view tomographic image reconstruction with deep learning**, Computerized Medical Imaging and Graphics, 2023;

[8] *D. Evangelista, E. Morotti, J. Nagy, E. Loli Piccolomini*, **To be or not to be stable, that is the question - stability and accuracy trade-off in neural networks for inverse problems**, Submitted at SIAM SISC, 2023;

[9] *D. Bianchi, M. Donatelli, D. Evangelista, W. Li, E. Loli Piccolomini*, **Graph Laplacian and Neural Networks for Inverse Problems in Imaging: GraphLaNet**, International Conference on Scale Space and Variational Methods in Computer Vision, 2023;

[10] *D. Evangelista, E. Morotti, J. Nagy, E. Loli Piccolomini*, **Ambiguity in solving imaging inverse problems with deep learning based operators**, Journal of Imaging, 2023;

[11] *A. Asperti, D. Evangelista, S. Marro, F. Merizzi*, **Image embedding for denoising generative models**, Artificial Intelligence Review, 2023;

[12] *E. Morotti, D. Evangelista, E. Loli Piccolomini*, **Increasing noise robustness of deep learning-based image processing with model-based approaches**, Numerical Computations: Theory and Algorithms (NUMTA), 2023;

Part I

Inverse Problems: an optimization approach

Chapter 1

Inverse Problems in Imaging: from optimization to data-driven algorithms

1.1 Ill-Posed Inverse Problems

As previously mentioned, the majority of the problems in life sciences can be mathematically formulated as inverse problems. In these problems, an *unknown* term f, which resides in a Hilbert space $\mathcal{X}$, must be estimated from observed data g that lies in another Hilbert space $\mathcal{Y}$, given a sensing operator $\mathcal{K}$ that maps $\mathcal{X}$ to $\mathcal{Y}$. Throughout this thesis, we assume that $\mathcal{K}$ is a linear bounded operator, allowing us to formulate the problem as follows:

$$\text{given } g = \mathcal{K}f, \text{ recover (an approximation of) } f. \tag{1.1}$$

When addressing the problem in (1.1), it is important to determine whether a solution *exists* and, if so, whether it can be reliably recovered from the available information. To this end, we introduce the concept of an ill-posed problem. Hadamard defined a problem like (1.1) to be well-posed [3, 13] if it satisfies the following criteria:

(HC1) For all $g \in \mathcal{Y}$, a solution *exists*.

(HC2) For all $g \in \mathcal{Y}$, the solution is *unique*.

(HC3) The solution depends *continuously* on g.

It's important to note that this definition is related solely to properties of the linear operator $\mathcal{K}$. Specifically, a solution to (1.1):

1. exists for all $g \in \mathcal{Y}$ if and only if $rg(\mathcal{K}) = \mathcal{Y}$ (i.e. $\mathcal{K}$ is subjective).

2. is unique for all $g \in \mathcal{Y}$ if and only if $\mathcal{K}$ is injective.

3. depends continuously on g if and only if its inverse $\mathcal{K}^{-1}$ is continuous (or, equivalently, bounded).

In the subsequent Sections, we will remark that for some significant problems in life sciences, at least one of these conditions does not hold. It is worth noting that (HC1) is almost always satisfied since, if g has been measured, it naturally fall within the range of $\mathcal{K}$. Conversely, achieving (HC2) and (HC3) can be challenging in practice. Moreover, additional complexity is introduced as the formulation (1.1) has to be discretized in practice to be made computationally feasible. During this process, the operator $\mathcal{K}$ gets discretized as a matrix $\boldsymbol{K} \in \mathbb{R}^{m \times n}$, while f and g naturally become vectors of dimension n and m, respectively. The discretized version of the continuous formulation (1.1) thus reads:

$$\text{given } \boldsymbol{y} = \boldsymbol{K}\boldsymbol{x}^{gt}, \text{ recover (an approximation of) } \boldsymbol{x}^{gt}. \tag{1.2}$$

It is worth noting that the conditioning properties of $\boldsymbol{K}$ are strictly related to the properties (HC1), (HC2) and (HC3) of its corresponding continuous formulation $\mathcal{K}$. Indeed, it is evident that if $\mathcal{K}$ does not satisfy (HC1) or (HC2), then its discretization $\boldsymbol{K}$ will not satisfy them either. Furthermore, even when (HC2) holds for $\mathcal{K}$, it does not necessarily mean that it holds for $\boldsymbol{K}$, since the discretization process can introduce non-injectivity, especially when the number of measurements is limited and the resulting matrix becomes undersampled. In this case, the solution to (1.2) is not unique, and it becomes necessary to introduce additional information to establish uniqueness. Conversely, it is not hard to show that any non-singular matrix $\boldsymbol{K}$ always satisfies (HC3), since the inverse of a discrete operator is another discrete operator, which is trivially bounded. In [14], the authors noted that when $\mathcal{K}$ does not satisfy (HC3), as it is the case when $\mathcal{K}$ is a compact operator, then the singular values of $\boldsymbol{K}$ quickly decrease to 0, implying that unexpected noise into $\boldsymbol{y}$ gets amplified by the inverse of $\boldsymbol{K}$ in the solution of (1.2). This property is referred to as *discrete Picard condition* (DPC) in the literature and has been described for the first time in [15]. In the following, we will say that a discrete linear operator $\boldsymbol{K}$ does not satisfy (DPC) if it comes from the discretization of a continuous operator $\mathcal{K}$ that does not satisfy (HC3). Addressing problems where the $\boldsymbol{K}$ does not satisfy (DPC) or (HC2), advanced techniques are required, as it will be elaborated on later in this Chapter.

When the problem in (1.2) admits a solution, i.e. when $\boldsymbol{K}$ satisfies (HC1), various techniques have been developed in the literature, ranging from *direct methods*, where the solution is computed through an approximation of the inverse $\boldsymbol{K}^{-1}$, to *iterative methods* that employ iterative algorithms to solve a variational problem, and more recently, *neural networks*, which learn a mapping from $\mathcal{Y}$ to $\mathcal{X}$ from a dataset, achieving remarkable results.

The topic of ill-conditioned inverse problems obtained from the discretization of a continuous operator that does not satisfy the Hadamard condition of well-posedness has been

extensively studied in the last few decades. This thesis mainly focuses on applications. For a theoretical analysis, refer e.g. to [3, 14, 16, 17, 18, 19].

Structure of the Chapter This Chapter is organized as follows: in Section 2, we formally introduce the problems of interest, namely image deblurring and computed tomography. We will also outline the fundamental properties of their corresponding operators. Section 3 introduces the concept of a reconstruction algorithm, a key element in solving inverse problems. We categorize the main methods into three groups: direct methods, model-based methods, and data-driven methods. For each of them, we analyze their accuracy and stability properties, emerging when applied to the solution of ill-conditioned inverse problems. In particular, we highlight that challenges related to ill-conditioning also emerge with neural networks and we propose techniques to approach them. Moreover, we will briefly discuss the problem of sustainability concerning modern neural networks, with a focus on how it can be attenuated. Finally, Section 4 introduces some experimental tools that will appear frequently in the following Chapters, such as the metrics and the datasets we use.

1.2 Problems of interest

As mentioned in the Introduction, this thesis primarily addresses two distinct types of inverse problems stemming from life sciences: image deblurring [20] and X-rays computed tomography [21]. These two problems differ significantly in their nature, mathematical structure, and the challenges they present. Image deblurring can be modeled as an invertible linear system with a matrix $\boldsymbol{K}$ that is both surjective and injective, satisfying the first two conditions of well-posedness. However, when the blurring operator $\boldsymbol{K}$ does not satisfy (DPC), the associated inverse problem results in a system where noise in the data gets amplified in the solution to the extent that the resulting image becomes unrecognizable. In contrast, computed tomography presents a linear system where the noise in the data gets only slightly amplified. Its challenge arises from the medical requirements of sustainability and reduced patient dose, leading to a highly sub-sampled matrix $\boldsymbol{K}$ that does not satisfy (HC2). Hence, it necessitates the introduction of sophisticated mathematical techniques to ensure the uniqueness of the solution.

The conditioning properties of both the image deblurring and the computed tomography operators are consequences of them being the discretization of *integral operators* [14]. By definition, a continuous operator $\mathcal{K}$ is an integral operator if it can be written as:

$$(\mathcal{K}f)(\boldsymbol{p}) = \int_{\mathbb{R}^n} k(\boldsymbol{p}, \boldsymbol{z}) f(\boldsymbol{z}) d\boldsymbol{z}, \quad \forall \boldsymbol{p} \in \mathbb{R}^n, \tag{1.3}$$

for some kernel $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. The following Proposition shows that integral operators fail to satisfy (HC3).

Proposition 1.1. *If $\mathcal{K} : L^2(\mathbb{R}^n) \to L^2(\mathbb{R}^n)$ is an* integral operator *, then $\mathcal{K}$ is a* compact operator*. In particular (see [3]), $\mathcal{K}$ does not satisfy (HC3).*

Proof. See e.g. [3, 14]. □

Consequently, any discrete operator $\boldsymbol{K}$ that comes from the discretization of an integral operator $\mathcal{K}$, does not satisfy (DPC).

1.2.1 Image Deblurring

Image deblurring is one of the most well-known and extensively studied inverse problems in imaging. It involves the task of recovering a clear, sharp image, denoted as $\boldsymbol{x}^{gt}$, from a blurred, out-of-focus acquisition $\boldsymbol{y}$. As we will soon discover, the associated operator $\boldsymbol{K}$ is invertible but exhibits severe ill-conditioning, meaning that $\boldsymbol{K}^{-1}$ can perfectly reconstruct noiseless acquisitions, but the quality of the reconstruction rapidly deteriorates when even a small amount of noise is added to $\boldsymbol{y}$. To formally introduce this problem, let's consider a continuous image $f \in L^2(\mathbb{R}^2)$ and assume that, instead of capturing f, we measure the result, denoted as $g \in L^2(\mathbb{R}^2)$, of applying a Fredholm operator of the first kind [20], denoted as $\mathcal{K}$. This operation can be expressed as:

$$g(\boldsymbol{p}) = (\mathcal{K}f)(\boldsymbol{p}) = \int_{\mathbb{R}^2} \tilde{k}(\boldsymbol{z}; \boldsymbol{p}) f(\boldsymbol{z}) d\boldsymbol{z} \quad \forall \boldsymbol{p} \in \mathbb{R}^2. \tag{1.4}$$

Here, $\tilde{k} : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2$ is the *blurring kernel* associated with $\mathcal{K}$. Equation (1.4) can be further simplified by assuming that the kernel $\tilde{k}$ is *shift-invariant*, meaning that $\tilde{k}(\boldsymbol{z}, \boldsymbol{p}) = \tilde{k}(\boldsymbol{z} - \boldsymbol{p}, \boldsymbol{0})$ for all $\boldsymbol{z}, \boldsymbol{p} \in \mathbb{R}^2$. Under this assumption, Equation (1.4) becomes:

$$g(\boldsymbol{p}) = (\mathcal{K}f)(\boldsymbol{p}) = \int_{\mathbb{R}^2} k(\boldsymbol{z} - \boldsymbol{p}) f(\boldsymbol{z}) d\boldsymbol{z}, \quad \forall \boldsymbol{p} \in \mathbb{R}^2, \tag{1.5}$$

where we define $k(\boldsymbol{z} - \boldsymbol{p}) := \tilde{k}(\boldsymbol{z} - \boldsymbol{p}, \boldsymbol{0})$. In Equation (1.5), the operator $\mathcal{K}$ is uniquely identified by the kernel $k \in L^2(\mathbb{R}^2)$. Specifically, we have:

$$k(\boldsymbol{p}) = (\mathcal{K}\delta_{\boldsymbol{0}})(\boldsymbol{p}), \quad \forall \boldsymbol{p} \in \mathbb{R}^2, \tag{1.6}$$

where $\delta_{\boldsymbol{0}}$ represents the Dirac delta function centered at the origin. Visualized as an image, $\delta_{\boldsymbol{0}}$ represents a single point, and the kernel $k(\boldsymbol{p}) = (\mathcal{K}\delta_{\boldsymbol{0}})(\boldsymbol{p})$ characterizes how this single point is spread by the operator $\mathcal{K}$. For this reason, $k(\boldsymbol{p})$ is commonly referred to as the *Point Spread Function (PSF)*. Furthermore, note that Equation (1.5) represents a *convolution* between the PSF k and the unknown image f, i.e.

$$g(\boldsymbol{p}) = (\mathcal{K}f)(\boldsymbol{p}) = (k * f)(\boldsymbol{p}). \tag{1.7}$$

Hence, the deblurring problem is often called *deconvolution*.

In practical computational applications, we need to discretize this continuous process. From Equation (1.5), it's evident that $\mathcal{K}$ is a linear operator. Therefore, its discretization, for example with a quadrature formula, leads to a linear system in the form:

$$\boldsymbol{y} = \boldsymbol{K}\boldsymbol{x}^{gt}, \tag{1.8}$$

where $\boldsymbol{x}^{gt} \in \mathcal{X} \subseteq \mathbb{R}^n$, $\boldsymbol{y} \in \mathcal{Y} \subseteq \mathbb{R}^n$ and $\boldsymbol{K} \in \mathbb{R}^{n\times n}$. It is worth noting that since $\boldsymbol{K}$ originates from the discretization of a 2-dimensional convolutional operator, it possesses a specific structure known as BCCB (Block Circulant with Circulant Blocks), where *circulant* signifies that each column has the same elements as the previous block, shifted by one, with periodic boundary conditions [20].

In this thesis, we focus on a specific type of discrete blurring operator $\boldsymbol{K}$, namely the *Gaussian blurring operator*. Its point spread function (PSF), denoted as k, has shape $s \times s$, where $s \in \mathbb{N}$ is a parameter that in the following is assumed to be known, and is defined as follows:

$$k_{i,j} = \frac{1}{\sum_{i',j'} k_{i',j'}} \exp\left(-\frac{(i-i_c)^2 + (j-j_c)^2}{2\sigma^2}\right), \quad i_c = j_c = \left\lfloor \frac{s-1}{2} \right\rfloor. \tag{1.9}$$

From the perspective of the inverse problem in Equation (1.8), the conditioning properties of the Gaussian blur matrix $\boldsymbol{K}$ hold particular significance. It is a symmetric matrix that can be diagonalized by the 2-dimensional Fourier transform, as every BCCB matrix, with nonzero eigenvalues. Consequently, $\boldsymbol{K}$ is both injective and surjective, satisfying (HC1) and (HC2). As for the third criterion, it can be readily demonstrated that $\boldsymbol{K}$ does not satisfy (DPC), since it comes from the discretization of the integral operator $\mathcal{K}$ defined in (1.4), which does not satisfy (HC3) as indicated in Proposition 1.1.

1.2.2 Sparse Computed Tomography

In Medical Imaging, Computed Tomography (CT) [21, 22] is the task of reconstructing an image $\boldsymbol{x}^{gt} \in \mathbb{R}^n$, which represents the *attenuation coefficients* (a quantity that is strictly related to the density of the tissues) of the interior of a patient, from the acquisition of the intensity of a X-ray beam as it pass through its body. From a mathematical standpoint, let's consider the process depicted in Figure 1.1. In this setup, a source emits a beam of X-rays through the body represented as $\boldsymbol{x}^{gt}$, each with a fixed, known intensity I_0. As these X-rays traverse $\boldsymbol{x}^{gt}$, some of the photons are absorbed by the patient's tissues, resulting in the acquired ray's intensity I at a detector placed on the opposite side of the body. Specifically, we denote $I(\boldsymbol{p})$ as the intensity of a given X-ray at position $\boldsymbol{p} \in \mathbb{R}^2$.

The physics governing this process, particularly the absorption of X-rays by tissues, is described by the Beer-Lambert law:

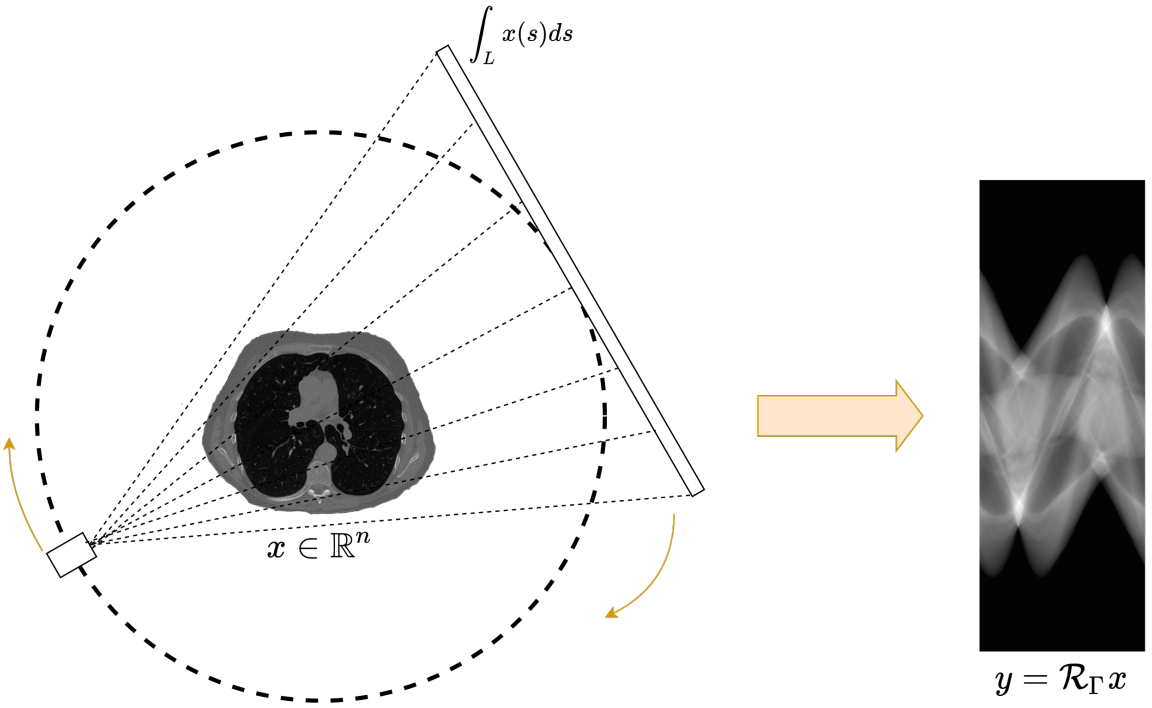


Figure 1.1: A schematic representation of the CT acquisition process.

$$dI(\boldsymbol{p}) = -\boldsymbol{x}^{gt}(\boldsymbol{p})I(\boldsymbol{p})d\boldsymbol{p}. \tag{1.10}$$

It's worth noting that Equation (1.10) can be solved as:

$$I(\boldsymbol{p}) = I_0 \exp\left(-\int_{L[\mathbf{0},\boldsymbol{p}]} \boldsymbol{x}^{gt}(\boldsymbol{z})d\boldsymbol{z}\right), \tag{1.11}$$

where $L[\mathbf{0}, \boldsymbol{p}]$ represents the trajectory of the X-ray being considered, from the origin to the point $\boldsymbol{p}$. If I_1 is the intensity of the X-ray as measured by the detector, we can derive from (1.11) that:

$$\log \frac{I_1}{I_0} = -\int_L \boldsymbol{x}^{gt}(\boldsymbol{p})d\boldsymbol{p}, \tag{1.12}$$

where L is the trajectory of the X-ray from the origin to the detector. Importantly, since I_1 is measured for every ray, and I_0 is known by design, the recovery problem is formulated as reconstructing $\boldsymbol{x}^{gt}(\boldsymbol{p})$ from measurements $g(s) = \log \frac{I_1}{I_0}$, where $s \in \mathbb{R}$ represents the measured ray's position. Clearly, it is impossible to reconstruct the complete density map $\boldsymbol{x}^{gt}(\boldsymbol{p})$ from a single projection $g(s)$. To tackle this problem effectively, multiple rays are collected by rotating the source around the object at various angles, acquiring multiple projections. For any angular parameter $\Gamma > 0$, we consider the acquisition $g(s, \Theta)$, which represents the X-ray

intensity at position s when the source is at angle Θ around the object. Mathematically, this problem is closely related to the inversion of the Radon transform $\mathcal{R}_\Gamma$ [23], defined as:

$$(\mathcal{R}_\Gamma \boldsymbol{x}^{gt})(s,\Theta) = \int_{L(s,\Theta)} \boldsymbol{x}^{gt}(\boldsymbol{p})d\boldsymbol{p}, \tag{1.13}$$

where $L(s,\Theta)$ is the trajectory of the ray in position s at angle Θ. With this notation, the continuous version of the CT reconstruction problem becomes:

$$\text{from } g(s,\Theta) = (\mathcal{R}_\Gamma \boldsymbol{x}^{gt})(s,\Theta), \text{ recover } \boldsymbol{x}^{gt}. \tag{1.14}$$

From an analytical perspective, this problem is relatively easy to solve when Γ is sufficiently large, as the operator $\mathcal{R}_\Gamma$ is invertible. However, issues come out when $\mathcal{R}_\Gamma$ gets discretized. To do that, only a finite number n_Θ of angles are sampled within the range $[0,\Gamma]$, denoted as $\Theta_1,\dots,\Theta_{n_\Theta}$, and a finite number n_d of rays are considered for each angular scan. Consequently, (1.14) turns into:

$$\boldsymbol{y} = \boldsymbol{R}_\Gamma \boldsymbol{x}^{gt}, \quad \boldsymbol{R}_\Gamma \in \mathbb{R}^{m\times n}, \quad m = n_d \cdot n_\Theta. \tag{1.15}$$

Note that the shape of the matrix $\boldsymbol{R}_\Gamma$ depends on the number of discretized X-rays and angles. If their count is not sufficiently large, the matrix becomes under-sampled, resulting in a non-unique solution for (1.15). As modern medical protocols aim to minimize the number of projection angles, due to their relationship with patient radiation exposure, this effectively renders $\boldsymbol{R}_\Gamma$ non-injective in practice. Moreover, $\boldsymbol{R}_\Gamma$ comes from the discretization of an integral operator $\mathbb{R}_\Gamma$ and consequently, from Proposition 1.1, it does not satisfy (DPC) [18]. Problems of this nature are referred to as *Sparse Computed Tomography (SparseCT)* in the literature. In this thesis, our primary focus lies in addressing SparseCT problems, where the non-injectivity of $\boldsymbol{R}_\Gamma$ introduces stability issues that, while visually similar, are fundamentally distinct from those encountered in deblurring problems. These issues necessitate the introduction of stabilization techniques to mitigate the challenges posed by non-injectivity.

Furthermore, the geometry of the X-ray beam plays a central role in defining the properties of the associated operator. It is a common practice in the literature to deal with what is known as *parallel-beam CT*, where the emitting source is assumed to have the same shape as the detector, and the ray-beam consists of parallel, non-intersecting rays. In this scenario, the associated Radon operator $\boldsymbol{R}_\Gamma$ is *normal-convolutional*, indicating that its associated normal operator $\boldsymbol{R}_\Gamma^*\boldsymbol{R}_\Gamma$ corresponds to a convolution matrix, as it is proved in [24]. Although this property simplifies computations involving $\boldsymbol{R}_\Gamma$, it relies on assumptions that are unrealistic in modern tomographic systems. In real-world scenarios, the emitting source is often approximately a point, emitting X-rays radially from its center. This setup is referred to as *Cone-Beam Computed Tomography (CBCT)* [25, 26, 27, 28] in the 3-dimensional case and *Fan-Beam Computed Tomography* [29, 30] in the 2-dimensional case. Figure 1.2

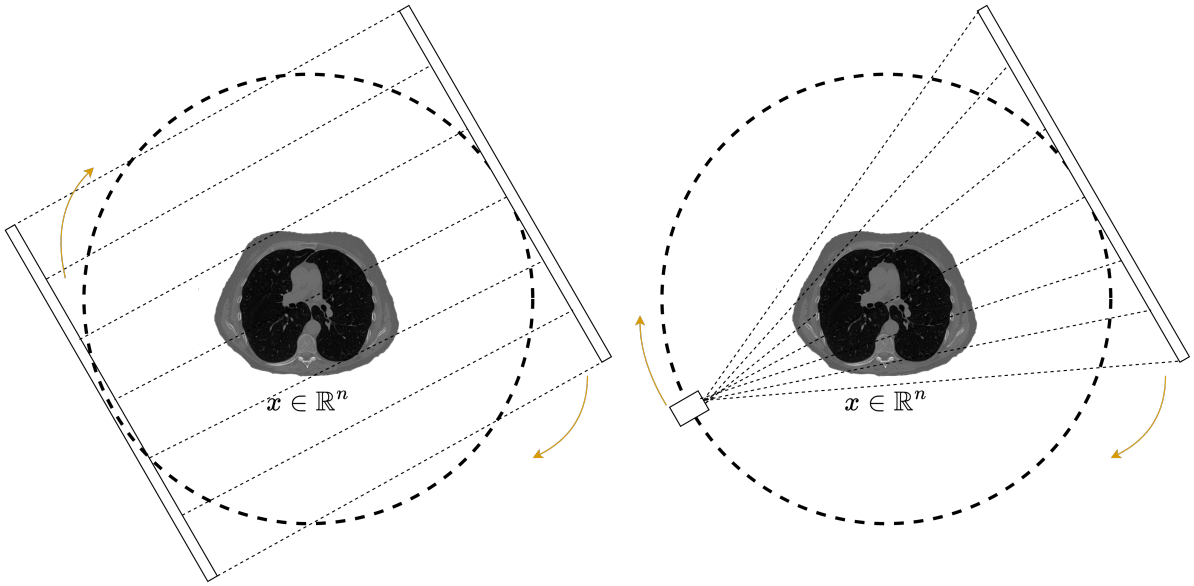


Figure 1.2: An illustration of the parallel vs fan beam CT geometry. Left. Parallel Beam. Right. Fan Beam.

visually represents the differences in the two modalities. Unlike parallel-beam CT, CBCT and Fan-Beam CT do not satisfy the normal-convolutional properties, necessitating more sophisticated approaches in developing the reconstruction algorithms [31]. To better align with the requirements of realistic medical protocols, this thesis consistently assumes the Fan-Beam Computed Tomography setup in the 2-dimensional experiments.

The non-injectivity of the Radon operator in the SparseCT setup and the absence of the normal-convolutional properties of Fan-Beam geometry introduce challenges that, although currently unsolved, we firmly believe to be of paramount importance to, at least, try to solve. Advances on this field have the potential to streamline the diagnosis of certain diseases while minimizing patients' exposure to potentially harmful levels of radiation.

1.3 Reconstruction algorithms

After the introduction of the two operators of interest and their conditioning properties, we now discuss the techniques used to address the associated inverse problem. It's important to note that in real applications, the measured data $\boldsymbol{y}$ is invariably corrupted by unpredictable noise, often stemming from the electronics of the acquisition device. As a result, we now consider an inverse problem in the form of:

$$\boldsymbol{y}^{\delta} = \boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}. \tag{1.16}$$

Here, the noise $\boldsymbol{e}$ has a norm bounded by δ (i.e., $||\boldsymbol{e}||_2 \leq \delta$), and for many applications, it

is assumed to follow a zero-mean Gaussian distribution. *Solving* an inverse problem entails finding a way to map the corrupted and noisy data $\boldsymbol{y}^\delta$ to an approximation of $\boldsymbol{x}^{gt}$. We refer to these methods as *reconstruction algorithms* in the following. In the literature, numerous reconstruction algorithms have been developed over the years, exploiting the specific properties of the tasks to which they are applied. These methods aim to strike a balance between accurate and stable reconstructions. As we will explore in the following Sections, the inherent ill-conditioning of the forward operator $\boldsymbol{K}$ can make any reconstruction algorithm more unstable as it becomes more accurate. This creates a trade-off that must be optimized, as we will discuss. Within the broad category of reconstruction algorithms, we can identify three primary classes of methods: the *direct methods*, the *iterative (model-based) methods*, and the *data-driven methods*. The remainder of this Section will delve into these three classes of reconstruction algorithms, elucidating their interrelationships and the associated challenges.

1.3.1 Direct Methods

The category of direct methods encompasses all the algorithms designed to address (1.16) by (approximately) *inverting* the operator $\boldsymbol{K}$. Each direct technique relies, to some extent, on the concept of the Moore-Penrose pseudoinverse of $\boldsymbol{K}$ [3]. To define it, let's consider a noiseless problem in the form of $\boldsymbol{y} = \boldsymbol{K}\boldsymbol{x}^{gt}$. By definition $\boldsymbol{y} \in Rg(\boldsymbol{K})$, indicating that the system has at least one solution, and possibly more. The Moore-Penrose pseudoinverse of $\boldsymbol{K}$, denoted as $\boldsymbol{K}^\dagger$, is the linear operator that maps $\boldsymbol{y}$ to the solution of $\boldsymbol{y} = \boldsymbol{K}\boldsymbol{x}^{gt}$ with the minimum norm. It's worth emphasizing the main property of $\boldsymbol{K}^\dagger$ that can be useful for our problems of interest:

Proposition 1.2. *If $\boldsymbol{K} \in \mathbb{R}^{m\times n}$ is full-column rank (i.e. $m \geq n$ and $rk(\boldsymbol{K}) = n$), then $\boldsymbol{K}^\dagger$ has the simple expression:*

$$\boldsymbol{K}^\dagger = (\boldsymbol{K}^*\boldsymbol{K})^{-1}\boldsymbol{K}^*. \tag{1.17}$$

In particular, if $m = n$ and K is invertible, then, $\boldsymbol{K}^\dagger = \boldsymbol{K}^{-1}$. For this reason, $\boldsymbol{K}^\dagger$ is also referred to as the generalized inverse.

Proof. Refer e.g. to [3]. □

Note that, as we observed in the introduction to Deblurring in the previous Section, when $\boldsymbol{y}$ is corrupted by noise and $\boldsymbol{K}$ does not satisfy (DPC), direct inversion of $\boldsymbol{K}$ will amplify the impact of the noise, making the image unrecognizable in practice. This occurs because when we measure $\boldsymbol{y}^\delta = \boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}$, the solution $\boldsymbol{K}^\dagger\boldsymbol{y}^\delta$ finds a value of $\boldsymbol{x}$ that satisfies the system $\boldsymbol{K}\boldsymbol{x} = \boldsymbol{y}^\delta$, which differs from the desired solution, where we require $\boldsymbol{K}\boldsymbol{x}$ to be equal to $\boldsymbol{y}$. Consequently, direct approaches cannot be employed in scenarios where the forward operator $\boldsymbol{K}$ does not satisfy (DPC), as is the case with deblurring inverse problems.

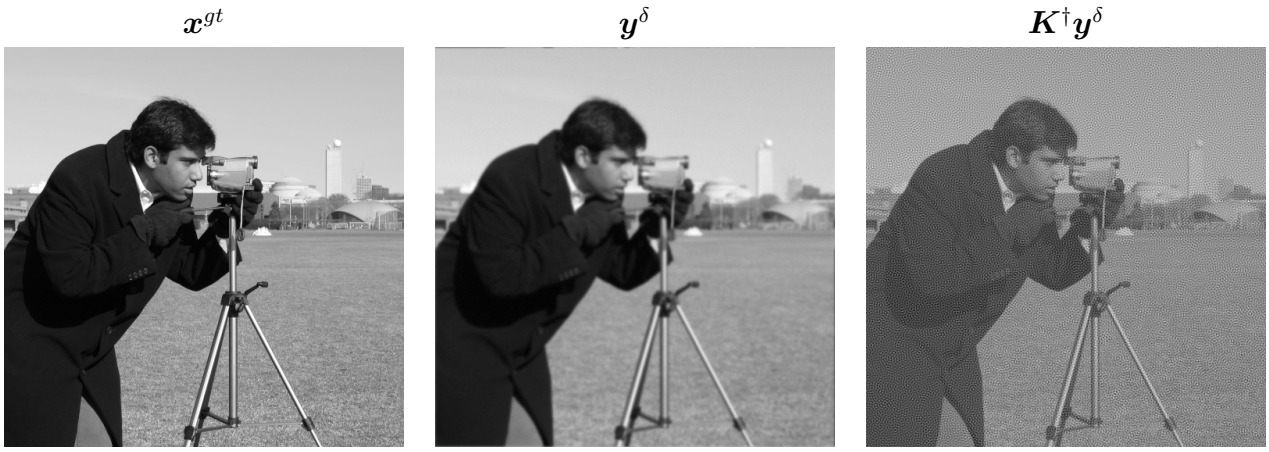


Figure 1.3: The result obtained by applying the Moore-Penrose pseudoinverse operator to an inverse problem where the operator $\boldsymbol{K}$ does not satisfy (DPC) and the datum is slightly corrupted by noise. *Left.* the ground-truth image $\boldsymbol{x}^{gt}$. *Center.* the blurred and slightly noisy version of $\boldsymbol{x}^{gt}$. *Right.* the reconstruction obtained by a direct method.

The results obtained by applying a direct method to an inverse problem where the operator $\boldsymbol{K}$ does not satisfy (DPC) can be seen in Figure 1.3. In this experiment, we consider an image $\boldsymbol{x}^{gt}$ with dimensions 512×512. This image is subjected to blurring by a Gaussian blur operator with parameters $s = 11$ and $\sigma^2 = 1$, whose definition can be found in Section 1.2.1. Subsequently, the blurred image is corrupted by additive Gaussian noise, with a norm equal to 1% of the norm of the image. Finally, the pseudoinverse operator $\boldsymbol{K}^{\dagger}$ is applied to $\boldsymbol{y}^{\delta} = \boldsymbol{K}\boldsymbol{x} + \boldsymbol{e}$ in an attempt to reconstruct the image. Regrettably, the result is heavily corrupted, highlighting that the Moore-Penrose pseudoinverse operator $\boldsymbol{K}^{\dagger}$ amplifies the noise present in the image.

Conversely, direct methods are commonly applied in slightly ill-posed inverse problems such as computed tomography, where the use of the pseudoinverse (known as the Filtered Back Projection (FBP) algorithm) is widespread in real medical applications. To understand how the FBP algorithm works, consider the case of a parallel-beam CT inverse problem (similar discussions apply to the fan-beam setup). As previously mentioned, in this case the operator $\boldsymbol{K}$ is a normal-convolutional operator, meaning that $\boldsymbol{K}^*\boldsymbol{K}$ is a convolution matrix. If the number of projections is sufficiently large, $\boldsymbol{K}^*\boldsymbol{K}$ can be efficiently inverted in the Fourier domain, functioning as a *filter* on the image it's applied to. Consequently, the pseudoinverse $\boldsymbol{K}^{\dagger}$, computed as $(\boldsymbol{K}^*\boldsymbol{K})^{-1}\boldsymbol{K}^*$ (from Proposition 1.2, since $\boldsymbol{K}$ is full-column rank when the number of projections is adequate), essentially acts as a filter applied to the back-projection (defined as the result of applying $\boldsymbol{K}^*$ to the sinogram). This is why the pseudoinverse is usually referred to as the *Filtered Back Projection (FBP)* operator, a term commonly used in the literature. When the number of projections n_Θ is low, as is the case with SparseCT, the filter $\boldsymbol{K}^*\boldsymbol{K}$ is not invertible. In such situations, the FBP operator

must be approximated using pre-built filters, like the commonly used ramp filter. The resulting operator is an approximation of the pseudoinverse $\boldsymbol{K}^{\dagger}$, which exhibits poor quality and prominent global artifacts in the reconstructed solution, degrading the image as evident in Figure 1.4, where the FBP operator is applied to a SparseCT problem with $n_{\Theta} = 60$ projection angles are considered, uniformly sampled in the $[0, \pi]$ range.

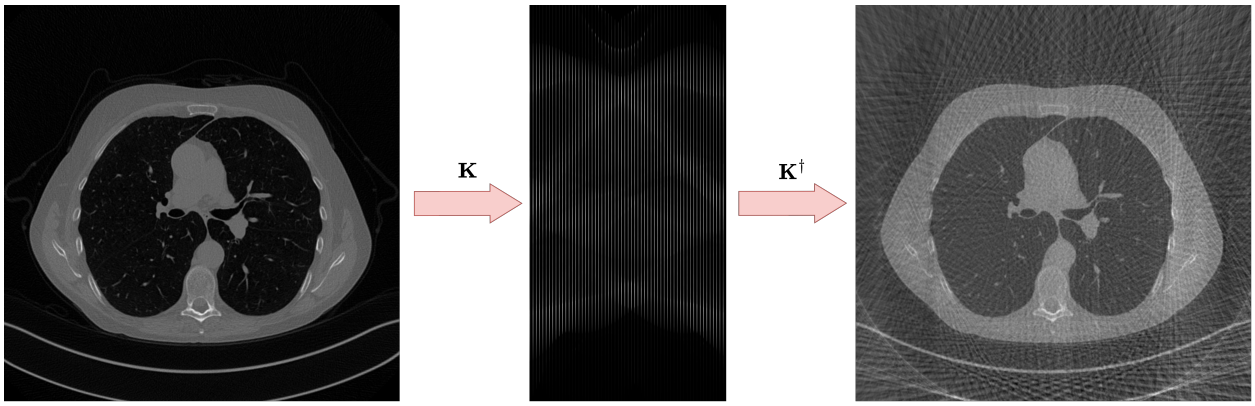


Figure 1.4: The result of the FBP operator on a SparseCT problem.

1.3.2 Model-Based Methods

To address the challenges posed by the severe ill-posedness of the blurring operator and the non-invertibility of the normal operator in the SparseCT inverse problem, numerous approaches have been proposed in the literature. One of the most successful is to reformulate the inverse problem as a regularized optimization problem [3, 19, 22], solved through the application of iterative algorithms.

Definition of the optimization problem Consider an optimization problem of the form:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^{\delta}) + \lambda \mathcal{R}(\boldsymbol{x}), \tag{1.18}$$

where $\mathcal{J}$ is a measure of the distance between $\boldsymbol{K}\boldsymbol{x}$ and $\boldsymbol{y}^{\delta}$ and it is usually referred to as *fidelity term*, $\lambda > 0$ is a scalar that balances the two terms in the optimization problem, known as *regularization parameter*, while $\mathcal{R}$ formalizes the prior knowledge we have on the true $\boldsymbol{x}$ by penalizing the solutions that do not follows that prior. Here, the choice of $\mathcal{J}$ is usually determined by the statistics of the noise as input and, in particular, it is defined as the logarithm of the likelihood distribution of the noisy data $\boldsymbol{y}^{\delta}$ given $\boldsymbol{K}\boldsymbol{x}^{gt}$ [19]. For example, when $\boldsymbol{e}$ is assumed to be Gaussian distributed, then $\mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^{\delta}) = \frac{1}{2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^{\delta}||_2^2$, while if $\boldsymbol{e}$ is Poisson noise, then $\mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^{\delta}) = D_{KL}(\boldsymbol{K}\boldsymbol{x}||\boldsymbol{y}^{\delta})$, where D_{KL} is the Kullback-Leibler Divergence [18, 32]. On the other side, the choice of the regularizer $\mathcal{R}$ requires more

attention and it is usually based on the conditioning properties of $\boldsymbol{K}$ and on the assumptions on the structure of the set $\mathcal{X}$ of *good* images. Consequently, different considerations have to be done with respect to the image deblurring and the SparseCT inverse problems.

When the forward operator $\boldsymbol{K}$ is a Gaussian blur operator, we have already shown that it does not satisfy (DPC). It is thus natural to introduce a regularization term that in a way substitutes $\boldsymbol{K}$ with a better-conditioned matrix, such that the solution of the regularized problem does not amplify the noise as input. To do that, $\mathcal{R}(\boldsymbol{x})$ can be simply chosen to be equal to $||\boldsymbol{L}\boldsymbol{x}||_2^2$, where $\boldsymbol{L} \in \mathbb{R}^{d\times n}$ is any matrix such that $\ker(\boldsymbol{K}) \cap \ker(\boldsymbol{L}) = \{\boldsymbol{0}\}$. The resulting optimization problem becomes

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{1}{2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta||_2^2 + \frac{\lambda}{2}||\boldsymbol{L}\boldsymbol{x}||_2^2, \tag{1.19}$$

which is known as *Tikhonov-regularized inverse problem* [33], due to the name classicaly attributed to this choice of $\mathcal{R}(\boldsymbol{x})$, which is *Tikhonov-regularization term.* Note that (1.19) can be rewritten as:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{1}{2}||\tilde{\boldsymbol{K}}^{\boldsymbol{L}}\boldsymbol{x} - \tilde{\boldsymbol{y}}^\delta||_2^2, \tag{1.20}$$

where we defined

$$\tilde{\boldsymbol{K}}^{\boldsymbol{L}} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \\ \boldsymbol{0} & \lambda\boldsymbol{L} \end{bmatrix} \text{ and } \tilde{\boldsymbol{y}}^\delta = \begin{bmatrix} \boldsymbol{y}^\delta \\ \boldsymbol{0} \end{bmatrix}. \tag{1.21}$$

The equation above defines a strictly convex optimization problem, whose solution exists and is unique if and only if $\ker(\boldsymbol{K})\cap\ker(\boldsymbol{L}) = \{\boldsymbol{0}\}$ and can be computed by any optimization algorithm such as the Conjugate Gradient Method for Least-Squares (CGLS) [34].

It can be shown that the matrix $\tilde{\boldsymbol{K}}^{\boldsymbol{L}}$ is better conditioned when compared to $\boldsymbol{K}$. In particular, it holds:

Proposition 1.3. *If $L = I$, then the i-th singular value of $\tilde{\boldsymbol{K}}^{\boldsymbol{L}}$ is equal to $\sqrt{\sigma_i^2 + \lambda^2}$, where σ_i is the i-th singular value of $\boldsymbol{K}$.*

Proof. Let $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be the SVD of $\boldsymbol{K}$. Then:

$$\left(\tilde{\boldsymbol{K}}^{\boldsymbol{L}}\right)^T \tilde{\boldsymbol{K}}^{\boldsymbol{L}} = \boldsymbol{K}^T\boldsymbol{K} + \lambda^2\boldsymbol{I} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T + \lambda^2\boldsymbol{I} = \boldsymbol{V}\left(\boldsymbol{\Sigma} + \lambda^2\boldsymbol{I}\right)\boldsymbol{V}^T \tag{1.22}$$

is the eigenvalue decomposition of $\left(\tilde{\boldsymbol{K}}^{\boldsymbol{L}}\right)^T \tilde{\boldsymbol{K}}^{\boldsymbol{L}}$. Consequently, the i-th eigenvalue of $\left(\tilde{\boldsymbol{K}}^{\boldsymbol{L}}\right)^T \tilde{\boldsymbol{K}}^{\boldsymbol{L}}$ is equal to $\sigma_i^2+\lambda^2$. Since the i-th singular value of $\tilde{\boldsymbol{K}}^{\boldsymbol{L}}$ is the square root of the i-th eigenvalue of its associated normal operator, we conclude. □

As a consequence of Proposition 1.3, $\tilde{\boldsymbol{K}}^{\boldsymbol{L}}$ is well-conditioned if λ is large enough. Clearly, the solution of the Tikhonov-regularized inverse problem will be less accurate than the

pseudo-inverse solution $\boldsymbol{K}^{\dagger}\boldsymbol{y}^{\delta}$ when no noise is considered in $\boldsymbol{y}^{\delta}$ since approximating $\boldsymbol{K}$ with $\boldsymbol{K}^{\boldsymbol{L}}$ introduces errors in the operator that reduces the quality of the reconstruction. Despite that, introducing a regularizer stabilizes the operator, allowing it to obtain good-quality results when noise is present in the data. For example, in Figure 1.5, we compared the quality of the reconstruction of the same experiments as in Figure 1.3, with Tikhonov regularization. It is clear that, especially when low noise levels are considered, the Tikhonov reconstruction has greater quality compared to the direct solution, as shown in the previous Figure.

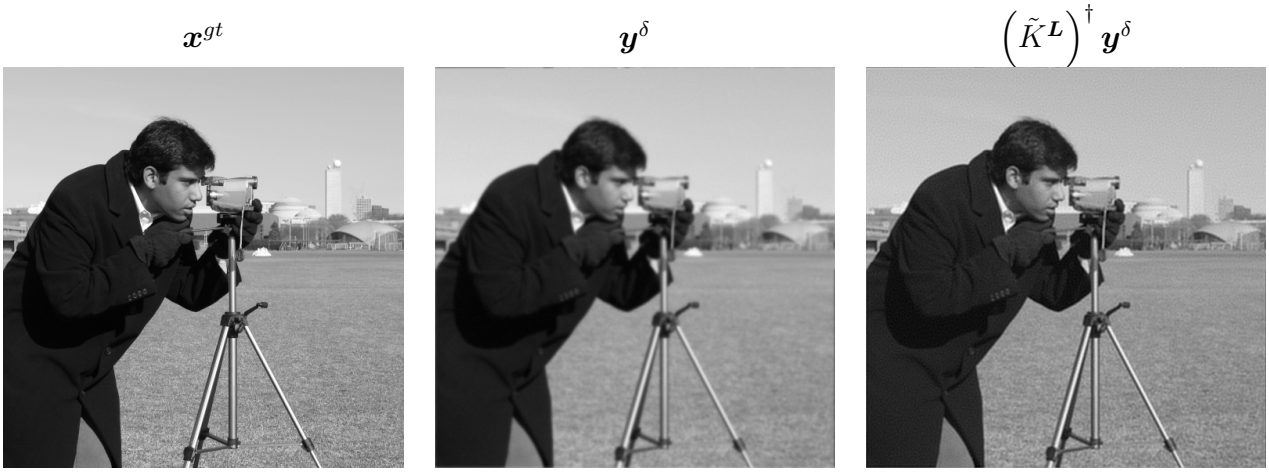


Figure 1.5: The result obtained by applying the Tikhonov regularization method to an inverse problem where the operator $\boldsymbol{K}$ does not satisfy (DPC) and the datum is slightly corrupted by noise. *Left.* the ground-truth image $\boldsymbol{x}^{gt}$. *Center.* the blurred and slightly noisy version of $\boldsymbol{x}^{gt}$. *Right.* the reconstruction obtained by solving the Tikhonov regularized inverse problem.

On the other side, when the system has infinite solutions such as in SparseCT, the formulation of $\mathcal{R}$ becomes more complicated. In particular, to restrict the space of possible solutions $\mathcal{X}$ to a singleton, it is common to assume that $\mathcal{X}$ has a *sparse* structure, meaning that there exists a *sparsifying transformation* $\boldsymbol{T} : \mathbb{R}^n \to \mathbb{R}^n$, such that for any $\boldsymbol{x} \in \mathcal{X}$, $\sum_{i=1}^{n} \mathbb{1}_{(\boldsymbol{T}\boldsymbol{x})_i \neq 0} =: ||\boldsymbol{T}\boldsymbol{x}||_0 \leq s$, where $s \ll n$, while $||\boldsymbol{T}\boldsymbol{x}||_0 \approx n$ for any $\boldsymbol{x} \notin \mathcal{X}$. When this is the case, we say that $\mathcal{X}$ is s-sparse [35, 36]. Note that, under this assumption, a natural formulation for the solution of $\boldsymbol{y}^{\delta} = \boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}$ would be:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{T}\boldsymbol{x}||_0 \text{ s.t. } \boldsymbol{K}\boldsymbol{x} = \boldsymbol{y}^{\delta}, \tag{1.23}$$

or, by relaxing the constraint:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{T}\boldsymbol{x}||_0 \text{ s.t. } \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^{\delta}) \leq \epsilon, \tag{1.24}$$

for some small $\epsilon > 0$. Since the ℓ_0-pseudonorm $||\boldsymbol{T}\boldsymbol{x}||_0$ is non-convex and non-smooth, it is common to substitute it in (1.24) by its convex relaxation $||\boldsymbol{T}\boldsymbol{x}||_1 = \sum_{i=1}^{n} |(\boldsymbol{T}\boldsymbol{x})_i|$, to obtain,

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{T}\boldsymbol{x}||_1 \text{ s.t. } \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta) \leq \epsilon. \tag{1.25}$$

The optimization problem (1.25) can be easily solved by considering the associated unconstrained Lagrangian formulation:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta) + \lambda||\boldsymbol{T}\boldsymbol{x}||_1, \tag{1.26}$$

leading to an optimization problem such as (1.18), with $\mathcal{R}(\boldsymbol{x}) = ||\boldsymbol{T}\boldsymbol{x}||_1$. When $\boldsymbol{T}$ is a coordinate change matrix, as is the case when it represents the Wavelet transform [37] or the Fourier Transform [38], the problem (1.26) can be solved with algorithms like the Basis-Pursuit [36] or FISTA [39]. In this case, the Compressive Sensing (CS) [35] literature proved important recovery results. For example, if the matrix $\boldsymbol{K}$ satisfies a property called *s-Restricted Isometry Property (s-RIP)*, then under suitable hypothesis on the number of measurements m, the global minimum $\boldsymbol{x}^*$ for the strictly convex optimization problem (1.26) is a minimum for (1.23) and satisfies $||\boldsymbol{x}^* - \boldsymbol{x}^{gt}||_2 \leq C\delta$ for a constant $C > 0$ (see [36] for more details).

In modern applications, it is common in the SparseCT literature to consider a particular sparsifying transform $\boldsymbol{T}$ that is not a coordinate change but works better in practice: the gradient operator. In particular, if $\boldsymbol{x} \in \mathbb{R}^n$ represents a 2-dimensional image, then we define $\boldsymbol{T}\boldsymbol{x} = \boldsymbol{D}\boldsymbol{x}$, where

$$\boldsymbol{D}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{D}_h\boldsymbol{x} \\ \boldsymbol{D}_v\boldsymbol{x} \end{bmatrix} \in \mathbb{R}^{2n\times n}. \tag{1.27}$$

Here, $\boldsymbol{D}_h$ and $\boldsymbol{D}_v$ are $n \times n$ matrices representing the discretization of the horizontal and vertical derivative of $\boldsymbol{x}$, respectively. The resulting optimization problem (1.26) reads:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta) + \lambda||\boldsymbol{D}\boldsymbol{x}||_{2,1}, \tag{1.28}$$

where we defined:

$$||\boldsymbol{D}\boldsymbol{x}||_{2,1} = \sum_{i=1}^{n} \sqrt{(\boldsymbol{D}_h\boldsymbol{x})_i^2 + (\boldsymbol{D}_v\boldsymbol{x})_i^2}. \tag{1.29}$$

To simplify the notation, $||\boldsymbol{D}\boldsymbol{x}||_{2,1}$ it is commonly referred to as *Total Variation* and indicated as $\text{TV}(\boldsymbol{x}) := ||\boldsymbol{D}\boldsymbol{x}||_{2,1}$. With this notation, Equation (1.28) is:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta) + \lambda\text{TV}(\boldsymbol{x}), \tag{1.30}$$

which is named *Total Variation-regularized inverse problem* [40, 41]. In practical applications, it is common to also enforce non-negative constraint for $\boldsymbol{x}$, as digital images are only defined for positive pixel values, leading to:

$$\min_{\boldsymbol{x}\geq\boldsymbol{0}} \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^{\delta}) + \lambda \mathrm{TV}(\boldsymbol{x}). \tag{1.31}$$

Since the above optimization problem is the sum of a convex, smooth functional $\mathcal{J}$ and a convex, non-smooth term TV, it can be efficiently solved by the Chambolle-Pock (CP) algorithm [42] for Total Variation minimization (named *CP-TV* in the following), or by the *Scaled Gradient Projection (SGP)* [43] algorithm. A quick note on the definition (1.29): some authors define $\mathrm{TV}(\boldsymbol{x}) := ||\boldsymbol{D}_h\boldsymbol{x}||_1 + ||\boldsymbol{D}_v\boldsymbol{x}||_1$. This is known as anisotropic Total Variation functional, in contrast with the one defined in (1.29) which is the *isotropic* Total Variation functional. Through this thesis, we will consistently consider the *isotropic* formulation in (1.29).

In modern SparseCT algorithms, since $\boldsymbol{D}$ is not invertible and the discretization of the fan-beam Radon transform $\boldsymbol{R}_{\Gamma}$ does not satisfy the s-RIP conditions, theoretical robust recovery properties from the CS literature do not hold. Authors in [44] indicated that error bounds similar to the one from CS literature hold in this case, while a complete understanding is still lacking. Despite that, the empirical analysis shows that minimizers of (1.30) exhibit remarkable reconstruction quality even with very few discretization angles n_{Θ} and high noise levels.

Aware that Tikhonov and Total Variation regularizers are not necessarily the most effective regularizers for both image deblurring and SparseCT, in this thesis we mainly focus on them as a case study for our proposed methods. Extending the methods introduced in my PhD to more advanced regularizers is left as a future work.

Optimization algorithms From an optimization point of view, in this thesis we solve Tikhonov-regularized inverse problems by the CGLS algorithm, while TV regularized inverse problems are solved by either the Chambolle-Pock or the Scaled Gradient Projection algorithms. The remainder of this Section will delve into the specific form of each considered algorithm.

Conjugate Gradient for Least Squares (CGLS) The CGLS algorithm is a very fast iterative method for the solution of quadratic optimization problems, i.e. where the objective function $\mathcal{F}(\boldsymbol{x})$ can be written as:

$$\mathcal{F}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^T\boldsymbol{x} + c. \tag{1.32}$$

In Equation (1.20), we already observed that the Tikhonov-regularized inverse problem has this form, with $\boldsymbol{K} := \left(\tilde{\boldsymbol{K}}^{\boldsymbol{L}}\right)^* \tilde{\boldsymbol{K}}^{\boldsymbol{L}}$, $\boldsymbol{y} = \left(\tilde{\boldsymbol{K}}^{\boldsymbol{L}}\right)^* \tilde{\boldsymbol{y}}^{\delta}$, $c = 0$. Note that, since $\mathcal{F}(\boldsymbol{x})$ is a strictly convex function, its only minimum can be computed as $\nabla\mathcal{F}(\boldsymbol{x}) = \boldsymbol{0}$. Straightforward computation shows that the solution to this equation is the solution of the linear system $\boldsymbol{K}^*\boldsymbol{K}\boldsymbol{x} = \boldsymbol{K}^*\boldsymbol{y}$, which is the system of normal equations associated with $\mathcal{F}(\boldsymbol{x})$.

Given a starting guess $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$, the CGLS method defines a sequence $\boldsymbol{p}^{(k)}$ of $\boldsymbol{K}$-conjugate vectors (meaning that $\left(\boldsymbol{p}^{(s)}\right)^T \boldsymbol{K}\boldsymbol{p}^{(t)} = 0$ for any $s, t \in \mathbb{N}$), and updating

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha^{(k)}\boldsymbol{p}^{(k)}, \tag{1.33}$$

where $\alpha^{(k)} > 0$ is the step size. In particular, by renaming $\boldsymbol{g}^{(k)} := -\nabla\mathcal{F}(\boldsymbol{x}^{(k)})$, the $\boldsymbol{K}$-conjugate vector $\boldsymbol{p}^{(k)}$ is computed as:

$$\begin{aligned} \beta^{(k+1)} &= \frac{\left(\boldsymbol{g}^{(k)}\right)^T \boldsymbol{K}\boldsymbol{p}^{(k)}}{\left(\boldsymbol{p}^{(k)}\right)^T \boldsymbol{K}\boldsymbol{p}^{(k)}}, \\ \boldsymbol{p}^{(k+1)} &= \boldsymbol{g}^{(k)} - \beta^{(k+1)}\boldsymbol{p}^{(k)}, \end{aligned} \tag{1.34}$$

while the step size $\alpha^{(k)}$ is chosen to maximize the decrease of $\mathcal{F}$ in the direction $\boldsymbol{p}^{(k)}$ starting from $\boldsymbol{x}^{(k)}$, that is:

$$\alpha^{(k+1)} = \frac{||\boldsymbol{g}^{(k)}||_2^2}{\left(\boldsymbol{p}^{(k)}\right)^T \boldsymbol{K}\boldsymbol{p}^{(k)}}. \tag{1.35}$$

Algorithm 1 shows an efficient implementation of the described method. The importance of this algorithm resides not only in its convergence speed, which is linear in the worst-case scenario, but converges to the exact solution in at most n iterations, but also in the fact that it does not require the matrix $\boldsymbol{K}$ to be stored in memory, since it requires only matrix-vector and matrix transposed-vector operations. Consequently, it can be used in problems where $\boldsymbol{K}$ is too large to be stored in memory but the application of $\boldsymbol{K}$ and its transposed can be done efficiently, as it is the case in the image deblurring problem. In that case, the algorithm returns an accurate solution to the ill-conditioned problem with low time requirements.

Scaled Gradient Projection (SGP) The Scaled Gradient Projection (SGP) algorithm [25, 43] is a fast algorithm for the optimization of smooth objective functions $\mathcal{F}(\boldsymbol{x})$, where it is possible to explicitly compute $\nabla\mathcal{F}(\boldsymbol{x})$ at any point $\boldsymbol{x} \in \mathbb{R}^n$. Since we are interested in utilizing this algorithm to solve (1.31), we note that the objective function is not smooth, due to the TV($\boldsymbol{x}$) term being non-smooth at $\boldsymbol{x} = \boldsymbol{0}$. To apply SGP to (1.31), it is thus

Algorithm 1 Conjugate Gradient method for Least Squares (CGLS)

task minimize $\mathcal{F}(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}||_2^2$ for $\boldsymbol{x} \in \mathbb{R}^n$
input a starting guess $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$
initialize $\boldsymbol{r}^{(0)} = \boldsymbol{y} - \boldsymbol{K}\boldsymbol{x}^{(0)}$, $\boldsymbol{g}^{(0)} = \boldsymbol{K}^T\boldsymbol{r}^{(0)}$
for $k \leftarrow 1 : n$ **do**
  if $k = 1$ **then**
    $\boldsymbol{p}^{(1)} = \boldsymbol{g}^{(0)}$
  else
    $\beta^{(k+1)} = -\frac{||\boldsymbol{g}^{(k)}||_2^2}{||\boldsymbol{g}^{(k-1)}||_2^2}$
    $\boldsymbol{p}^{(k+1)} = \boldsymbol{g}^{(k)} - \beta^{(k+1)}\boldsymbol{p}^{(k)}$
  end if
  $\alpha^{(k+1)} = \frac{||\boldsymbol{g}^{(k)}||_2^2}{||\boldsymbol{K}\boldsymbol{p}^{(k+1)}||_2^2}$
  $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha^{(k+1)}\boldsymbol{p}^{(k+1)}$
  $\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k+1)} - \alpha^{(k+1)}\boldsymbol{K}\boldsymbol{p}^{(k+1)}$
  $\boldsymbol{g}^{(k+1)} = \boldsymbol{K}^T\boldsymbol{r}^{k+1}$
end for
return $\boldsymbol{x}^{(n)}$

necessary to substitute the TV regularization term by a smooth approximation of it. A common choice is to consider the TV_β functional, defined as

$$\mathrm{TV}_\beta(\boldsymbol{x}) = \sum_{i=1}^{n} \sqrt{(\boldsymbol{D}_h\boldsymbol{x})_i^2 + (\boldsymbol{D}_v\boldsymbol{x})_i^2 + \beta^2}, \tag{1.36}$$

for a small $\beta > 0$. The resulting optimization problem

$$\min_{\boldsymbol{x} \geq \boldsymbol{0}} \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta) + \lambda \mathrm{TV}_\beta(\boldsymbol{x}), \tag{1.37}$$

has a smooth objective function and can be solved by the SGP algorithm. It is an iterative, gradient-based method, where for each iteration $k \geq 0$, the iterate $\boldsymbol{x}^{(k)}$ gets updated as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}, \tag{1.38}$$

with $\alpha_k > 0$ step-size and $\boldsymbol{d}^{(k)}$ a descent direction. In particular, the descent direction of SGP is chosen as:

$$\boldsymbol{d}^{(k)} = \mathcal{P}_+\left(\boldsymbol{x}^{(k)} - \eta_k \boldsymbol{S}_k \nabla\mathcal{F}(\boldsymbol{x}^{(k)})\right) - \boldsymbol{x}^{(k)}, \tag{1.39}$$

where $\mathcal{P}_+ : \mathbb{R}^n \to \mathbb{R}^n$ is the projection on the non-negative axis, η_k is a step-size, and $\boldsymbol{S}_k$ is a diagonal matrix known as *scaling matrix*, whose entries depends on $\nabla\mathcal{F}(\boldsymbol{x}^{(k)})$ [45]. In

particular, consider a splitting of $\nabla\mathcal{F}(\boldsymbol{x})$ into its positive and negative part, as

$$\nabla\mathcal{F}(\boldsymbol{x}) = V(\boldsymbol{x}) - U(\boldsymbol{x}), \tag{1.40}$$

where $V(\boldsymbol{x}) > 0$ and $U(\boldsymbol{x}) \geq 0$. Then the elements on the diagonal of $\boldsymbol{S}_k$ are computed as

$$s_{i,i}^{(k)} = \min\left(\rho_k, \max\left(\frac{1}{\rho_k}, \frac{\boldsymbol{x}_i^{(k)}}{V_i(\boldsymbol{x}^{(k)})}\right)\right), \tag{1.41}$$

where $\{\rho_k\}_{k\geq 0}$ is a decreasing positive sequence, usually defined as $\rho_k = \sqrt{1 + k^{-2.1} \cdot 10^{15}}$ [46].

In our experiments, the step sizes η_k are selected by the alternating Barzilai-Borwein (BB) method [47], which has been shown to accelerate convergence in many imaging applications [43, 48]. In particular, let:

$$\eta_{k+1}^{BB1} = \arg\min_{\eta_k \in \mathbb{R}} ||(\eta_k \boldsymbol{S}_{k+1})^{-1}\boldsymbol{s}^{(k)} - \boldsymbol{z}^{(k)}||_2 = \frac{\left(\boldsymbol{s}^{(k)}\right)^T \boldsymbol{S}_{k+1}^{-1}\boldsymbol{S}_{k+1}^{-1}\boldsymbol{s}^{(k)}}{\left(\boldsymbol{s}^{(k)}\right)^T \boldsymbol{S}_{k+1}^{-1}\boldsymbol{z}^{(k)}}, \tag{1.42}$$

and

$$\eta_{k+1}^{BB2} = \arg\min_{\eta_k \in \mathbb{R}} ||\boldsymbol{s}^{(k)} - (\eta_k \boldsymbol{S}_{k+1})\,\boldsymbol{z}^{(k)}||_2 = \frac{\left(\boldsymbol{s}^{(k)}\right)^T \boldsymbol{S}_{k+1}\boldsymbol{z}^{(k)}}{\left(\boldsymbol{z}^{(k)}\right)^T \boldsymbol{S}_{k+1}\boldsymbol{S}_{k+1}\boldsymbol{z}^{(k)}}, \tag{1.43}$$

the classical Barzelai-Borwein rules [49] for η_k, where

$$\boldsymbol{s}^{(k)} = \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}, \quad \boldsymbol{z}^{(k)} = \nabla\mathcal{F}(\boldsymbol{x}^{(k+1)} - \nabla\mathcal{F}(\boldsymbol{x}^{(k)}. \tag{1.44}$$

The alternating Barzelai-Borwein method select the step-size η_{k+1} based on the the ratio between η_{k+1}^{BB2} and η_{k+1}^{BB1}. In particular,

$$\begin{cases} \text{if } \frac{\eta_{k+1}^{BB2}}{\eta_{k+1}^{BB1}} \leq \tau_k : \\ \qquad \eta_{k+1} = \min\{\eta_j^{BB2}, \text{ for } j = \max\{1, k+1-m_\eta\}, \ldots, k, k+1\}, \tau_{k+1} = 0.9\tau_k, \\ \text{otherwise:} \\ \qquad \eta_{k+1} = \eta_{k+1}^{BB1}, \tau_{k+1} = 1.1\tau_k, \end{cases} \tag{1.45}$$

where m_η is a non-negative integer and τ_k is a positive real number.

The resulting optimization algorithm, described in detail in Algorithm 2, has a theoretical convergence rate of $\mathcal{O}(\frac{1}{k})$ [50] but, for suitable choices of the scaling matrix $\boldsymbol{S}_k$, it is empirically very well comparable with other state-of-the-art algorithms [48, 50]. For this reason, the SGP algorithm is commonly used among the SparseCT community.

Algorithm 2 Scaled Gradient Projection (SGP)

task minimize smooth function $\mathcal{F}(\boldsymbol{x})$ with $\nabla\mathcal{F}(\boldsymbol{x}) = V(\boldsymbol{x}) - U(\boldsymbol{x})$, $V(\boldsymbol{x}) > 0$, $U(\boldsymbol{x}) \geq 0$
input a starting guess $\boldsymbol{x}^{(0)} \geq \boldsymbol{0}$, $\gamma, \sigma \in (0,1)$, $0 < \eta_{min} \leq \eta_{max}$, $k_{max} \in \mathbb{N}$
for $k \leftarrow 1 : k_{max}$ **do**
  compute $\boldsymbol{g}^{(k)} = \nabla\mathcal{F}(\boldsymbol{x}^{(k)}$
  compute $\boldsymbol{S}^{(k)}$ as in (1.41)
  define $\eta_k \in [\eta_{min}, \eta_{max}]$ with Barzelai-Borwei
  compute $\boldsymbol{d}^{(k)} = \mathcal{P}_+\left(\boldsymbol{x}^{(k)} - \eta_k \boldsymbol{S}_k \boldsymbol{g}^{(k)}\right) - \boldsymbol{x}^{(k)}$
  find α_k by the backtracking algorithm:
  $\alpha_k = 1$
  while $\mathcal{F}(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}) > \mathcal{F}(\boldsymbol{x}^{(k)}) + \sigma\alpha_k \left(\boldsymbol{g}^{(k)}\right)^T \boldsymbol{d}^{(k)}$ **do**
    $\alpha_k \leftarrow \gamma\alpha_k$
  end while
  update $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$
end for
return $\boldsymbol{x}^{(k+1)}$

Chambolle-Pock (CP) The Chambolle-Pock (CP) algorithm is arguably the most used iterative method to solve TV-regularized inverse problems. In its general formulation, it can be used to minimize functionals of the form:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{D}\boldsymbol{x}) + G(\boldsymbol{x}), \tag{1.46}$$

where both F and G are proper, convex, lower semi-continuous functions from $\mathbb{R}^n$ to $\mathbb{R}$. Note that there is no restriction on the smoothness of neither F and G, thus the method can be applied to the general TV-regularized formulation (1.31) by setting $F(\boldsymbol{D}\boldsymbol{x}) = \lambda\mathrm{TV}(\boldsymbol{x})$ and $G(\boldsymbol{x}) = \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta)$. To simplify the computation, in this paragraph, we will consider $\mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta) = \frac{1}{2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta||_2^2$, but similar arguments hold for general functions $\mathcal{J}$. The CP algorithm considers the primal-dual formulation of (1.46):

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \max_{\boldsymbol{z}\in\mathbb{R}^n} \boldsymbol{z}^T \boldsymbol{D}\boldsymbol{x} + G(\boldsymbol{x}) - F^*(\boldsymbol{z}), \tag{1.47}$$

where F^* is the convex conjugate of F [51]. Given a starting guess for both the primal variable $\boldsymbol{x}^{(0)}$ and the dual variable $\boldsymbol{z}^{(0)}$, the update rule is as follows:

$$\begin{cases} \boldsymbol{z}^{(k+1)} = (I + \sigma\partial F^*)^{-1}\left(\boldsymbol{z}^{(k)} + \sigma\boldsymbol{D}\bar{\boldsymbol{x}}^{(k)}\right) \\ \boldsymbol{x}^{(k+1)} = (I + \tau\partial G)^{-1}\left(\boldsymbol{x}^{(k)} - \tau\boldsymbol{D}^*\boldsymbol{z}^{(k+1)}\right) \\ \bar{\boldsymbol{x}}^{(k+1)} = \boldsymbol{x}^{(k+1)} + \Theta\left(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right) \end{cases} \tag{1.48}$$

where $\bar{\boldsymbol{x}}^{(0)} = \mathbf{0}$, $\Theta \in [0,1]$ is a parameter that we set equal to 1 in the experiments, while $\sigma > 0$, $\tau > 0$ are defined as $\sigma = \tau = \frac{1}{||\boldsymbol{M}||_2}$, where $\boldsymbol{M}$ is the operator obtained by concatenating $\boldsymbol{K}$ and $\boldsymbol{D}$, and $||\boldsymbol{M}||_2$ is computed by the power method in all the experiments. Explicit computation of $(I + \sigma\partial F^*)^{-1}$ and $(I + \tau\partial G)^{-1}$ for the TV-regularized inverse problem can be found in [42], and in Algorithm 3. The theoretical convergence rate for the CP algorithm is $\mathcal{O}(\frac{1}{k^2})$, which is slightly faster than that of SGP but emprically the performance of the two methods is comparable.

Algorithm 3 Chambolle-Pock for Total Variation minimization (CP-TV)

task minimize convex function $\mathcal{F}(\boldsymbol{x}) = F(\boldsymbol{D}\boldsymbol{x}) + G(\boldsymbol{x})$
input the operator $\boldsymbol{K}$, corrupted data $\boldsymbol{y}^\delta \in \mathbb{R}^m$, regularization parameter $\lambda > 0$.
define $\boldsymbol{M} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{D} \end{bmatrix}$, $\Gamma = ||\boldsymbol{M}||_2$, $\nu = \frac{||\boldsymbol{K}||_2}{||\boldsymbol{M}||_2}$, $\eta = 2 \cdot 10^{-3}$, $\tau = \sigma = \Gamma^{-1}$, $\Theta = 1$
initialize primal variables $\boldsymbol{x}^{(0)} = \bar{\boldsymbol{x}}^{(0)} = \mathbf{0}$, dual variables $\boldsymbol{s}^{(0)} = \mathbf{0} \in \mathbb{R}^m$, $\boldsymbol{q}^{(0)} = \mathbf{0} \in \mathbb{R}^{2m}$
for $k \leftarrow 1 : k_{max}$ **do**
  update dual variable $\boldsymbol{q}^{(k+1)} = \frac{\boldsymbol{q}^{(k)} + \sigma(\boldsymbol{K}\bar{\boldsymbol{x}} - \boldsymbol{y}^\delta)}{1 + \sigma\lambda}$
  define $\bar{\boldsymbol{s}}^{(k+1)} = \boldsymbol{s}^{(k)} + \sigma\boldsymbol{D}\boldsymbol{x}^{(k)}$
  update dual variable $\boldsymbol{s}^{(k+1)} = \frac{\lambda\bar{\boldsymbol{s}}^{(k+1)}}{\max\{\lambda\mathbf{1}, |\bar{\boldsymbol{s}}^{(k+1)}|\}}$
  update primal variable $\boldsymbol{x}^{(k+1)} = \mathcal{P}_+\left(\boldsymbol{x}^{(k)} - \tau(\boldsymbol{K}^T\boldsymbol{q}^{(k+1)} + \nu\boldsymbol{D}^T\boldsymbol{s}^{(k+1)})\right)$
  update $\bar{\boldsymbol{x}}^{(k+1)} = \boldsymbol{x}^{(k+1)} + \Theta(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)})$
end for
return $\boldsymbol{x}^{(k+1)}$

To conclude this Section, we underline the major drawback of iterative algorithms, especially those used to optimize TV-regularized inverse problems, which motivated my PhD research. Even if the quality of the optimal point of those problems exhibits great accuracy and superior stability at input noise, their best convergence rate is sublinear, which implies that the number of iterations required to get close enough to the minimum point is usually large. However, the medical application often requires computing the solution in a very low time (of the order of at most one minute, if not real-time). This is impossible to achieve with modern algorithms and technology, thus forcing medical protocols to use direct methods such as FBP, with the instability issues described above.

1.3.3 Neural Networks

The advent of neural networks to the field of inverse problems is relatively recent. Although they were theoretically introduced at the end of the last century, the technology was not capable of handling the computational demands required for image reconstruction tasks until the beginning of the last decade. This was made possible by the increased computing power

of Graphics Processing Units (GPUs), enabling the efficient implementation of large-scale convolutional neural networks for various image-to-image tasks. Modern GPUs are capable of performing parallel computations at exceptional speeds, processing multiple images in a fraction of a second. In this thesis, we will primarily work with neural networks, particularly convolutional neural networks (CNNs) [24, 52, 53]. The beginning of this Section is dedicated to a theoretical introduction to the neural network architectures used throughout the thesis, highlighting their main advantages and disadvantages.

Definition of Neural Networks To begin, let's define what a neural network is.

Definition 1.1. Given a neural network architecture $\mathcal{A} = (\nu, \boldsymbol{S}, \rho)$ where $\nu = (\nu_0, \nu_1, \dots, \nu_L) \in \mathbb{N}^{L+1}$, $\nu_0 = m, \nu_L = n$, defines the *width* of each layer, $\boldsymbol{S} = (\boldsymbol{S}_{1,1}, \dots, \boldsymbol{S}_{L,L}), \boldsymbol{S}_{j,k} \in \mathbb{R}^{\nu_j \times \nu_k}$ is the set of matrices representing the *skip connections*, and $\rho : \mathbb{R} \to \mathbb{R}$ is a non-linear *activation function* that is applied element-wise when calculated on a vector, we define the parametric family of neural network with architecture $\mathcal{A}$, parameterized by $\Theta \in \mathbb{R}^s$, as

$$\mathcal{F}_\Theta^{\mathcal{A}} = \{\Psi_\Theta : \mathbb{R}^m \to \mathbb{R}^n; \Theta \in \mathbb{R}^s\},$$

where for any $\boldsymbol{y}^\delta \in \mathbb{R}^m$, $\Psi_\Theta(\boldsymbol{y}^\delta) = \boldsymbol{z}^L$ is given by:

$$\begin{cases} \boldsymbol{z}^0 = \boldsymbol{y}^\delta \\ \boldsymbol{z}^{l+1} = \rho(\boldsymbol{W}^l \boldsymbol{z}^l + \boldsymbol{b}^l + \sum_{k=1}^{l} \boldsymbol{S}_{l,k} \boldsymbol{z}^k) \quad \forall\, l = 0, \dots, L-1 \end{cases} \tag{1.49}$$

and $\boldsymbol{W}^l \in \mathbb{R}^{\nu_{l+1} \times \nu_l}$ and $\boldsymbol{b}^l \in \mathbb{R}^{\nu_{l+1}}$ is the *weight matrix* and the *bias vector* at layer l, respectively.

It's important to note that when no specific structure is assumed for the weight matrices $\boldsymbol{W}^l$, the total number of parameters, denoted as s, can be calculated as:

$$s = \sum_{l=0}^{L-1} \underbrace{\nu_l \nu_{l+1}}_{\text{param. of } \boldsymbol{W}^l} + \underbrace{\nu_{l+1}}_{\text{param. of } \boldsymbol{b}^l}. \tag{1.50}$$

This number can become exceedingly large, especially in problems with a high number of pixels, even for relatively small values of L. To reduce the computational complexity associated with such models, the authors in [52] proposed to restrict $\boldsymbol{W}^l$ to be a convolution matrix for all $l = 0, \dots, L-1$. In particular, for each layer l, consider a set of c_l 2-dimensional convolutional kernels with a shape of $\kappa \times \kappa$. Then, if $\boldsymbol{W}^l$ represents the matrix associated with the convolution operation between this set of kernels and the output $\boldsymbol{z}^l$ of the previous layer, the number of free parameters in $\boldsymbol{W}^l$ is $c_l \kappa^2 \nu_l$. This number is significantly lower than $\nu_l \nu_{l+1}$, especially when $\kappa \ll \nu_l$, $c_l \ll \nu_l$, as it is common in real applications. With this constraint in place, the total number of parameters in the model becomes:

$$s = \sum_{l=0}^{L-1} \underbrace{c_l \kappa^2 \nu_l}_{\text{param. of } \boldsymbol{W}^l} + \underbrace{\nu_{l+1}}_{\text{param. of } \boldsymbol{b}^l}, \tag{1.51}$$

resulting in a linear growth in the total number of parameters with respect to the network's width, as opposed to quadratic growth. The resulting model, known as a Convolutional Neural Network (CNN), serves as the foundation for nearly all modern neural network architectures, including the networks we will explore in the following Sections. A variant of classical CNNs are the *strided Convolutional Neural Networks*. In particular, each convolutional layer l is associated with a parameter $s_l \in \mathbb{N}$, called the *stride*, which represents the amount of downscaling applied to the output of the layer $\boldsymbol{z}^{l+1}$ before the next layer's application. If the processed image after l layers, denoted as $\boldsymbol{z}^l$ and referred to as the *feature map* in CNN literature, has dimensions (m_l, n_l), then the dimensions of the next feature map $\boldsymbol{z}^{l+1}$, when processed by a strided convolutional layer with stride s_l, become $(m_{l+1}, n_{l+1}) = \left(\frac{m_l}{s_l}, \frac{n_l}{s_l}\right)$. It's important to note that classical convolutional neural networks can be considered as strided CNNs with a stride of $s_l = 1$ for all $l = 0, 1, \ldots, L-1$. The number of *scales* in a Convolutional Neural Network refers to the number of convolutional layers with a stride greater than 1. Consequently, a model is termed *multi-scale* when it includes at least one layer with a stride greater than 1, and *single-scale* otherwise. Due to their popularity it's common, with a slight abuse of notation, to refer to them as *convolutional neural networks (CNNs)* without specifying 'strided'.

A crucial property of Convolutional Neural Networks (CNNs) is their Receptive Field (RF) [54]. To understand the concept of RF, let's take as example a model with L convolutional layers, each having a kernel size of $\kappa_l = 3$, as depicted in Figure 1.6. It's evident that each pixel in the feature map of the second layer is influenced only by a 3×3 portion of the first layer (the input of the network). Similarly, each pixel in the feature map of the third layer is influenced by a 3×3 portion of the previous layer, and consequently by a 5×5 portion of the input image. The number of input image pixels that affect the value of each pixel in the l-th feature map is what we call the receptive field of the l-th layer. By continuing this process through the network's layers to the output, we can compute the receptive field of the network, which represents the number of input image pixels that influence the network's output. This measurement is crucial because when reconstructing corrupted data containing artifacts, understanding the RF helps ensure that the model captures and addresses these artifacts accurately. In particular, when the artifacts are local, a small receptive field is enough to give the model the ability to distinguish between the artifact and the image features for the reconstruction, while when the artifacts are global, such as in the SparseCT reconstruction, a large RF is required to produce an accurate reconstruction.

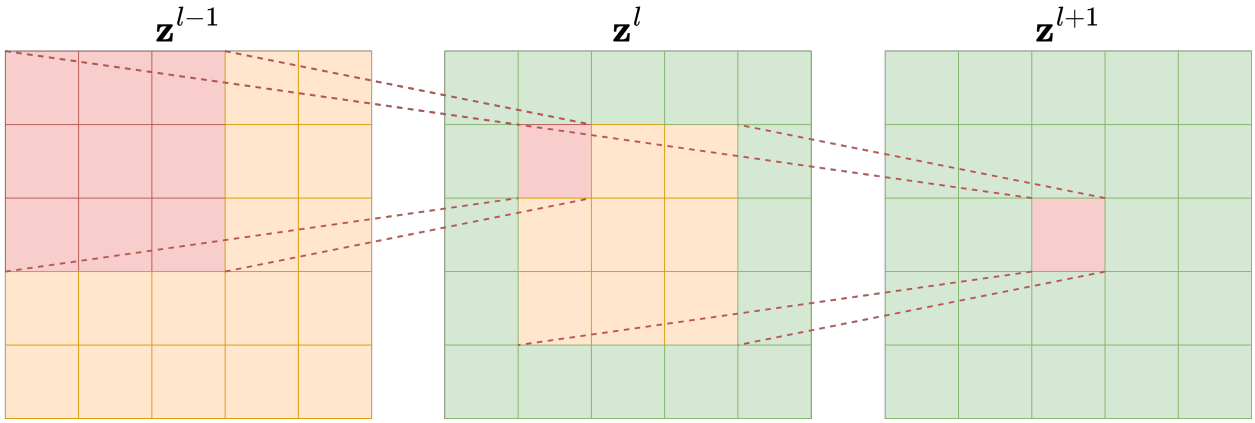


Figure 1.6: A diagram to visually describe the receptive field in a three-layer neural network, each with a kernel size of $\kappa_l = 3$. The red pixel in the third layer is influenced only by the yellow square of the second layer, which is influenced by the yellow square of the first layer. As a result, the receptive field of this network is equal to 5.

Since we are interested in comparing neural network architectures in terms of their receptive field, we need to derive a formula to compute it for any given network. For each layer l, let κ_l and s_l be its kernel dimension and stride, respectively. Moreover, let r_l be the receptive field, where the receptive field of the input layer is $r_0 = 1$ by definition. The value of r_l can be computed with the recursive formula [54]:

$$r_l = r_{l-1} + A_l, \tag{1.52}$$

where A_l is the non-overlapping area between subsequent filter applications. Note that A_l can be simply computed as

$$A_l = (k_l - 1) \prod_{i=1}^{l} s_i, \tag{1.53}$$

which implies that the receptive field at each l-th layer is:

$$\begin{cases} r_0 = 1 \\ r_l = r_{l-1} + (k_l - 1) \prod_{i=1}^{l} s_i. \end{cases} \tag{1.54}$$

Equation (1.54) shows that the receptive field scales linearly with the depth of the network if the kernel dimension is fixed, while it is exponentially related to the stride. For this reason, utilizing strided convolutional layers exponentially enlarges the receptive field of the model.

Our architectures We will now briefly present the three main architectures considered in this thesis, namely the *3-layer single-scale fully convolutional neural network (3L-SSNet)* [6], the *U-Net* [55] and the *Non-linear Activation-Free Network (NAFNet)* [56].

3L-SSNet The 3L-SSNet [6] is a very simple end-to-end architecture designed to reconstruct images from other images of the same shape. It is a three-layered fully Convolutional Neural Network with a constant channel number, $c_l = 128$ for $l = 0, 1, 2$ and kernels of dimension $(\kappa_1, \kappa_2, \kappa_3) = (9, 5, 3)$. In particular, each layer is a Convolutional layer, followed by a Batch Normalization layer and ReLU activation function. A draft of the structure of the network is reported in Figure 1.7. The network does not contain stided convolutions, hence it works in single-scale mode. As a consequence, its Receptive Field at the output layer is small (15×15) and the resulting model will be unable to accurately recover global artifacts, if present.

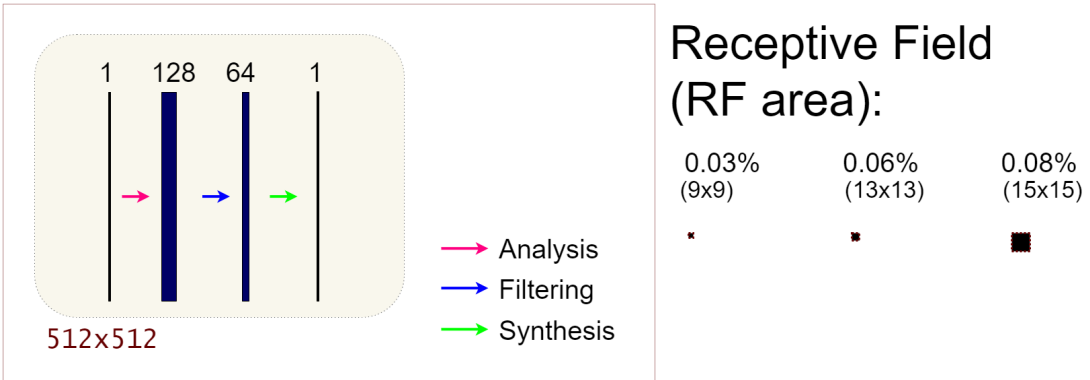


Figure 1.7: On the left: graphical representation of 3L-SSNet architecture; on the right: details on the receptive fields for each of the three layers of the network (RF percentage with respect to the input 512x512 image and size of RF). The name of the three layers follows the notation in [57].

U-Net The U-Net is a popular multi-scale Convolutional Neural Network architecture introduced in [55], that operates efficiently whenever the input image shows global artifacts. It is a fully convolutional neural network with a symmetric encoder-decoder structure and strided convolutions to enlarge its receptive field. The strides in the encoder layers naturally divide the network into distinct levels of resolution, to which we will refer as g, $g = 0, \dots, \mathcal{G}$, where $\mathcal{G} + 1$ is the total number of levels in the network. At each level, a fixed number n_g of blocks $B_1, \dots B_{n_g}$ is applied. Specifically, each block is defined as a convolutional layer with number c_g of channels constant along the level, followed by a batch normalization layer and then by a ReLU activation function. Given a baseline number of convolutional channels c_0 (that corresponds to the number of channels in the first level), we compute c_g for the next levels with the recursive formula $c_{g+1} = 2c_g$, $g = 0, \dots, L - 1$. In particular,

we fixed $\mathcal{G} = 4$, $n_0 = \cdots = n_3 = 3$ and $c_0 = 64$ in the experiments. As already said, the decoder is symmetric to the encoder, with upsampling convolutional layers instead of strided convolutions. Moreover, to maintain high-frequency information, skip connections are added between the last layer at each level of the encoder and the first layer at the correspondent level of the decoder. To reduce the number of parameters with respect to the original architecture [58], it is common to implement the skip connections as additions instead of concatenations. A residual connection is added between the input layer and the output layer too, following the implementation in [24]. As a result, the output of the model can be described as

$$\Psi_\Theta(\boldsymbol{y}) = \boldsymbol{y} + \Phi_\Theta(\boldsymbol{y}), \tag{1.55}$$

which implies that the network has to learn the residual mapping between the input and the expected output. For this reason, this model has been named *Residual U-Net (ResU-Net)* in the literature. The importance of the residual connection has been observed in the theoretical work [59] by Han et al., where the authors proved that the residual manifold containing the artifacts is easier to learn than the true image manifold. For this reason, we will always employ a residual U-Net architecture (depicted in details in Figure 1.8) throughout this thesis.

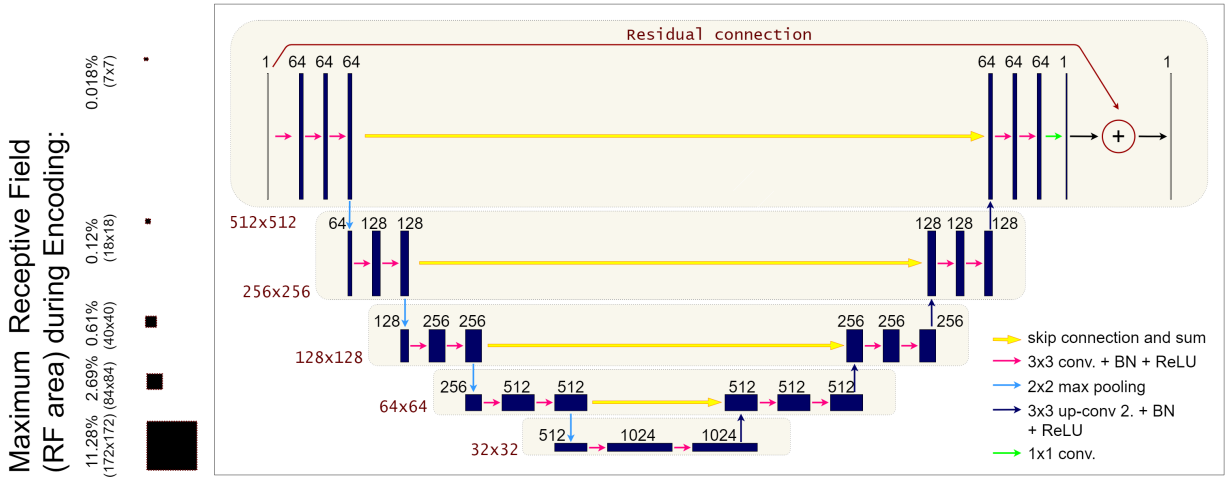


Figure 1.8: On the right: graphical representation of ResUNet architecture; On the left: details on the maximum receptive fields for each of the five levels of the network encoder (RF percentage respect to the input 512x512 image and size of RF).

NAFNet The Nonlinear Activation Free Network (NAFNet), as introduced in [56], is a state-of-the-art model designed for image restoration and inspired by the U-Net architecture. In particular, as illustrated in Figure 1.9, the high-level diagram of the NAFNet aligns seamlessly with that of the U-Net. Both architectures feature multiple levels of blocks of layers arranged within a symmetric encoder-decoder structure. Additionally, to maintain

information integrity, a skip connection is included between the encoder and the decoder at each level of resolution. However, the distinguishing feature of the NAFNet architecture is the construction of each individual block, which differs from the U-Net, as depicted in the left portion of Figure 1.9. Specifically, while each block in the U-Net architecture comprises a concatenation of a convolution layer, a ReLU activation function, and batch normalization, NAFNet introduces a more intricate structure in each block. These blocks involve convolutions, gates, normalization layers, and skip connections. It's worth noting that, in line with its name, NAFNet does not make use of any non-linear activation function, instead relying on the complexity introduced by the gates for its expressiveness.

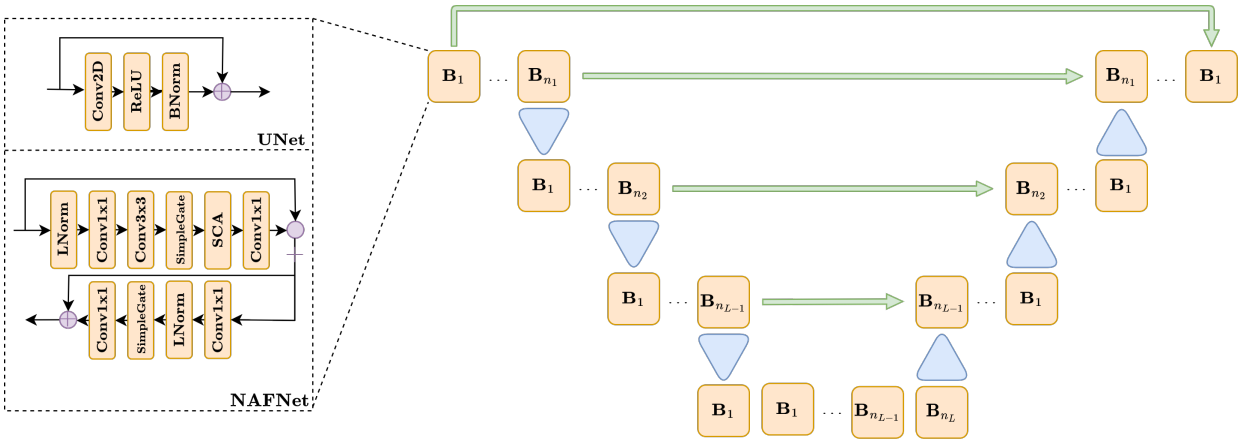


Figure 1.9: A diagram representing the UNet and NAFNet architectures.

Training a Neural Network Given a neural network architecture, a fundamental step in its development is to *train* it. To do that, consider a dataset $\{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1}^N \subset \mathcal{Y} \times \mathcal{X}$, together with a loss function $\ell : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$. Training a network means finding the element $\Psi_{\Theta^*} \in \mathcal{F}_\Theta^{\mathcal{A}}$ that minimizes the average of the loss function over the whole dataset, i.e.

$$\Psi_{\Theta^*} = \underset{\Psi_\Theta \in \mathcal{F}_\Theta^{\mathcal{A}}}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \ell\left(\Psi_\Theta(\boldsymbol{y}_i), \boldsymbol{x}_i\right). \tag{1.56}$$

To do that, it is common to consider gradient-based stochastic optimization algorithms, such as *Stochastic Gradient Descent (SGD)* [60] or the *Adaptive Moment Estimation method (Adam)* [61]. Since the optimization problem (1.56) is highly non-convex, the solution computed by those algorithms probably won't be a global optimum. However, extended experiments in the field show that the computed solution is close enough to the global optimum, especially if residual architectures are employed [62]. In the following, we will always consider the L_2-norm as a loss function as it is common in the literature, meaning that $\ell\left(\Psi_\Theta(\boldsymbol{y}_i), \boldsymbol{x}_i\right) = ||\Psi_\Theta(\boldsymbol{y}_i) - \boldsymbol{x}_i||_2^2$.

Neural Networks for Inverse Problems Neural networks can be applied to solve inverse problems in multiple ways. The most common is to directly learn a mapping from the corrupted data $\boldsymbol{y}^\delta = \boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}$ to the corresponding $\boldsymbol{x}^{gt}$ solution. This approach, named *end-to-end*, is able to reconstruct the data with remarkable accuracy, basically incomparable to any other reconstruction technique such as regularized variational methods. Moreover, when the architecture of the model is a CNN, this operation can be done almost instantaneously if compared with the amount of time required to execute a whole iterative algorithm to convergence. Note that, in this case, particular attention has to be given to the solution of SparseCT inverse problems. Indeed, the Fourier slice Theorem shows that the Radon transform can be seen as a coordinate change in the Fourier domain. As a consequence, the mapping between the measured sinogram $\boldsymbol{y}^\delta$ and the solution $\boldsymbol{x}^{gt}$ is highly non-local, thus requiring a model with a receptive field that is as large as the sinogram itself. For this reason, it is common in literature to pre-process the sinogram by applying a transform that maps $\boldsymbol{y}^\delta$ back to the image space, in such a way that the reconstruction map becomes local and can be performed by a CNN. This way, the neural network acts as a post-processing operator and is named *Learnt Post-Processing (LPP)* [6] in the literature. A common choice is to use the FPB as a pre-processing since, given that the Radon operator is normal-convolutional, the application of the back-projection to the sinogram makes the transformation a convolution, which can be easily solved by a convolutional neural network.

A drawback of the end-to-end approach is that, when unexpected noise is present in the input of the network, the quality of the reconstruction degrades, as we will deeply discuss in the following. My research was entirely dedicated to attenuating that particular kind of instability, by combining techniques from the world of iterative algorithms with careful architecture design.

Hybrid techniques, defined as the set of all the methods that combine neural networks in variational algorithms to produce accurate and stable results, can be seen as the intersection of the world of variational methods with the world of end-to-end neural network algorithms, as depicted in Figure 1.10. Most state-of-the-art techniques reside in this class, ranging from Unrolled methods [63, 64, 65] to Deep Image Prior (DIP) [66, 67], Plug-and-Play (PnP) [68, 69] and Deep Generative Prior (DGP) [70, 71]. The common aspect of all those methods is that they consider the physics of the process (described by the forward operator $\boldsymbol{K}$) in the algorithm, with the intent of minimizing the impact of the noise and improving the consistency of the solution. Indeed, it is known [72] that the instability to noise observed with the end-to-end neural networks comes from the fact that, minimizing the average L_2 loss among the training dataset, resulting in a a model that is too specific for the considered task and, when unexpected noise is added to the input, since the model does not know the physics of the acquisition process, it generates a solution that is far from the true image $\boldsymbol{x}^{gt}$.

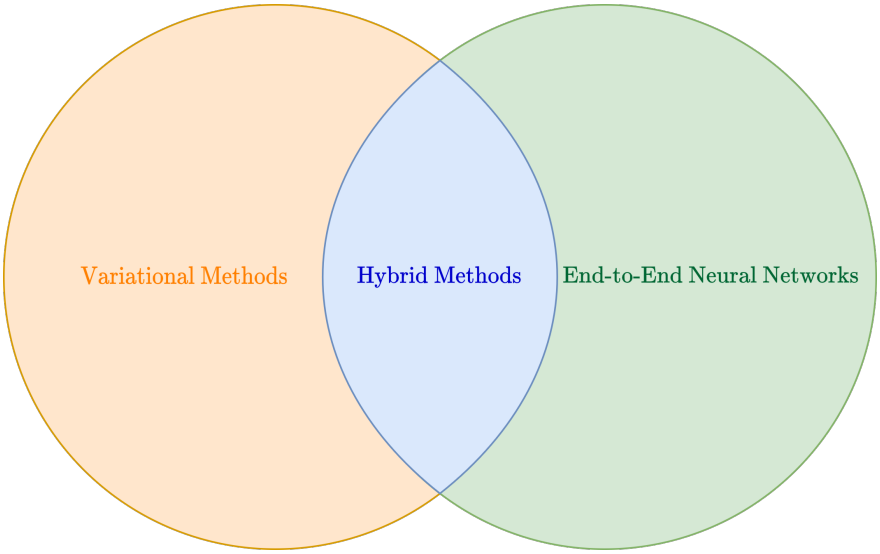


Figure 1.10: A diagram of the neural network techniques for inverse problems.

1.4 Datasets and Metrics

In this Section, we introduce the experimental setup frequently considered in the following Chapters, with particular attention to the considered datasets, the metrics.

1.4.1 Datasets

We present here three datasets that will be used for the experiments in the following Chapters: a slightly modified version of the GoPro dataset [73], containing photographic images and mainly used for experiments in image deblurring, the widely used Mayo's dataset [74], containing a large number of Computed Tomography acquisitions of real patients, and the COULE dataset, introduced in [7], containing synthetic images representing overlapping ellipsis and lines, with the intent of simulating the interior of a human body.

GoPro The GoPro dataset contains 3,214 RGB images with a resolution of 1280×720, representing the frames of videos recorded by a GoPro camera, released by NVIDIA in 2017. To reduce the computational requirements to train our models, all the images have been cropped to 256×256 patches, converted into greyscale, and normalized into $[0, 1]$. The dataset has then been split into a training set containing 2103 images and a test set of 1111 images. We made the modified dataset public at `https://huggingface.co/datasets/TivoGatto/gopro_small`.

Figure 1.11: Ground-truth images from the modified GoPro dataset, used for the training of our models.

Mayo's dataset To test the methods on real medical images, we have downloaded the widely used AAPM Low Dose CT Grand Challenge dataset by the Mayo Clinic [75]. It contains images of the human abdomen, reconstructed from full-dose acquisitions, in a resolution of 512×512 pixels. In Figure 1.12 we depict one image with two zooms-in highlighting areas with different anatomical structures, such as pulmonary details, Sections of ribs, and low-contrast inter-costal muscles. As observable from Figure 1.12, real full-dose medical images still present little noise and slightly visible streaking artifacts. This makes it quite difficult to compute reliable metrics on the reconstructions by referencing them as ground truth.

COULE To fully exploit the full-reference image quality assessment metrics and validate our experiments, we created synthetic images and built few-view CT simulations. In particular, the Constrasted Overlapping Uniform Lines and Ellipses (COULE) dataset contains 430 sparse-gradient gray-scale images of size 256×256 with many overlying objects, varying in size and contrast with respect to the background. The left image of Figure 1.12 shows one image of the data set as an example. The whole data set is downloadable from `www.kaggle.com/loiboresearchgroup/coule-dataset`.

1.4.2 Metrics

To quantitatively compare the results of our methods with other state-of-the-art techniques, we measured reconstruction error based on three, commonly used metrics: the *Root Mean Squared Error (RMSE)*, the *Peak Signal-to-Noise Ratio (PSNR)* and the *Structural Similarity Image Metrics (SSIM)*. In the following, we will denote by $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ the reconstructed signal by the considered model, while $\boldsymbol{x}^{gt}$ represents the true solution we aim to recover.

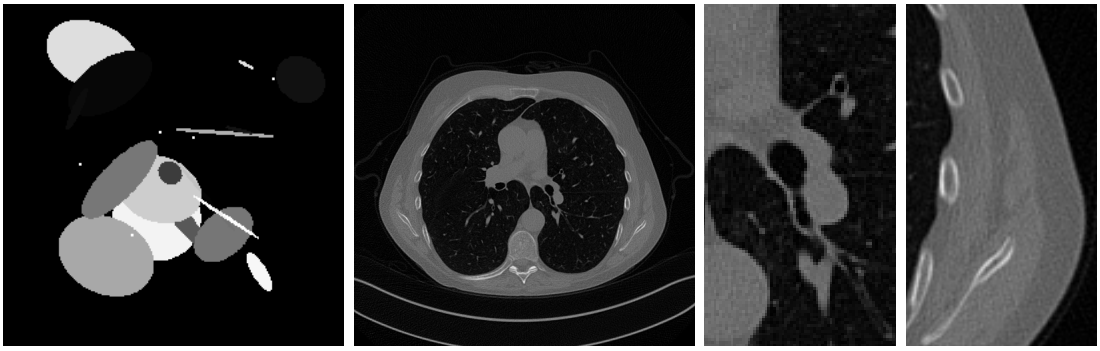


Figure 1.12: Ground-truth images from the COULE (on the left) and the Low Dose Mayo (center) data sets, with two zoomed crops (on the right) on regions with different anatomical structures.

RMSE The Root Mean Squared Error (RMSE) is among the most popular metrics that measure the quality of the reconstructed signal. It is defined as:

$$RMSE(\tilde{\boldsymbol{x}}, \boldsymbol{x}^{gt}) = \sqrt{\frac{||\tilde{\boldsymbol{x}} - \boldsymbol{x}^{gt}||_2^2}{n}}. \tag{1.57}$$

Clearly, the RMSE is simply the Euclidean distance between vectors $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}^{gt}$, normalized by the dimensionality. As a consequence, it follows the mathematical definition of distance and, in particular, the smaller it is, the closer the vectors are.

PSNR The Peak Signal-to-Noise Ratio (PSNR) is a widely used metric to evaluate the amount of noise remaining in the signal after the reconstruction. If both $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}^{gt}$ are in the range $[0, 1]$, it is defined as:

$$PSNR(\tilde{\boldsymbol{x}}, \boldsymbol{x}^{gt}) = -10 \log_{10} \left(||\tilde{\boldsymbol{x}} - \boldsymbol{x}^{gt}||_2^2 \right). \tag{1.58}$$

Due to the logarithm, the PSNR does not represent a distance and, due to the negative sign, the quality of the reconstruction is better if the PSNR is large. Note that, since the squared Euclidean distance between the reconstruction $\tilde{\boldsymbol{x}}$ and the ground-truth $\boldsymbol{x}^{gt}$ is processed by a logarithm, a distance of few units in the PSNR signifies an exponential distance between the two signals.

SSIM The Structural Similarity Image Metrics (SSIM) [76] is a metric used to evaluate the qualitative similarity between two images. Given a parameter $\kappa \in \mathbb{N}$, consider a set of N overlapping windows of dimension $\kappa \times \kappa$, in the same position between $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}^{gt}$, denoted as $\boldsymbol{w}_i^{\tilde{\boldsymbol{x}}}$ and $\boldsymbol{w}_i^{\boldsymbol{x}^{gt}}$ for $i = 1, \dots, N$, respectively. Then, if $\boldsymbol{\mu}_i^{\tilde{\boldsymbol{x}}}$ and $\boldsymbol{\mu}_i^{\boldsymbol{x}^{gt}}$ are the pixel mean of $\boldsymbol{w}_i^{\tilde{\boldsymbol{x}}}$, $\boldsymbol{w}_i^{\boldsymbol{x}^{gt}}$, $\boldsymbol{\sigma}_i^{\tilde{\boldsymbol{x}}}$ and $\boldsymbol{\sigma}_i^{\boldsymbol{x}^{gt}}$ are the pixel standard deviation of $\boldsymbol{w}_i^{\tilde{\boldsymbol{x}}}$, $\boldsymbol{w}_i^{\boldsymbol{x}^{gt}}$, then

$$SSIM(\tilde{\boldsymbol{x}}, \boldsymbol{x}^{gt}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\left(2\boldsymbol{\mu}_i^{\tilde{\boldsymbol{x}}} \boldsymbol{\mu}_i^{\boldsymbol{x}^{gt}} + c_1\right) \left(2\boldsymbol{\sigma}_i^{\tilde{\boldsymbol{x}}\boldsymbol{x}^{gt}} + c_2\right)}{\left((\boldsymbol{\mu}_i^{\tilde{\boldsymbol{x}}})^2 + (\boldsymbol{\mu}_i^{\boldsymbol{x}^{gt}})^2 + c_1\right) \left((\boldsymbol{\sigma}_i^{\tilde{\boldsymbol{x}}})^2 + (\boldsymbol{\sigma}_i^{\boldsymbol{x}^{gt}})^2 + c_2\right)}. \tag{1.59}$$

Importantly, the SSIM is always in the range $[0, 1]$ and, since it measures the *similarity* between two images, the higher the SSIM the more similar are the two images. An SSIM equal to 1 represents perfectly equal signals.

Structure of the Thesis In the following Chapters, we will introduce a set of techniques, developed during my PhD, to alleviate the problems related with the instability of the solution of ill-posed inverse problems to unexpected noise at input. It is organized as follows: in Chapter 2 we introduce the concept of reconstructors as functions to solve inverse problems, we analyze their main property and we propose a technique, named *stabilizer*, to increase the stability of reconstructors. We also provide extended experiments on the Deblurring inverse problem. In Chapter 3, we extend the ideas discussed in the previous Chapter to the SparseCT inverse problem, designing an algorithm to stably reconstruct medical images from sparse sinograms that does not require a dataset of ground-truth to be trained. We name that model RISING. In Chapter 4 we realize that a good solution to the SparseCT inverse problem requires the introduction of non-convex regularizers such as the ℓ_p-norm for $0 < p < 1$. To this aim, we extend the methods from RISING to the non-convex setup in an algorithm named *T_pV-RISING*.

Chapter 2

Neural Networks as reconstructors to solve ill-conditioned inverse problems

In this Chapter, we address the linear inverse problems presented in the previous Chapter (1.16):

$$\boldsymbol{y}^{\delta} = \boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}, \qquad ||\boldsymbol{e}||_2 \leq \delta, \tag{2.1}$$

where we assume that $\boldsymbol{K} \in \mathbb{R}^{m \times n}$ is a full-rank matrix discretizing a continuous linear operator not satisfying (HC3), such as deblurring [17], with $m \geq n$. Moreover, we assume that the noise $\boldsymbol{e}$ is sampled from a zero-mean Gaussian distribution with variance $\sigma^2\boldsymbol{I}$. We already noted in the previous Chapter that, since $\boldsymbol{K}$ does not satisfy (DPC), computing a solution of (2.1) is very challenging when noise affects the data.

Traditional model-based approaches tackle this problem as the minimization of an objective function containing a data-fit term and a regularization prior, with possible further constraints on the solution, as remarked in Section 1.3.2. These terms theoretically grant stability, but, in general, the computational time required by iterative solvers is high and it may be necessary to choose many parameters, tuning them on the data.

In the last few years, as described in Section 1.3.3, end-to-end neural networks have been introduced with great success for the solution of problem (2.1), since they are capable of achieving greater accuracy than iterative regularized methods [68, 77, 78]. However, their high accuracy is obtained at the expense of robustness with respect to noise on the input data. In particular, they often produce poor results when applied to data corrupted by noise different from the one learned in the training phase. Such a feature is usually referred to as *network instability*. Some authors have studied the behavior of neural networks for the solution of under-determined imaging inverse problems (i.e. $m < n$ in equation (2.1)), in the presence of noise on the data [72, 79, 80, 81, 82, 83, 84, 85]. However, to the best of our knowledge, no works address the case of $m \geq n$ and, moreover, a mathematically grounded understanding is still lacking.

Contributions The contribution of this Chapter is twofold: in the first part, we theoretically investigate the properties of neural networks as solvers of discrete ill-posed problems, by adapting the regularization theory presented in its generality by the authors in [3]. In this setting, we formalize the two fundamental concepts of accuracy and stability for solvers, and we derive a mathematical relation quantifying the trade-off between stability and accuracy and showing that it is not possible to increase a solver's stability without decreasing its accuracy. We also propose three solutions to improve stability while preserving high accuracy, namely the REgularized Neural Network (ReNN), a ground truth-free approach where the target images are the solution computed by a model-based method, the STabilized Neural Network (StNN), where a pre-processing step defined by a few iteration of a regularized iterative algorithm is provided to improve the robustness of the network, and a combination of the two, named StReNN. In the second part of this Chapter, motivated by the promising theoretical results of the first part, we tested the Stabilized Neural Network approach on image deblurring, by considering two pre-processing schemes: one, denoted as *FiNN*, applies a model-free low-pass filter to the datum before passing it as input to the NN, the other, generically called *Stabilized Neural Network (StNN)*, exploits an estimation of the noise statistics and the mathematical modeling of both noise and image corruption process. The accuracy and stability of those methods are compared with a classical, unstabilized neural network, and tested with the three architectures introduced in Section 1.3.3. Figure 2.1 shows a draft of the proposed frameworks, whose robustness is evaluated from a theoretical perspective and tested on an image data set.

The methods described in this Chapter refer to my publications [8] and [10]. My personal contributions to the cited works were in developing the theoretical framework, the code implementation, and in the experimental setup. The code to replicate the experiments can be found in our GitHub repositories for the cited works, at `https://github.com/loibo/ToBeOrNotToBeStable` and
`https://github.com/devangelista2/Ambiguity-in-solving-Inverse-Problems`.

Structure of the Chapter The Chapter is organized as follows. In Section 2.1 we introduce the concept of reconstructor together with the definition of its accuracy and stability; in Section 2.2, after considering as reconstructors Tikhonov and neural networks, we propose a new regularized reconstructor, named ReNN. A new technique to obtain two stabilized reconstructors, StNN and StReNN, is proposed in Section 2.3. In Section 2.4 and Section 2.5 we describe the experiments performed with the proposed new reconstructors. Finally, in Section 2.6 we report some conclusions.

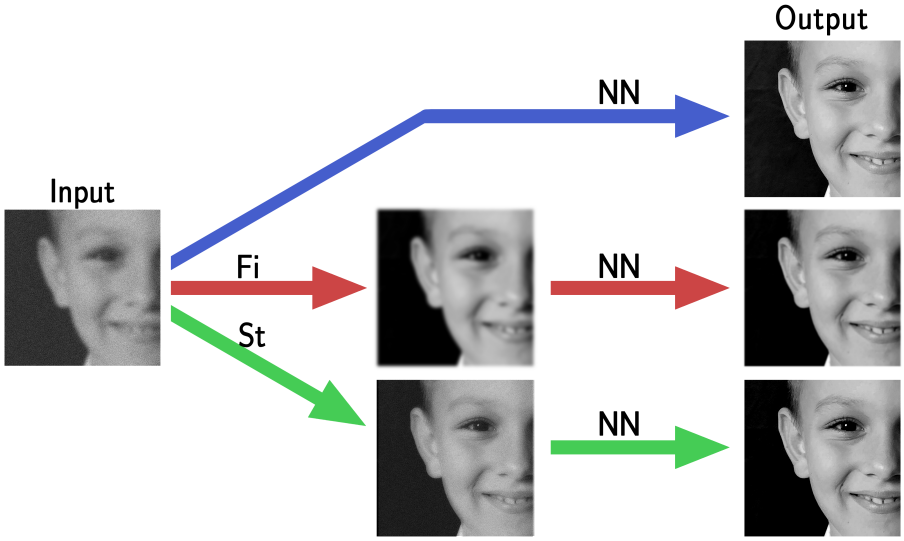


Figure 2.1: A graphical draft highlighting the introduction of pre-processing steps Fi and St defining the proposed frameworks FiNN and StNN, respectively.

2.1 Reconstructors for the solution of linear inverse problems

This Section establishes the theoretical background of the Chapter, providing essential definitions and preliminary results. To improve the readability of this Chapter, however, we start by introducing the notation we will use in the following. We always consider the ground-truth solution $\boldsymbol{x}^{gt}$ to lie in a subset $\mathcal{X}$ of $\mathbb{R}^n$, the set of admissible data. We denote as $\mathcal{Y} = Rg(\boldsymbol{K}, \mathcal{X})$ the range of $\boldsymbol{K}$ over $\mathcal{X}$. We assume $\mathcal{Y}$ to be dense-in-itself (i.e. with no isolated point) so that, for any admissible $\boldsymbol{x}^{gt} \in \mathcal{X}$ and any neighborhood V of $\boldsymbol{y} = \boldsymbol{K}\boldsymbol{x}^{gt}$, there is at least an $\boldsymbol{x}' \in \mathcal{X}$ such that $\boldsymbol{y}' = \boldsymbol{K}\boldsymbol{x}' \in V$. For any $\delta > 0$, we also define $\mathcal{Y}^\delta = \{\boldsymbol{y} + \boldsymbol{e}; \boldsymbol{y} \in \mathcal{Y}, ||\boldsymbol{e}||_2 \leq \delta\}$. Since $\mathcal{Y}$ is dense-in-itself, then also $\mathcal{Y}^\delta$ is dense-in-itself. To solve problem (2.1), we formalize the concept of reconstructor.

Definition 2.1. Any continuous function $\Psi : \mathbb{R}^m \to \mathbb{R}^n$, mapping $\boldsymbol{y}$ to $\boldsymbol{x} = \Psi(\boldsymbol{y})$, is called a reconstructor.

Given a reconstructor Ψ, we define its accuracy.

Definition 2.2. A reconstructor $\Psi : \mathbb{R}^m \to \mathbb{R}^n$ is said to be η^{-1}-accurate, with $\eta > 0$, if

$$\eta = \sup_{\boldsymbol{x}^{gt} \in \mathcal{X}} ||\Psi(\boldsymbol{K}\boldsymbol{x}^{gt}) - \boldsymbol{x}^{gt}||_2.$$

Note that, without any other restriction, η could be infinite. To avoid any issue, we will always consider reconstructors with finite η in the following.

Example 2.1. An accurate reconstructor of problem (2.1) is given by:

$$\Psi^{\dagger}(\boldsymbol{y}) = \boldsymbol{K}^{\dagger}\boldsymbol{y}, \tag{2.2}$$

where $\boldsymbol{K}^{\dagger}$ is the pseudo-inverse matrix introduced in Section 1.3.1. In this case $\Psi^{\dagger} : \mathbb{R}^m \to \mathbb{R}^n$ is ∞-accurate. Indeed, since $\boldsymbol{K}$ is full-rank, we already proved that $\boldsymbol{K}^{\dagger} = (\boldsymbol{K}^T\boldsymbol{K})^{-1}\boldsymbol{K}^T$. Thus,

$$||\Psi^{\dagger}(\boldsymbol{K}\boldsymbol{x}^{gt}) - \boldsymbol{x}^{gt}||_2 = ||(\boldsymbol{K}^T\boldsymbol{K})^{-1}\boldsymbol{K}^T(\boldsymbol{K}\boldsymbol{x}^{gt}) - \boldsymbol{x}^{gt}||_2 = ||(\boldsymbol{K}^T\boldsymbol{K})^{-1}(\boldsymbol{K}^T\boldsymbol{K})\boldsymbol{x}^{gt} - \boldsymbol{x}^{gt}||_2 = 0.$$

However, reconstructors are rarely applied to noise-free data, hence a focus on the robustness of reconstructors with respect to noise is necessary.

Definition 2.3. Let $\delta > 0$ and Ψ be an η^{-1}-accurate reconstructor applied to problem (2.1). We define the δ-stability constant C_{Ψ}^{δ} of Ψ as

$$C_{\Psi}^{\delta} = \sup_{\substack{\boldsymbol{x}^{gt}\in\mathcal{X} \\ ||\boldsymbol{e}||_2\leq\delta}} \frac{||\Psi(\boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}) - \boldsymbol{x}^{gt}||_2 - \eta}{||\boldsymbol{e}||_2}. \tag{2.3}$$

We will consider in the following the realistic case of $C_{\Psi}^{\delta} < \infty$.

Definition 2.4. The reconstructor Ψ is said to be δ-stable for a given $\delta > 0$ if $C_{\Psi}^{\delta} \in [0, 1)$.

A δ-stable reconstructor Ψ does not amplify corruptions which have norm less than δ (as graphically represented in Figure 2.2), since from (2.3) we derive

$$||\Psi(\boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}) - \boldsymbol{x}^{gt}||_2 \leq \eta + C_{\Psi}^{\delta}||\boldsymbol{e}||_2 \qquad \forall\, \boldsymbol{x}^{gt} \in \mathcal{X}, \forall\, ||\boldsymbol{e}||_2 \leq \delta.$$

Definition 2.5. We define the *stability radius* ρ of Ψ as

$$\rho = \sup\{\delta > 0;\ C_{\Psi}^{\delta} \in [0, 1)\}\,. \tag{2.4}$$

Example 2.2. A reconstructor with an infinite stability radius is the following. Given $\delta > 0$, if μ is a probability distribution over $\mathcal{X}$ (for example, μ is the normalized Lebesgue distribution over $\mathcal{X}$), the reconstructor defined as

$$\Psi^{\mathcal{X},\delta}(\boldsymbol{y}^{\delta}) = \int_{\mathcal{X}} \boldsymbol{x}\mu(d\boldsymbol{x}), \qquad \forall \boldsymbol{y}^{\delta} \in \mathcal{Y}^{\delta}$$

is δ-stable independently from the value of $\delta > 0$. Indeed,

$$||\Psi^{\mathcal{X},\delta}(\boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}) - \boldsymbol{x}^{gt}||_2 = \left\| \int_{\mathcal{X}} \boldsymbol{x}\mu(d\boldsymbol{x}) - \boldsymbol{x}^{gt} \right\| \leq \rho(\mathcal{X}),$$

where $\rho(\mathcal{X})$ is the radius of $\mathcal{X}$, defined as $\rho(\mathcal{X}) = \inf\{r > 0 : \mathcal{X} \subseteq \mathcal{B}(\int_{\mathcal{X}} \boldsymbol{x}\mu(d\boldsymbol{x}); r)\}$. As a consequence, the stability constant is infinite regardless δ and $\Psi^{\mathcal{X},\delta}(\boldsymbol{y})$ has accuracy $\rho(\mathcal{X})^{-1}$.

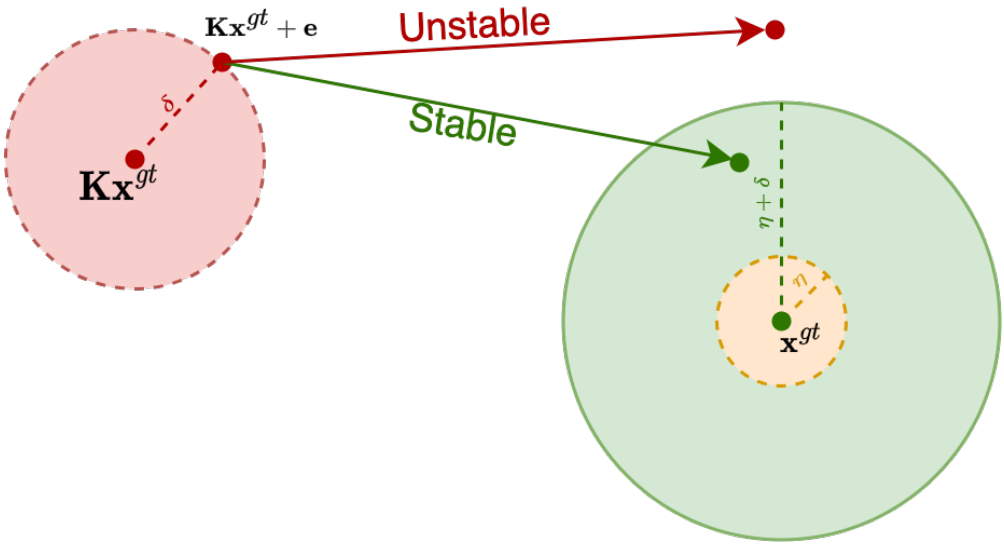


Figure 2.2: Graphical representation of the δ-stability for an η^{-1}-accurate reconstructor.

Example 2.3. The pseudo-inverse reconstructor $\Psi^{\dagger}(\boldsymbol{y})$ in equation (2.2) is unstable for any $\delta > 0$ when $\boldsymbol{K}$ is ill-conditioned. Indeed,

$$||\Psi^{\dagger}(\boldsymbol{K}\boldsymbol{x}^{gt}+\boldsymbol{e})-\boldsymbol{x}^{gt}||_2 = ||(\boldsymbol{K}^T\boldsymbol{K})^{-1}(\boldsymbol{K}^T\boldsymbol{K})\boldsymbol{x}^{gt}+(\boldsymbol{K}^T\boldsymbol{K})^{-1}\boldsymbol{K}^T\boldsymbol{e}-\boldsymbol{x}^{gt}||_2 = ||(\boldsymbol{K}^T\boldsymbol{K})^{-1}\boldsymbol{K}^T\boldsymbol{e}||_2.$$

If $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the Singular Value Decomposition (SVD) of $\boldsymbol{K}$, then

$$(\boldsymbol{K}^T\boldsymbol{K})^{-1}\boldsymbol{K}^T\boldsymbol{e} = (\boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{e} = \boldsymbol{V}\boldsymbol{\Sigma}^{\dagger}\boldsymbol{U}^T\boldsymbol{e} = \sum_{i=1}^{n}\frac{\boldsymbol{u}_i^T\boldsymbol{e}}{\sigma_i}\boldsymbol{v}_i,$$

which implies that $||(\boldsymbol{K}^T\boldsymbol{K})^{-1}\boldsymbol{K}^T\boldsymbol{e}||_2 \gg ||\boldsymbol{e}||_2$ when $\boldsymbol{K}$ has singular values close to zero.

These examples shed light on a possible conflict between accuracy and stability for a given reconstructor Ψ. In the next paragraphs, we study this relationship.

2.1.1 Accuracy vs. stability trade-off

We can derive a relation between accuracy and stability, which becomes particularly interesting when $\boldsymbol{K}$ is ill-conditioned.

Lemma 2.4. *Let $\Psi : \mathbb{R}^m \to \mathbb{R}^n$ be an η^{-1}-accurate reconstructor. Then, for any $\boldsymbol{x}^{gt} \in \mathcal{X}$ and for any $\delta > 0$, $\exists\, \tilde{\boldsymbol{e}} \in \mathbb{R}^m$ with $||\tilde{\boldsymbol{e}}||_2 \leq \delta$ such that*

$$||\Psi(\boldsymbol{K}\boldsymbol{x}^{gt} + \tilde{\boldsymbol{e}}) - \boldsymbol{x}^{gt}||_2 \geq ||\boldsymbol{K}^{\dagger}\tilde{\boldsymbol{e}}||_2 - \eta\,. \tag{2.5}$$

Proof. See [8]. □

Since the corruption $\tilde{\boldsymbol{e}}$ such that the relationship (2.5) holds depends on $\boldsymbol{x}^{gt}$, for any $\boldsymbol{x}^{gt} \in \mathcal{X}$, we will consider the set

$$E(\boldsymbol{x}^{gt}) = \{\boldsymbol{e} \in \mathbb{R}^m; \text{ Equation (2.5) holds for } \boldsymbol{e}, \text{ for some } \delta > 0\}. \tag{2.6}$$

Theorem 2.5 (Trade-off Theorem)**.** *Under the same assumptions of Lemma 2.4 it holds that, for any $\boldsymbol{x}^{gt} \in \mathcal{X}$ and for any $\tilde{\boldsymbol{e}} \in E(\boldsymbol{x}^{gt})$ with $||\tilde{\boldsymbol{e}}||_2 \leq \delta$,*

$$C_{\Psi}^{\delta} \geq \frac{||\boldsymbol{K}^{\dagger}\tilde{\boldsymbol{e}}||_2 - 2\eta}{||\tilde{\boldsymbol{e}}||_2}. \tag{2.7}$$

Proof. See [8]. □

Corollary 2.6. *Given the assumptions of Theorem 2.5, if $\mathcal{X} = \mathbb{R}^n$, there is a constant $C(\boldsymbol{K}) > 0$ which depends only on $\boldsymbol{K}$, such that*

$$\rho \leq \frac{2}{\eta^{-1}C(\boldsymbol{K})}. \tag{2.8}$$

Proof. The proof considers the SVD decomposition of $\boldsymbol{K}$ and, by applying triangular inequalities to the formula in Theorem 2.5, proves the relationship in the statement. The proof also shows an explicit formula for $C(\boldsymbol{K})$ which is equal to $\frac{1-\sigma_n}{\sigma_n}$. For the complete proof see [8]. □

The relation (2.8) between the stability radius ρ and the accuracy η^{-1} suggests that there exists a trade-off between accuracy and stability, showing that a very accurate reconstructor is unstable for noise corruption larger than $\frac{2}{\eta^{-1}C(\boldsymbol{K})}$. We remark that for ill-conditioned problems $C(\boldsymbol{K}) = \frac{1-\sigma_n}{\sigma_n}$ can be very large, making the radius potentially very small.

Similarly, Theorem 2.5 shows that a reconstructor Ψ can be δ-stable only if its accuracy is bounded.

Corollary 2.7. *Given the assumptions of Theorem 2.5, there exists $\bar{\eta}(\boldsymbol{K}, \delta, \mathcal{X}) \in \mathbb{R} \cup \{+\infty\}$, such that any reconstructor Ψ with accuracy $\eta^{-1} \geq \bar{\eta}(\boldsymbol{K}, \delta, \mathcal{X})^{-1}$ is δ-unstable, i.e. $C_{\Psi}^{\delta} \geq 1$. Moreover, if $\mathcal{X} = \mathbb{R}^n$ and $\eta^{-1} \geq \frac{2}{C(\boldsymbol{K})\delta}$, where $C(\boldsymbol{K}) = \frac{1-\sigma_n}{\sigma_n}$, then Ψ is δ-unstable.*

Proof. See [8]. □

2.1.2 A sufficient condition for stability

Whenever the reconstructor is (locally) Lipschitz continuous, we can also derive conditions assessing stability. First of all, we recall the definition of locally Lipschitz continuous reconstructors.

Definition 2.6. Given $\mathcal{Y} \subseteq \mathbb{R}^m$ and $\delta > 0$, we define the δ-Lipschitz (also called local Lipschitz) constant of Ψ over $\mathcal{Y}$ as:

$$L^\delta(\Psi, \mathcal{Y}) = \sup_{\substack{\boldsymbol{y}\in\mathcal{Y}, \boldsymbol{z}\in\mathbb{R}^m \\ ||\boldsymbol{z}-\boldsymbol{y}||_2 \leq \delta}} \frac{||\Psi(\boldsymbol{z}) - \Psi(\boldsymbol{y})||_2}{||\boldsymbol{z} - \boldsymbol{y}||_2}. \tag{2.9}$$

If $L^\delta(\Psi, \mathcal{Y}) < \infty$ for some $\delta > 0$, then Ψ is said to be locally Lipschitz continuous.

Focusing on our problem (2.1), we are interested in the cases where $\mathcal{Y} = Rg(\boldsymbol{K}, \mathcal{X})$. In this case, $\boldsymbol{y} \in \mathcal{Y}$ implies that $\exists\, \boldsymbol{x}^{gt} \in \mathcal{X}$ such that $\boldsymbol{y} = \boldsymbol{K}\boldsymbol{x}^{gt}$ and each $\boldsymbol{z} \in \mathbb{R}^m$ with $||\boldsymbol{z} - \boldsymbol{y}||_2 \leq \delta$ can be characterized by $\boldsymbol{z} = \boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}$ for some $\boldsymbol{e} \in \mathbb{R}^m$ with $||\boldsymbol{e}||_2 \leq \delta$. Thus, the definition of $L^\delta(\Psi, \mathcal{Y})$ can be rewritten as:

$$L^\delta(\Psi, \mathcal{Y}) = \sup_{\substack{\boldsymbol{x}^{gt}\in\mathcal{X} \\ ||\boldsymbol{e}||_2 \leq \delta}} \frac{||\Psi(\boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}) - \Psi(\boldsymbol{K}\boldsymbol{x}^{gt})||_2}{||\boldsymbol{e}||_2}.$$

From now on, we always assume $\mathcal{Y} = Rg(\boldsymbol{K}, \mathcal{X})$.

Proposition 2.8. *Let $\Psi : \mathbb{R}^m \to \mathbb{R}^n$ be a reconstructor with accuracy $\eta^{-1} > 0$ and local Lipschitz constant $L^\delta(\Psi, \mathcal{Y})$. Then, given $||\boldsymbol{e}||_2 \leq \delta$, it holds:*

$$||\Psi(\boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}) - \boldsymbol{x}^{gt}||_2 \leq \eta + L^\delta(\Psi, \mathcal{Y})||\boldsymbol{e}||_2\,. \tag{2.10}$$

Proof. See [8]. □

Corollary 2.9. *Under the assumptions of Proposition 2.8,*

$$C^\delta_\Psi \leq L^\delta(\Psi, \mathcal{Y}). \tag{2.11}$$

Proof. The Corollary follows from the Proposition 2.8 by the minimality of C^δ_Ψ. A complete proof can be found in [8]. □

Corollary 2.9 proves that Ψ is δ-stable if $L^\delta(\Psi, \mathcal{Y}) < 1$.

2.2 Neural Networks as reconstructors

In this Section, we analyze neural networks as particular reconstructors for problem (2.1).

Consider a sequence of approximators $\{\Psi_\Theta\}$, depending on a vector of parameters Θ, approximating a reconstructor Ψ. In the following Theorem, we prove that the stability of Ψ_Θ is strongly related to the stability of Ψ.

Theorem 2.10 (Approximation Theorem for Reconstructors)**.** *Let Ψ be an η^{-1}-accurate reconstructor and let $\{\Psi_\Theta\}_{\Theta\in\mathbb{R}^s}$ be a set of reconstructors with accuracy η_Θ^{-1} for any Θ. We define, for any $\Theta \in \mathbb{R}^s$:*

$$\Delta(\Theta) := \sup_{\boldsymbol{x}^{gt}\in\mathcal{X}} ||\Psi_\Theta(\boldsymbol{K}\boldsymbol{x}^{gt}) - \Psi(\boldsymbol{K}\boldsymbol{x}^{gt})||_2,$$

and

$$\Delta_\delta(\Theta) := \sup_{\boldsymbol{y}^\delta\in\mathcal{Y}^\delta} ||\Psi_\Theta(\boldsymbol{y}^\delta) - \Psi(\boldsymbol{y}^\delta)||_2.$$

If $\Delta(\Theta) \to 0$ when $\Theta \to \Theta^*$, then:

$$\lim_{\Delta(\Theta)\to 0} \eta_\Theta = \eta. \tag{2.12}$$

Moreover, if $\Delta_\delta(\Theta) \to 0$ when $\Theta \to \Theta^*_\delta$, then:

$$\lim_{\Delta_\delta(\Theta)\to 0} C^\delta_{\Psi_\Theta} = C^\delta_\Psi. \tag{2.13}$$

Proof. See [8]. □

Corollary 2.11. *For any $\Theta \in \mathbb{R}^s$, it holds:*

$$\eta_\Theta \le \eta + \Delta(\Theta). \tag{2.14}$$

Proof. See [8]. □

Note that $\Delta(\Theta)$ and $\Delta_\delta(\Theta)$ are, in general, not independent, as proved in the following proposition.

Proposition 2.12. *For any $\delta > 0$, let $\Delta(\Theta)$ and $\Delta_\delta(\Theta)$ be the quantities defined in Theorem 2.10. Then:*

$$\Delta(\Theta) \le \Delta_\delta(\Theta).$$

Proof. See [8]. □

An insight on the stability properties of Ψ_Θ can be obtained by the following proposition.

Proposition 2.13. *Let Ψ_Θ be a reconstructor parameterized by $\Theta \in \mathbb{R}^s$, approximating a reconstructor Ψ with error $\Delta(\Theta) > 0$. Let η_Θ^{-1} and η^{-1} be the accuracy of Ψ_Θ and Ψ, respectively. If:*

$$\Delta(\Theta) \le \bar{\eta}(\boldsymbol{K}, \delta, \mathcal{X}) - \eta \tag{2.15}$$

for a fixed $\delta > 0$, where $\bar{\eta}(\boldsymbol{K}, \delta, \mathcal{X})$ is the constant defined in Corollary 2.7, then $C^\delta_{\Psi_\Theta} \ge 1$.

Proof. See [8]. □

In the following, we will consider as family $\{\Psi_\Theta\}_{\Theta\in\mathbb{R}^s}$ the neural networks, with the notation introduced in Definition 1.1.

Given $\mathcal{S}\subseteq\mathcal{X}$, consider the dataset $\mathbb{D}=\{(\boldsymbol{y}_i^\delta,\boldsymbol{x}_i^{gt});\boldsymbol{x}_i^{gt}\in\mathcal{S}\}_{i=1}^N$ of images such that, for any $i=1,\dots,N$, $\boldsymbol{y}_i^\delta=\boldsymbol{K}\boldsymbol{x}_i^{gt}+\boldsymbol{e}_i$, $\boldsymbol{e}_i\sim\mathcal{N}(\boldsymbol{0},\sigma^2\boldsymbol{I})$. Recall from Section 1.3.3 that training a neural network to solve the inverse problem (2.1) means finding the reconstructor $\Psi_\Theta\in\mathcal{F}_\Theta^{\mathcal{A}}$ solving the minimization problem:

$$\min_{\Psi_\Theta\in\mathcal{F}_\Theta^{\mathcal{A}}}\frac{1}{N}\sum_{i=1}^N\ell(\Psi_\Theta(\boldsymbol{y}_i^\delta),\boldsymbol{x}_i^{gt}),\tag{2.16}$$

where $\delta\geq 0$ and $\ell:\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}_+$ is the loss function. Whenever we choose as loss function the Mean Squared Error (MSE) on noiseless data ($\delta=0$), (2.16) corresponds to:

$$\min_{\Psi_\Theta\in\mathcal{F}_\Theta^{\mathcal{A}}}\sum_{i=1}^N||\Psi_\Theta(\boldsymbol{y}_i)-\boldsymbol{x}_i^{gt}||_2^2=\min_{\Psi_\Theta\in\mathcal{F}_\Theta^{\mathcal{A}}}\sum_{i=1}^N||\Psi_\Theta(\boldsymbol{K}\boldsymbol{x}_i^{gt})-\Psi^\dagger(\boldsymbol{K}\boldsymbol{x}_i^{gt})||_2^2,\tag{2.17}$$

which results in the minimization of $\Delta(\Theta)$ as introduced in Theorem 2.10 with $\Psi=\Psi^\dagger$. We will name this family as NN in the following.

We observe that when $\boldsymbol{K}$ is ill-conditioned, the value $\bar{\eta}(\boldsymbol{K},\delta,\mathcal{X})$ defined in Corollary 2.7 is large. This becomes particularly apparent when $\mathcal{X}=\mathbb{R}^n$, as under these circumstances, $\bar{\eta}(\boldsymbol{K},\delta,\mathcal{X})$ is bounded below by a quantity depending on $C(\boldsymbol{K})=\frac{1-\sigma_n}{\sigma_n}$. Additionally, the value of $\Delta(\Theta^*)$ derived from NN training likely meets the established inequality in Proposition 2.13, which leads to instability. This confirms that effective neural network training can produce a very accurate but unstable reconstructor Ψ_Θ.

Recently, more than one approach has been proposed to soften this instability. One of them is to modify the loss function ℓ to enforce consistency, as developed in [86, 87]. The idea is to define Ψ_Θ such as

$$\Psi_\Theta(\boldsymbol{y}^\delta)\in\operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^n}\frac{1}{2}||\boldsymbol{K}\boldsymbol{x}-\boldsymbol{y}^\delta||_2^2.\tag{2.18}$$

However, it has been proved that this approach does not fully solve instabilities [72].

Another common approach to reduce instabilities in neural networks is adversarial training [88], which can be implemented by data augmentation [89, 90]. The idea is to use equivariant transformations of $\boldsymbol{K}$ to generate new data. However, for ill-conditioned matrices, enlarging the dataset potentially increases instability [72].

A third approach, called noise injection, consists of adding noise to the input of the network during training. It has been proved in [91] that this is equivalent to adding a Tikhonov regularizing term to the loss function. Even if, as explained in [92], this technique improves the stability of the resulting network, it is however not clear to which extent the accuracy of the resulting model is affected by noise injection and, moreover, how much noise should be added to any input to maximize the trade-off between stability and accuracy.

In the following, we propose new approaches to stabilize accurate neural networks as solvers of (2.1).

2.2.1 Better conditioning implies better reconstructors: the ReNN approach

To introduce our proposal, we first consider Tikhonov method, already introduced in Section 1.3.2 and reported here to relate it with the reconstructor theory, representing a stable reconstructor [19, 33], defined as:

$$\Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}^\delta) = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\arg\min}\, \frac{1}{2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta||_2^2 + \frac{\lambda}{2}||\boldsymbol{L}\boldsymbol{x}||_2^2, \tag{2.19}$$

where $\lambda > 0$ is the regularization parameter and $\boldsymbol{L} \in \mathbb{R}^{d\times n}$ is a matrix such that $\ker(\boldsymbol{K}) \cap \ker(\boldsymbol{L}) = \{\boldsymbol{0}\}$. $\boldsymbol{L}$ is usually chosen to be the identity or the forward-difference operator. We can prove the following proposition.

Proposition 2.14. *Let $\delta > 0$ and $\boldsymbol{L} \in \mathbb{R}^{d\times n}$. Then $\exists\, \lambda > 0$ such that*

$$C^\delta_{\Psi^{\lambda,\boldsymbol{L}}} \leq L^\delta(\Psi^{\lambda,\boldsymbol{L}}, \mathcal{Y}) < 1. \tag{2.20}$$

Proof. See [8]. □

Now, we can improve neural networks stability by re-defining their training, forcing the convergence of Ψ_Θ towards Tikhonov regularized reconstructor $\Psi^{\lambda,\boldsymbol{L}}$, which is δ-stable for a suitable value of λ. To this aim, we train the neural network by computing:

$$\min_{\Psi_\Theta\in\mathcal{F}^{\mathcal{A}}_\Theta} \sum_{i=1}^{N} ||\Psi_\Theta(\boldsymbol{y}_i^\delta) - \Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}_i^\delta)||_2^2, \tag{2.21}$$

which corresponds to the minimization of $\Delta_\delta(\Theta)$ in Theorem 2.10. We will refer to such a network as the Regularized Neural Network (ReNN) and we will indicate it as $\Psi^{\lambda,\boldsymbol{L}}_\Theta$.

We underline that ReNN does not require any ground-truth solution $\boldsymbol{x}^{gt}$ since the target is computed from the corrupted datum $\boldsymbol{y}^\delta$ via the Tikhonov-regularized reconstructor. Furthermore, in the training of ReNN, noise is present not solely to the input of the neural network model, as is the case with NN, but also to the input of the Tikhonov-regularized reconstructor, which is responsible for generating the target. In the following, we consider for simplicity the case $\mathcal{X} = \mathbb{R}^n$, but similar results hold for a general $\mathcal{X} \subset \mathbb{R}^n$.

Why does ReNN exhibit greater stability than NN? Starting from inequality (2.15) it is easy to notice that (2.21) corresponds to the minimization of $\Delta_\delta(\Theta)$ in Theorem 2.10. Moreover, by Theorem 2.12, if $\Delta_\delta(\Theta)$ is small, as it is common when Ψ_Θ is a neural network, then $\Delta(\Theta) \in [0, \Delta_\delta(\Theta)]$ is also small. Regarding the right hand side $\bar{\eta}(\boldsymbol{K}, \delta, \mathcal{X}) - \eta$

of Equation (2.15), it is noted that in this instance $\eta = \eta(\lambda)$ and $\eta(\lambda) \to \infty$ for $\lambda \to \infty$. Consequently, for sufficiently large values of λ, it is probable that ReNN does not fulfill the inequality (2.15) that would lead to instability. Moreover, minimizing $\Delta_\delta(\Theta)$ is crucial for enforcing the method's stability, as proven by Theorem 2.10, where we have shown that in our hypothesis the stability constant $C^\delta_{\Psi^{\lambda,\boldsymbol{L}}_\Theta} < 1$ for sufficiently small $\Delta_\delta(\Theta)$. Hence, effective training of ReNN should produce an accurate and stable reconstructor. The pseudocode to compute $\Psi^{\lambda,\boldsymbol{L}}_\Theta$ is given in Algorithm 4.

Algorithm 4 Regularized Neural Network (ReNN)

input a collection $\{\boldsymbol{x}_i^{gt}\}_{i=1}^N \subseteq \mathcal{X}$ of data points, $\delta > 0$, $\boldsymbol{K} \in \mathbb{R}^{m\times n}$ and $\Psi^{\lambda,\boldsymbol{L}}$, $\mathcal{A}$
for $i \leftarrow 1 : N$ **do**
  Sample $\boldsymbol{e}_i \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$ such that $||\boldsymbol{e}_i||_2 \leq \delta$
  Compute $\boldsymbol{y}_i^\delta \leftarrow \boldsymbol{K}\boldsymbol{x}_i^{gt} + \boldsymbol{e}_i$
  Compute $\Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}_i^\delta)$
  Append $(\boldsymbol{y}_i^\delta, \Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}_i^\delta))$ to $\mathbb{D}^{\lambda,\boldsymbol{L}}$
end for
Find

$$\arg\min_{\Psi_\Theta \in \mathcal{F}^{\mathcal{A}}_\Theta} \sum_{i=1}^N ||\Psi_\Theta(\boldsymbol{y}_i^\delta) - \Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}_i^\delta)||_2^2.$$

return a trained ReNN Ψ_Θ

2.3 Stabilizers

In this Section, we propose a further stabilization of neural networks by defining the concept of stabilizer.

Definition 2.7. A continuous functions $\phi : \mathbb{R}^m \to \mathbb{R}^t$ is a δ-stabilizer of a reconstructor $\Psi : \mathbb{R}^m \to \mathbb{R}^n$ if:

1. $\forall\, \boldsymbol{e} \in \mathbb{R}^m$ with $||\boldsymbol{e}||_2 \leq \delta$, $\exists\, C^\delta_\phi \in [0, 1)$ and $\exists\, \boldsymbol{e}' \in \mathbb{R}^n$ with $||\boldsymbol{e}'||_2 = C^\delta_\phi||\boldsymbol{e}||_2$ such that

$$\phi(\boldsymbol{K}\boldsymbol{x} + \boldsymbol{e}) = \phi(\boldsymbol{K}\boldsymbol{x}) + \boldsymbol{e}'. \tag{2.22}$$

2. $\exists\, \gamma : \mathbb{R}^t \to \mathbb{R}^n$ such that $\Psi = \gamma \circ \phi$.

In this case, the reconstructor Ψ is said to be δ-stabilized. The smallest constant C^δ_ϕ for which the definition holds is the stability constant of the stabilizer ϕ.

Note that, in the definition of δ-stabilizer, we only require a stability condition for ϕ in the first item. Interestingly, given a δ-stabilized reconstructor $\Psi = \gamma \circ \phi$, we can estimate the δ-stability constant C_Ψ^δ of Ψ by the constant C_ϕ^δ and the local Lipschitz constant of γ as proved in the following proposition.

Proposition 2.15. *Let $\Psi : \mathbb{R}^m \to \mathbb{R}^n$ be an δ-stabilized reconstructor, $\Psi = \gamma \circ \phi$. If C_ϕ^δ is the constant defined in (2.22), $L^\delta(\gamma, \mathcal{T})$ is the local Lipschitz constant of γ with $\mathcal{T} = \phi(\mathcal{Y})$, then*

$$C_\Psi^\delta \leq L^\delta(\gamma, \mathcal{T}) C_\phi^\delta.$$

Proof. See [8]. □

Proposition 2.15 implies the following result:

Theorem 2.16. *For any $\delta > 0$, $\eta_1, \eta_2 > 0$, let $\Psi_1 = \gamma_1 \circ \phi_1$ be an η_1^{-1}-accurate δ-stabilized reconstructor, and let Ψ_2 be an η_2^{-1}-accurate reconstructor. If:*

$$C_{\phi_1}^\delta \in \left[0, \frac{C_{\Psi_2}^\delta}{L^\delta(\gamma_1, \mathcal{T})}\right], \tag{2.23}$$

then:

$$C_{\Psi_1}^\delta \leq C_{\Psi_2}^\delta.$$

Proof. See [8]. □

This Theorem yields interesting consequences for the special case where Ψ_1 and Ψ_2 share the same accuracy. For instance, when both $\Psi_1 = \gamma_1 \circ \phi_1$ and Ψ_2 are η^{-1}-accurate, if Equation (2.23) holds, the Theorem suggests that Ψ_1 is preferable to Ψ_2, as Ψ_1 is more stable than Ψ_2. In addition, we can state the following result, whose proof is trivial.

Corollary 2.17. *Let $\Psi_1 = \gamma_1 \circ \phi_1$ and $\Psi_2 = \gamma_2 \circ \phi_2$ be δ-stabilized reconstructors. If Equation (2.23) holds, then $C_{\phi_1}^\delta \leq C_{\phi_2}^\delta$.*

In the next Proposition, we show that the accuracy of any δ-stabilized reconstructor strongly depends on how far is ϕ from an injective function. Indeed, it holds:

Proposition 2.18. *Let $\phi : \mathbb{R}^m \to \mathbb{R}^t$ be an δ-stabilizer for an η^{-1}-accurate reconstructor $\Psi = \gamma \circ \phi$. Let*

$$\sigma(\phi) := \sup\{||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2; \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}, \phi(\boldsymbol{K}\boldsymbol{x}_1) = \phi(\boldsymbol{K}\boldsymbol{x}_2)\}. \tag{2.24}$$

Then

$$\eta^{-1} \leq \frac{2}{\sigma(\phi)}. \tag{2.25}$$

Proof. See [8]. □

Note that, if ϕ is the constant reconstructor, $\sigma(\phi) = \infty$, which implies that for any γ, the accuracy of $\Psi = \gamma \circ \phi$ will be zero. Similarly, if $\sigma(\phi) < \infty$, then for any $\eta^{-1} \in (0, \frac{2}{\sigma(\phi)}]$, there exists $\Psi = \gamma \circ \phi$ that is η^{-1}-accurate. A flexible way to define such a γ, which is mathematically complex to describe, is to use a neural network.

2.3.1 Iterative algorithms as stabilizers for neural networks

Consider the variational regularized reconstructor $\Psi^{\lambda,\boldsymbol{L}}$ as in (2.19). Suppose we compute a sequence of functions $\{\phi_k\}_{k\in\mathbb{N}}$ approximating $\Psi^{\lambda,\boldsymbol{L}}$, i.e. :

$$\lim_{k\to\infty} ||\phi_k - \Psi^{\lambda,\boldsymbol{L}}||_{L^2} = 0.$$

A simple way to get it is to consider a convergent iterative algorithm for the solution of (2.19):

$$\begin{cases} \boldsymbol{x}^{(0)} \in \mathbb{R}^n \\ \boldsymbol{x}^{(k+1)} = \mathcal{T}_k(\boldsymbol{x}^{(k)}, \boldsymbol{y}^\delta). \end{cases}$$

Then, for any $k \in \mathbb{N}$ we can define

$$\phi_k(\cdot, \boldsymbol{y}^\delta) = \bigcirc_{i=1}^{k} \mathcal{T}_i(\cdot, \boldsymbol{y}^\delta), \tag{2.26}$$

where $\bigcirc$ is the composition operator. Clearly, as a consequence of the convergence of the iterates $\mathcal{T}_k$, $\{\phi_k\}_{k\in\mathbb{N}}$ is a sequence of functions approximating $\Psi^{\lambda,\boldsymbol{L}}$. We can now define an δ-stabilizer by fixing $K \in \mathbb{N}$ and computing ϕ_k for some $k \geq K$.

Proposition 2.19. *Given a reconstructor $\Psi : \mathbb{R}^m \to \mathbb{R}^n$ with local Lipschitz constant $L^\delta(\Psi, \mathcal{Y}) < 1$ and a sequence of functions $\{\phi_k\}_{k\in\mathbb{N}}$ approximating Ψ, there exists $K \in \mathbb{N}$ such that for any $k \geq K$, ϕ_k is an δ-stabilizer.*

Proof. See [8]. □

Consider the case study of the Tikhonov regularized reconstructor $\Psi^{\lambda,\boldsymbol{L}}$. We already proved in Proposition 2.14 that $L^\delta(\Psi, \mathcal{Y}) < 1$ for some $\lambda > 0$. Moreover, from (2.19) we know that $\Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}^\delta)$ is the solution of the normal equations system

$$(\boldsymbol{K}^T\boldsymbol{K} + \lambda\boldsymbol{L}^T\boldsymbol{L})\boldsymbol{x} = \boldsymbol{K}^T\boldsymbol{y}^\delta. \tag{2.27}$$

By Proposition 2.19, the functions ϕ_k chosen as the iterates of an iterative method for the solution of (2.27), such as the Conjugate Gradient for Least Squares (CGLS) [16], define a stabilizer for a neural network Ψ_Θ. We can join either NN or ReNN with this stabilizer, obtaining StNN and StReNN, respectively, whose implementation is reported in Algorithms 5 and 6. Figure 2.3 schematically represents all the proposed approaches.

2.4 Experimental Setup

To assess the theoretical issues proposed, we perform three experiments, denoted as A, B, and C in the following. For each experiment, we test two aspects: first of all, we compute an estimation of the accuracy and the δ-stability constant for some fixed $\delta > 0$ in the three

Algorithm 5 Stabilized Neural Network (StNN)

input a collection $\mathbb{D} = \{(\boldsymbol{y}_i^\delta, \boldsymbol{x}_i^{gt})\}_{i=1}^N$ of data points, a convergent algorithm $\mathcal{T}_k$ for $\Psi^{\lambda,\boldsymbol{L}}$, an integer $k \in \mathbb{N}$ such that ϕ_k is an δ-stabilizer, $\mathcal{A}$

Find

$$\gamma_\Theta = \underset{\gamma_\Theta \in \mathcal{F}_\Theta^{\mathcal{A}}}{\arg\min} \sum_{i=1}^{N} ||\gamma_\Theta(\phi_k(\boldsymbol{y}_i^\delta)) - \boldsymbol{x}_i^{gt}||_2^2.$$

Define $\Psi_\Theta = \gamma_\Theta \circ \phi_k$

return a trained StNN Ψ_Θ

Algorithm 6 Stabilized Regularized Neural Network (StReNN)

input a collection $\mathbb{D} = \{(\boldsymbol{y}_i^\delta, \boldsymbol{x}_i^{gt})\}_{i=1}^N$ of data points, a parameter $\delta > 0$, a convergence algorithm $\mathcal{T}_k$ for $\Psi^{\lambda,\boldsymbol{L}}$, an integer $k \in \mathbb{N}$ such that ϕ_k is an δ-stabilizer, $\mathcal{A}$

for $i \leftarrow 1 : N$ **do**

Sample $e_i \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ such that $||\boldsymbol{e}_i||_2 \leq \delta$

Compute $\boldsymbol{y}_i^\delta \leftarrow \boldsymbol{K}\boldsymbol{x}_i^{gt} + \boldsymbol{e}_i$

Compute $\Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}_i^\delta)$

Append $(\boldsymbol{y}_i^\delta, \Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}_i^\delta))$ to $\mathbb{D}^{\lambda,\boldsymbol{L}}$

end for

Find

$$\gamma_\Theta = \underset{\gamma_\Theta \in \mathcal{F}_\Theta^{\mathcal{A}}}{\arg\min} \sum_{i=1}^{N} ||\gamma_\Theta(\phi_k(\boldsymbol{y}_i^\delta)) - \boldsymbol{x}_i^{\lambda,\boldsymbol{L}}||_2^2.$$

Define $\Psi_\Theta = \gamma_\Theta \circ \phi_k$

return a trained StReNN Ψ_Θ

proposed frameworks, with the intent of verifying their ability to balance the accuracy-stability trade-off. To this end, we define the empirical accuracy $\hat{\eta}^{-1}$ and the empirical stability constant $\hat{C}_\Psi^\delta$, as:

$$\hat{\eta} = \sup_{\boldsymbol{x}^{gt} \in \mathcal{S}} ||\Psi(\boldsymbol{K}\boldsymbol{x}^{gt}) - \boldsymbol{x}^{gt}||_2, \tag{2.28}$$

and

$$\hat{C}_\Psi^\delta = \sup_{\boldsymbol{x}^{gt} \in \mathcal{S}} \frac{||\Psi(\boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}) - \boldsymbol{x}^{gt}||_2 - \hat{\eta}}{||\boldsymbol{e}||_2}, \tag{2.29}$$

where $\mathcal{S} \subseteq \mathcal{X}$ is the test set and $\boldsymbol{e}$ is a noise realization from $\mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ with $||\boldsymbol{e}||_2 \leq \delta$ (different realization for any datum $\boldsymbol{x}^{gt} \in \mathcal{S}$).

To enhance the stochasticity of our tests, we repeated $T = 20$ times the experiments on the test sets, with different realizations of noise. In the following, we report the maximum

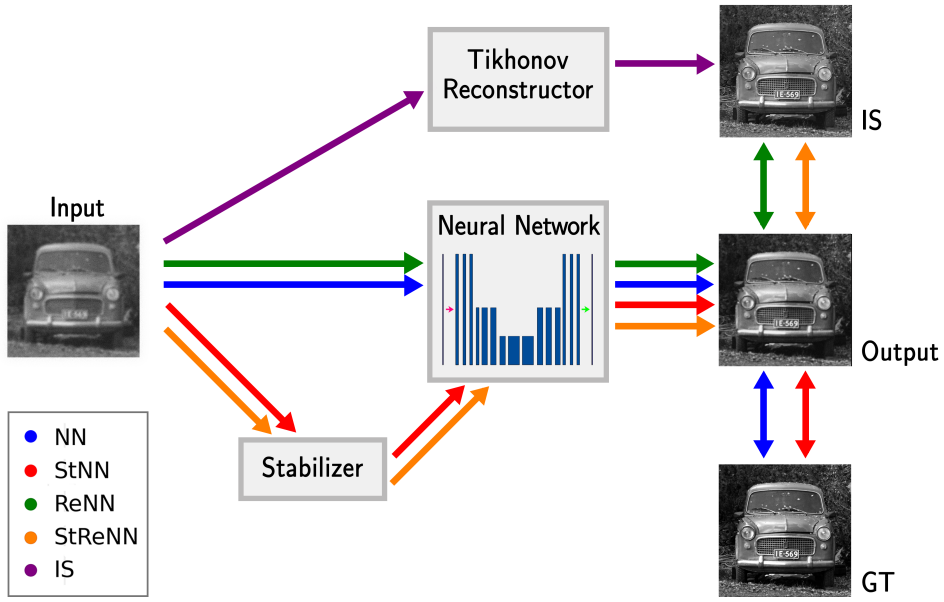


Figure 2.3: A schematic representation of the proposed methods.

value of the computed parameters $\hat{\eta}$ and $\hat{C}_{\Psi}^{\delta}$ over the T experiments.

In this phase, we perform all the tests on the end-to-end U-Net architecture introduced in Section 1.3.3, as depicted in Figure 2.3, with the intent of isolating the improvement given by the framework to the network architecture. The stabilizer applied to StNN and StReNN is obtained with $k = 3$ iterations of the CGLS algorithm applied on (2.19) and is indicated as ϕ_k in the following. In the second part, we focus on the Stabilized Neural Networks introduced in Section 2.3. In particular, we quantitatively and qualitatively compare the results of the three different architectures introduced in Section 1.3.3, i.e. the 3L-SSNet, the U-Net and the NAFNet, with two different stabilizers, namely the Tikhonov Stabilized Neural Network (StNN) already considered and a newly introduced Filtered Neural Network (FiNN). The latter is based on the intuition that a pre-processing step should reduce the noise present in the input data. As a consequence, we consider as a pre-processing step the application of a Gaussian denoising filter, which is a low-pass filter that reduces the impact of noise on the high frequencies [93]. Thus, the resulting pre-processed image is a low-frequency version of $\boldsymbol{y}^{\delta}$ and the neural network $\Psi_{\Theta} \in \mathcal{F}_{\Theta}^{A}$ has to recover the high frequencies corresponding to the image details. We will indicate this operator as $\phi_{\mathcal{G}}$ in the following.

As a test case, we consider the image deblurring inverse problem introduced in Section 1.2.1, where $\boldsymbol{K}$ is the $256^2 \times 256^2$ Gaussian blur matrix described in Section 1.2.1, with $s = 11$ and $\sigma^2 = 1.3$. All the tests have been performed on the GoPro dataset, as described in Section 1.4.1. From it, we built two datasets required from the experiments, as depicted

in Algorithms 4, 5 and 6:

- $\mathbb{D} = \{(\boldsymbol{y}_i^\delta, \boldsymbol{x}_i^{gt}); \boldsymbol{y}_i^\delta = \boldsymbol{K}\boldsymbol{x}_i^{gt} + \boldsymbol{e}_i\}_{i=1}^N$ contains the images representing the blurred and noisy version of $\boldsymbol{x}_i^{gt}$, by sampling $\boldsymbol{e}_i \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$.

- $\mathbb{D}^{\lambda,\boldsymbol{L}} = \{(\boldsymbol{y}_i^\delta, \Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}_i^\delta))\}_{i=1}^N$ is constituted by the images obtained by applying the Tikhonov reconstructor defined in (2.19) to the corrupted data, using $\boldsymbol{L} = \boldsymbol{I}$. In particular, we choose λ heuristically and we computed $\Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{y}^\delta)$ by using the CGLS algorithm [16] to solve the normal equations of (2.19).

2.4.1 Experiment A

In this first experiment, we trained the networks to solve the noiseless task $\boldsymbol{y}_i = \boldsymbol{K}\boldsymbol{x}_i^{gt}$ to fit the theoretical assumptions. Thus, we trained the neural networks of NN and StNN approaches with the input $\mathbb{D}$, ReNN and StReNN with the input $\mathbb{D}^{\lambda,\boldsymbol{L}}$.

We remark that by minimizing the MSE either on $\mathbb{D}$ or $\mathbb{D}^{\lambda,\boldsymbol{L}}$ we are minimizing $\Delta(\Theta)$ as defined in Theorem 2.10. In order to test the stability of our frameworks with respect to unseen noise on the data, we tested on noisy images $\boldsymbol{y}_i^\delta = \boldsymbol{y}_i + \boldsymbol{e}_i$ where $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \overline{\sigma}^2\boldsymbol{I})$. In the following, we will say that a test is *in-domain* if the data is corrupted with the same amount of noise as in the training, *out-of-domain* if the amount of noise corrupting $\boldsymbol{y}^\delta$ is different to those in the training.

2.4.2 Experiment B

Among the stabilizing solutions listed in Section 2.2, in this experiment, we tested noise injection, which is considered the most reliable. Hence we modified the training of experiment A by adding Gaussian noise with variance $\sigma^I = 0.025$ to the input of each network. We observe that concerning StNN and StReNN, the noise has been added after stabilizing the input. In this experiment, we have tested on noisy images with $\overline{\sigma} = 0.075$ and $\overline{\sigma} = 0.105$.

2.4.3 Experiment C

The last test aims at experimentally checking the results proved in the second part of Theorem 2.10, where in equation (2.13) we state that, by reducing $\Delta_\delta(\Theta)$, the stability constant $C^\delta_{\Psi_\Theta}$ converges to C^δ_Ψ. In this experiment, we show that if we choose Ψ as the Tikhonov reconstructor, we can modify the training of ReNN and StReNN methods to minimize $\Delta_\delta(\Theta)$ in place of $\Delta(\Theta)$, aiming at dramatically improving their stability. To do that, we modified the input $\mathbb{D}^{\lambda,\boldsymbol{L}}$ of the training for ReNN and StReNN by introducing noise in the target, thus obtaining $\hat{\mathbb{D}}^{\lambda,\boldsymbol{L}} = \{(\boldsymbol{K}\boldsymbol{x}_i^{gt} + \hat{\boldsymbol{e}}, \Psi^{\lambda,\boldsymbol{L}}(\boldsymbol{K}\boldsymbol{x}_i^{gt} + \hat{\boldsymbol{e}})\}_{i=1}^N$ where $\hat{\boldsymbol{e}} \sim \mathcal{N}(\boldsymbol{0}, \hat{\sigma}^2\boldsymbol{I})$ with $\hat{\sigma} = 0.025$.

2.5 Numerical Results

In this Section, we present the results obtained in our deblurring experiments described in Section 2.4.

2.5.1 Comparison of ReNN, StNN and StReNN

In the following, we compare the accuracy and the stability of the NN approach with the ReNN, StNN and StReNN frameworks discussed in the previous Sections. We remark that we consider the U-Net architecture for all these tests.

Results of Experiment A Table 2.1 reports the mean values of the empirical accuracy $\hat{\eta}^{-1}$ and empirical stability constant $\hat{C}_{\Psi}^{\delta}$, computed on the test set, with the setup described in Section 2.4.1. To graphically visualize the trade-off between accuracy and stability, we plot in Figure 2.4a $\hat{C}_{\Psi}^{\delta}$ versus $\hat{\eta}^{-1}$. As expected, NN has excellent accuracy but it lacks stability; on the contrary, ReNN appears as a good trade-off between accuracy and stability, since it shows a slightly higher value of $\hat{\eta}^{-1}$ with respect to NN, whereas its constant $\hat{C}_{\Psi}^{\delta}$ is dramatically lower (from a value greater than 32 in NN to about 10 in ReNN). With the introduction of a stabilizer, we observe that StNN and StReNN are δ-stable since $\hat{C}_{\Psi}^{\delta} < 1$. Finally, by comparing the deep learning-based frameworks with the variational reconstructor $\Psi_K^{\lambda,\boldsymbol{L}}$, denoted as IS in the following, we observe that the proposed StNN and StReNN improve the IS accuracy while preserving the stability.

In Figure 2.4b we plot the reconstruction error:

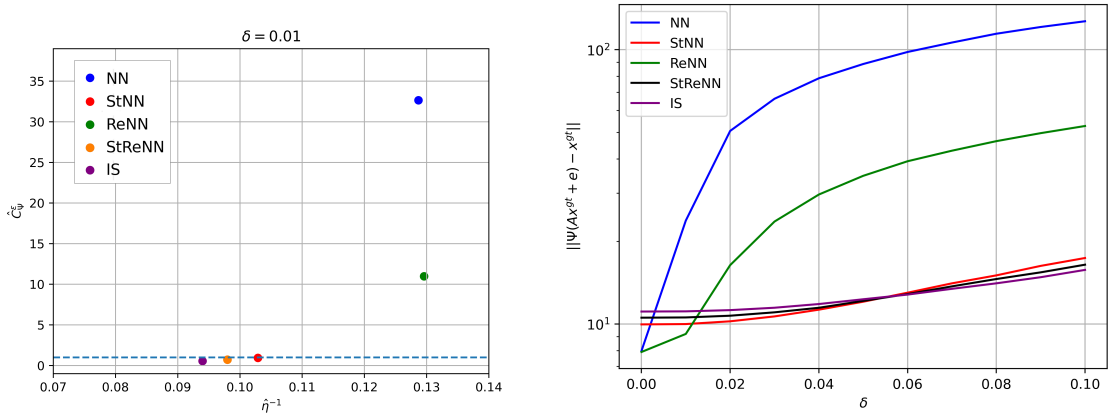
$$\mathcal{E}(\Psi; \boldsymbol{x}^{gt}) = ||\Psi(\boldsymbol{y}^{\delta}) - \boldsymbol{x}^{gt}||_2 \tag{2.30}$$

when the value of $\overline{\sigma}$ varies in the interval $[0, 0.1]$ for all the considered reconstructors. We observe that the errors of NN and ReNN methods increase very rapidly: for values of $\overline{\sigma}$ near zero they have slightly lower reconstruction error, but StNN and StReNN perform much better as $\overline{\sigma}$ increases. Moreover, we emphasize that the stabilizer introduced in StNN and StReNN is very effective since the curves have a gentle slope with increasing $\overline{\sigma}$. Finally, to show the statistical behavior over all of the $T = 20$ tests, in Figure 2.5 we report the box plots of $\hat{\eta}^{-1}$ and $\hat{C}_{\Psi}^{\delta}$ (Figures 2.5a and 2.5b respectively). They show that the values lie in a very small range, except for very few exceptions, confirming the robustness of the proposed methods.

Results of Experiment B The results obtained in experiment B, described in Section 2.4.2, are reported in Table 2.2 and graphically shown in Figure 2.6. We remark that the difference between this experiment and the previous one is noise injection in the input of each network. Since this technique stabilizes the results, we have tested the methods with higher

$\hat{\delta} = 0$	NN	StNN	ReNN	StReNN	IS
$\hat{\eta}^{-1}$	0.1287	0.1029	0.1296	0.0980	0.0940
$\hat{C}_{\Psi}^{\delta}(\sigma = 0.01)$	32.6398	0.9269	10.9700	0.7046	0.5392

Table 2.1: Values of empirical accuracy and stability constant obtained in experiment A.



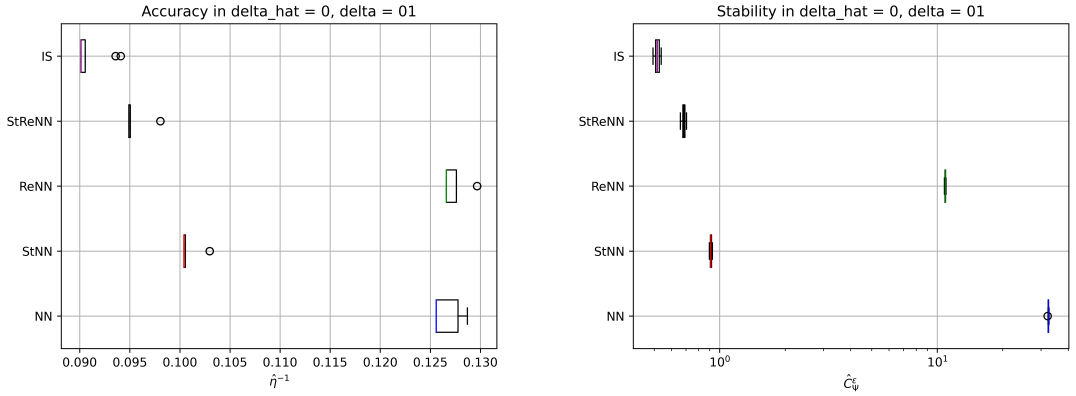
(a) (b)

Figure 2.4: Results obtained in Experiment A. *Left.* Scatterplot of the stability constant versus the accuracy. *Right.* Plot of $\mathcal{E}(\Psi; \boldsymbol{x}^{gt})$ versus δ.

values of the noise standard deviation ($\overline{\sigma} = 0.075$ and $\overline{\sigma} = 0.105$). In Figure 2.6a, we observe that $C_{\Psi}^{\delta} < 1$ for all the reconstructors. Moreover, from the values of C_{Ψ}^{δ} reported in Table 2.2, it is evident that the stabilized reconstructors StNN and StReNN benefit a lot from noise injection. In Figure 2.6 we plot the reconstruction error for increasing noise standard deviation σ in $[0, 0.025]$. We observe that the error values of NN and ReNN methods are almost constant, verifying that noise injection prevents instability (at least up to the noise level introduced as input of the network). For values of $\sigma > 0.025$, the curves relative to NN and ReNN grow rapidly, overtaking StNN and StReNN which show great stability even in the case of high noise levels.

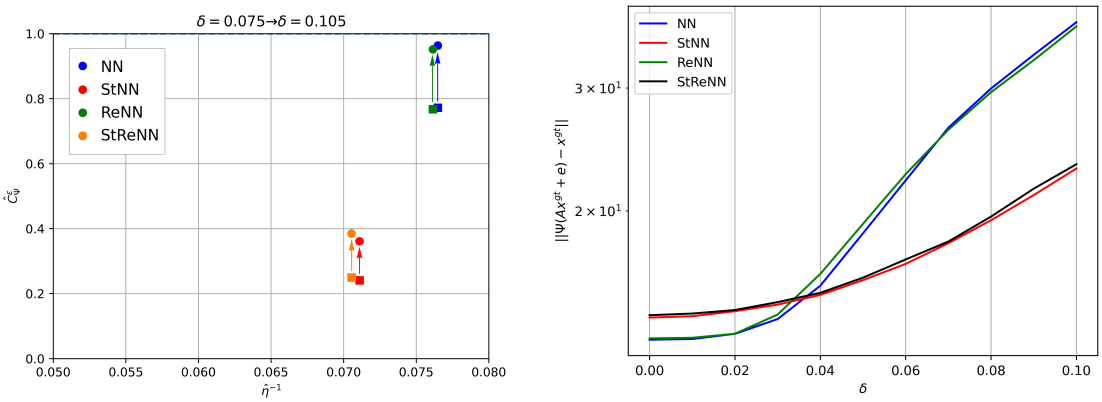
$\sigma^I = 0.025$	NN	StNN	ReNN	StReNN
$\hat{\eta}^{-1}$	0.0764	0.0710	0.0761	0.0705
$\hat{C}_{\Psi}^{\delta}(\sigma = 0.075)$	0.7719	0.2408	0.7674	0.2495
$\hat{C}_{\Psi}^{\delta}(\sigma = 0.105)$	0.9632	0.3608	0.9516	0.3846

Table 2.2: Values of empirical accuracy and stability constant obtained in experiment B.



(a) (b)

Figure 2.5: Boxplot of the results obtained in experiment A for accuracy (*Left*) and stability constant (*Right*) with $T = 20$ executions.



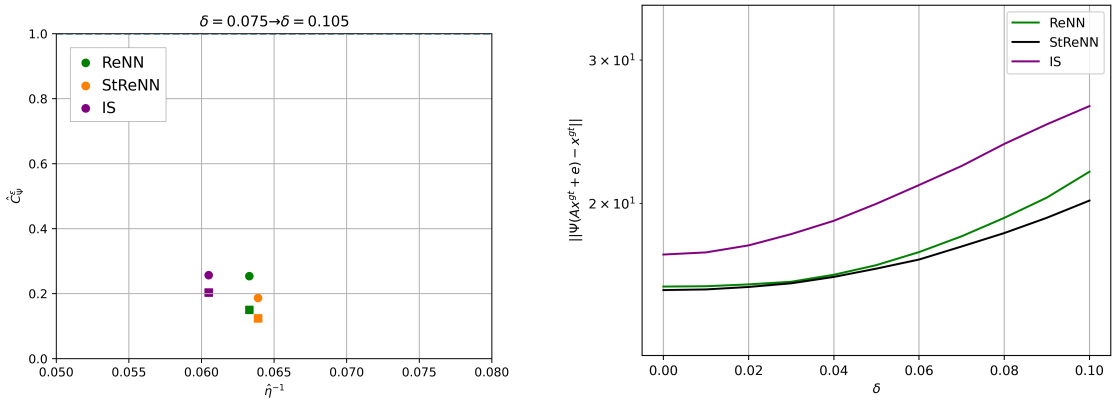
(a) (b)

Figure 2.6: Results obtained in Experiment B. *Left.* Scatterplot of the stability constant versus the accuracy. *Right.* Plot of $\mathcal{E}(\Psi; \boldsymbol{x}^{gt})$ versus σ.

Results of Experiment C The results obtained in experiment C, described in Section 2.4.3, are reported in Table 2.3 and graphically shown in Figure 2.7. We remark that here we aim to experimentally verify Theorem 2.10 by showing that if we train ReNN and StReNN as described in Section 2.4.3, their accuracy and stability constant are close to those of $\Psi^{\lambda,\boldsymbol{L}}$. We test the reconstructors with $\overline{\sigma} = 0.075$ and $\overline{\sigma} = 0.105$. Figure 2.7a shows that the markers relative to ReNN and StReNN are indeed close to the markers associated with IS. Table 2.3 confirms that the accuracy of ReNN and StReNN, as well as the stability constant of ReNN, well approximate those of IS. StReNN appears even more stable than IS since its stability constant is lower than the one of IS by approximately 0.07. These results are confirmed by Figure 2.7.

$\overline{\sigma} = 0.025$	ReNN	StReNN	IS
$\hat{\eta}^{-1}$	0.0633	0.0639	0.0605
$\hat{C}_{\Psi}^{\delta}(\sigma = 0.075)$	0.1500	0.1238	0.2034
$\hat{C}_{\Psi}^{\delta}(\sigma = 0.105)$	0.2536	0.1865	0.2566

Table 2.3: Values of empirical accuracy and stability constant obtained in experiment C.



(a) (b)

Figure 2.7: Results obtained in Experiment C. *Left.* Scatterplot of the stability constant versus the accuracy. *Right.* Plot of $\mathcal{E}(\Psi; \boldsymbol{x}^{gt})$ versus δ.

2.5.2 Comparison of StNN with different architectures and stabilization

In the following, we qualitatively and quantitatively compare the accuracy and the stability of the NN, FiNN, and StNN reconstructions with the three different architectures presented in Section 1.3.3.

Results of Experiment A We show and comment on the results obtained on experiment A described in Section 2.4. We remark that the aim of these tests is to measure the accuracy of the three considered neural reconstructors and of the stabilizers proposed in Section 2.3 and verify their sensitivity to noise in the input data. In a word, how these reconstructors handle the ill-posedness of the imaging inverse problem.

To this purpose, we visually compare the reconstructions of a single test image by the U-Net and 3L-SSNet in Figure 2.8. The first column of each block shows the results of the end-to-end NN reconstructors, where the out-of-domain images are clearly damaged by the noise. The FiNN and, particularly, the StNN stabilizer drastically reduce noise, producing accurate results even for out-of-domain tests.

The computed values of accuracies and stability constants are reported in Table 2.4. Fo-

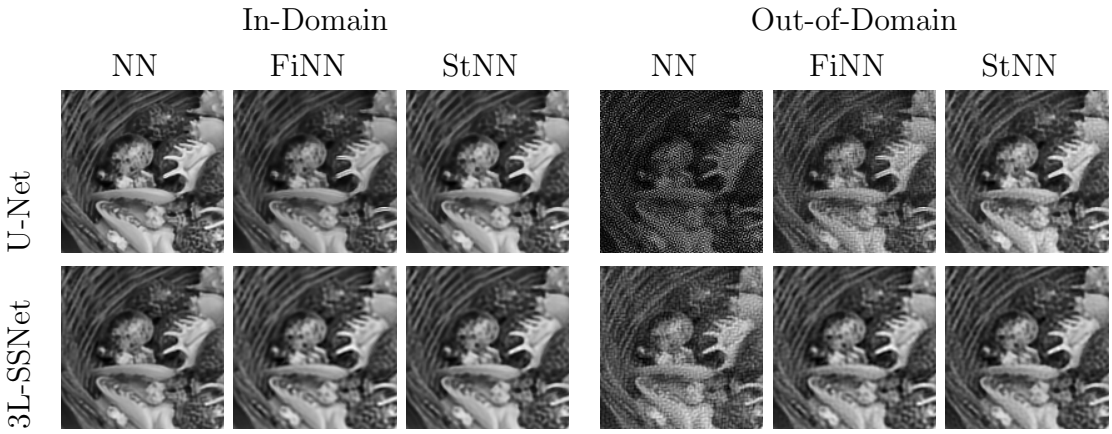


Figure 2.8: Results from experiment A with U-Net and 3L-SSNet.

cusing on the estimated accuracies, the results confirm that NN with the U-Net architecture is the most accurate method, followed by NAFNet and 3L-SSNet, as expected. As a consequence of Theorem 2.4, the values of the stability constant $\hat{C}_{\Psi}^{\delta}$ are in reverse order: the most accurate is the less stable (notice the very high value of $\hat{C}_{\Psi}^{\delta}$ for NN!). By applying the stabilizers, the accuracy is slightly lower but the stability is highly improved (in most of cases the constant is less than one), confirming the efficacy of the proposed solutions to handle noise and, at the same time, maintain good image quality. In particular, StNN is a stable reconstructor independently of the architecture.

	$\hat{\eta}^{-1}$			$\hat{C}_{\Psi}^{\delta}$		
	NN	$FiNN$	$StNN$	NN	$FiNN$	$StNN$
U-Net	0.118	0.085	0.087	36.572	2.519	0.878
3L-SSNet	0.082	0.055	0.072	2.563	0.148	0.243
NAFNet	0.104	0.080	0.078	15.624	1.053	0.434

Table 2.4: Estimated accuracy and stability constants for experiment A on out-of-domain test (input images corrupted by noise with $\delta = 2.56$).

To analyze the stability of the test set with respect to noise, we have plotted in Figure 2.9, for each test image, $\mathcal{E}(\Psi; \boldsymbol{x}^{gt}) - \hat{\eta}$ vs. $\|\boldsymbol{e}\|_2$, where the reconstruction error is defined in (2.30). With green and red dots we have plotted the experiments with stability constant lower and greater than one, respectively, and with the blue dashed line the bisect. We notice that the values reported in Table 2.4 for the empirical stability constant computed as supremum (see Equation (2.29)) are not outliers but they are representative of the results of the whole test set.

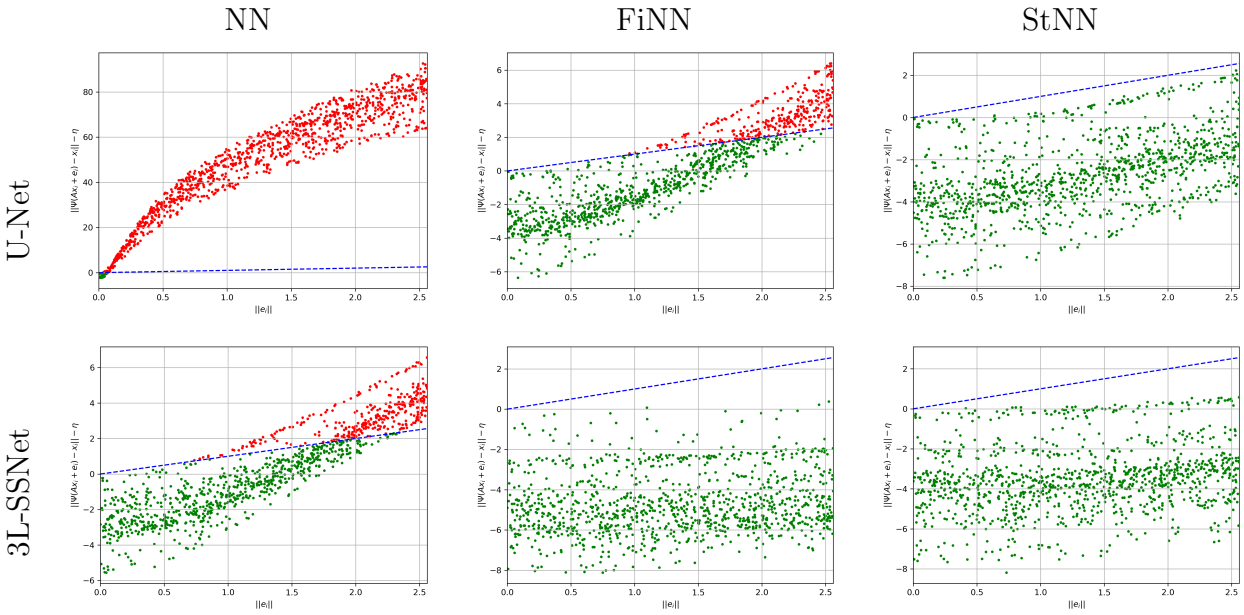


Figure 2.9: Results from experiment A. Plot of $\mathcal{E}(\Psi; \boldsymbol{x}^{gt}) - \eta$ vs. $\|\boldsymbol{e}\|_2$ for all the test images. The blue dashed line represents the bisect.

Results of Experiment B In this experiment we used noise injection in the neural networks training, as described in Section 2.4. This quite common strategy reduces the network's accuracy but improves its stability with respect to noise. However, we show that the reconstructions are not totally satisfactory when we test on out-of-domain images, i.e. when input images are affected by noise of different intensities with respect to training.

Figure 2.10 displays the reconstructions obtained by testing with both in-domain (on the left) and out-of-domain (on the right) images. Even if the NN reconstructions (column 4) are not so injured by noise as in experiment A (see Figure 2.8), however, noise artifacts are clearly visible, especially in U-Net and NAFNet. Both the stabilizers proposed act efficiently and remove most of the noise. We observe that the restorations obtained with FiNN are smoother but also more blurred with respect to the ones computed by StNN.

An overview of the tests is displayed by the boxplots of the SSIM values sketched in Figure 2.11. The light blue, orange, and green boxes represent the results obtained with NN, FiNN, and StNN methods, respectively. They confirm that the neural network's performance worsens with noisy data (see the different positions of light blue boxes from the left to the right column), whereas the proposed frameworks including FiNN and StNN are far more stable.

2.5.3 Analysis with noise varying on the test set

Finally, we have analysed the performance of the methods when the input image $\boldsymbol{y}^{\delta}$ is corrupted by noise $\|\boldsymbol{e}\|_2$ from $\mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$, with σ^2 varying.

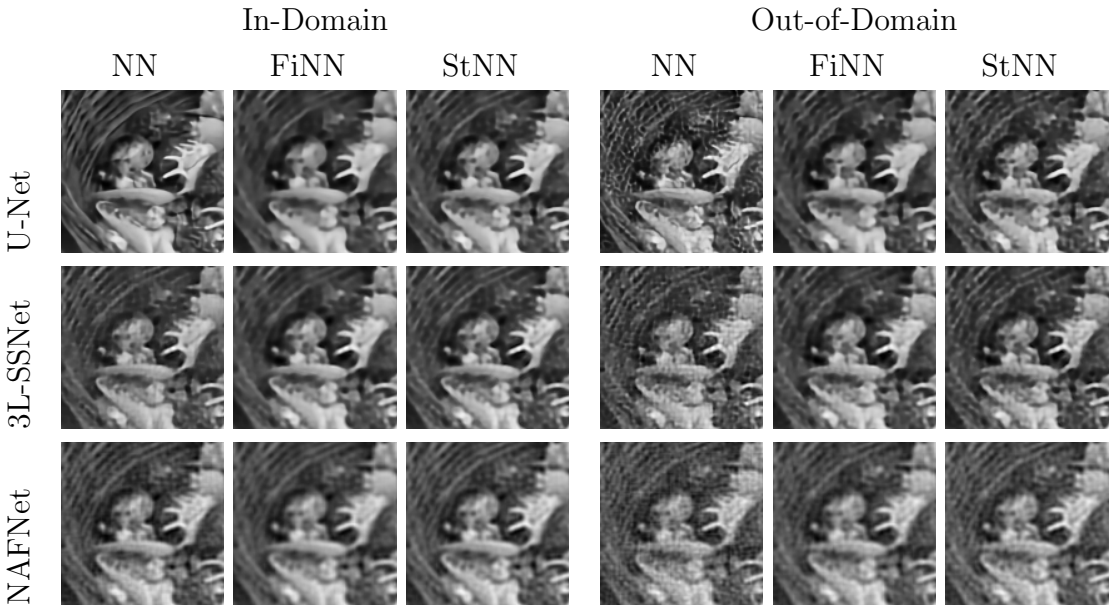


Figure 2.10: Results from the experiment B. On the left, tests with images with the same noise as in the training ($\overline{\sigma} = 0.025$). On the right, tests on images with higher noise ($\overline{\sigma} = 0.075$).

In Figure 2.12 we plot, for one image in the test set, the absolute error between the reconstruction and the true image vs. the noise standard deviation $\overline{\sigma}$. In the upper row the results from experiment A (we remark that in this experiment we trained the networks on no noisy data). The NN error (blue line) is out of range for very small values of $\overline{\sigma}$ for both U-Net and NAFNet, whereas the 3L-SSNet is far more stable. In all the cases, the orange and green line shows that FiNN and StNN improve the reconstruction error. In particular, StNN performs best in all these tests.

Concerning experiment B (in the lower row of the figure), it is very interesting to notice that when the noise is smaller than the training one (corresponding to $\overline{\sigma} = 0.025$) the NN methods are the best performing for all the considered architectures. When $\overline{\sigma} \simeq 0.05$ the behavior changes and the stabilized methods are more accurate.

2.6 Conclusions

In this Chapter, we have theoretically formulated the concept of accuracy and stability in the solution of a discrete linear inverse problem. With these tools in mind, we have performed a theoretical analysis on deep learning-based reconstructors, proving, in Theorem 2.5, that it is not possible to increase stability without decreasing the accuracy of a neural network. To balance the trade-off between stability and accuracy we have proposed new deep learning-based approaches: ReNN, which increases the stability by inheriting, in the network training, regularization from a model-based scheme, and StNN, which stabilizes the solution process

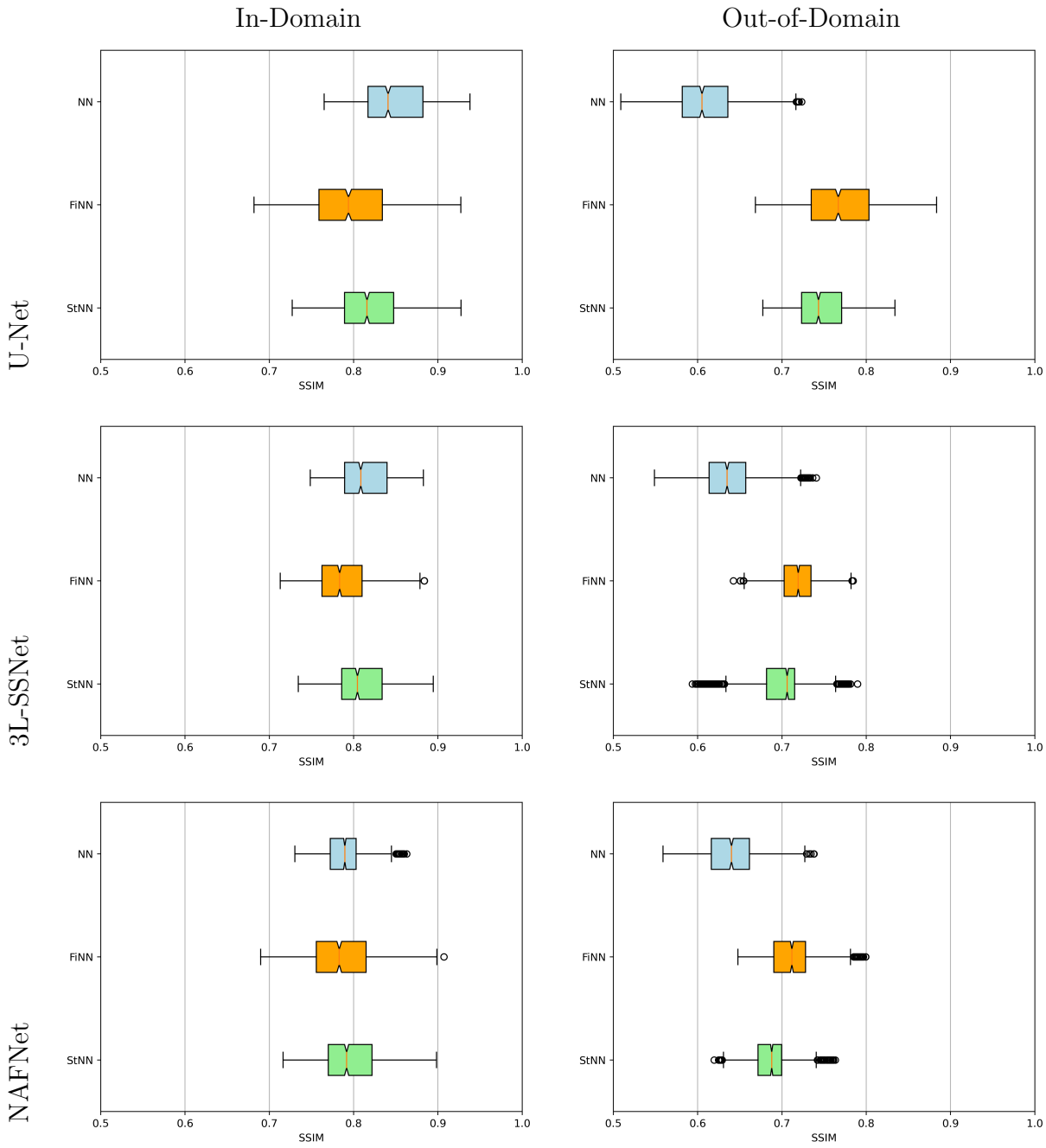


Figure 2.11: Boxplots for the SSIM values in experiment B. The light blue, orange and green boxplots represent the results computed by NN, FiNN and StNN, respectively.

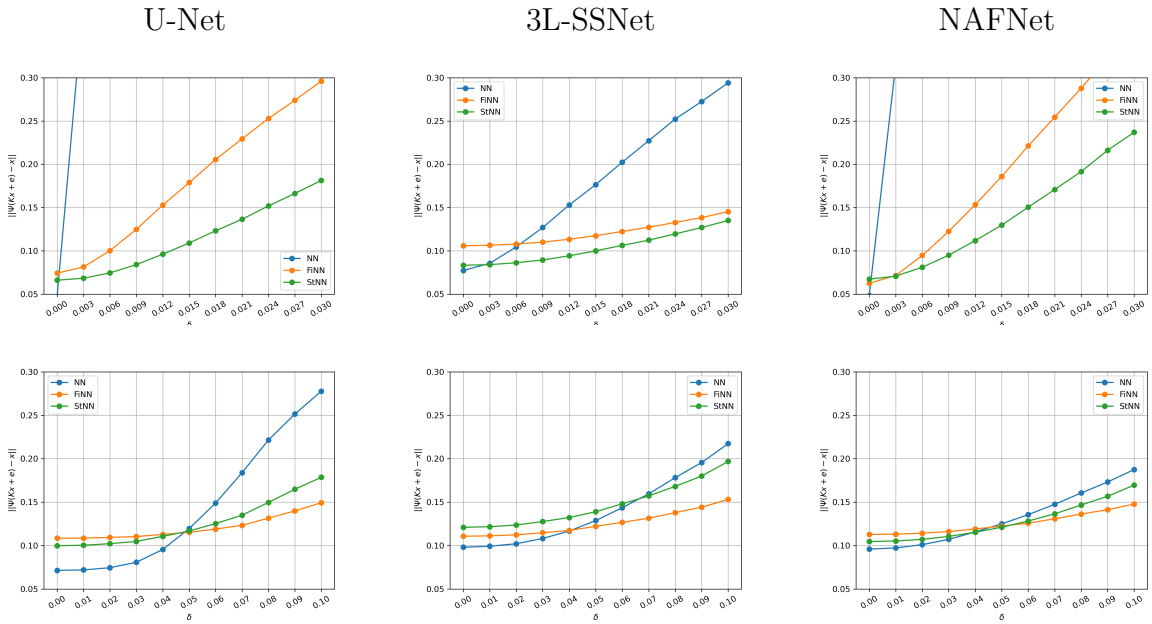


Figure 2.12: Plots of the absolute error vs. the variance σ of the noise for one image in the test set. Upper row: experiment A. Lower row: experiment B.

by reducing the impact of the noise on the input data with few iterations of a model-based algorithm. Combining the two previous approaches we obtain StReNN, which approximates the entire execution of a model-based iterative regularized solver. In the numerical experiments on image deblurring, we compared our proposals with a standard neural network, which exhibits high instability to noise perturbation on the data, by varying the network architecture. The results show that our methods outperform end-to-end neural networks even when stabilization techniques such as noise injection are introduced. In particular, the proposed frameworks reduce the stability constant of NN up to 92% when stabilizers are introduced in StNN and StReNN, with minimal accuracy loss of about $10 - 20\%$ (in one case we even gain accuracy with respect to neural networks). Regarding the choice of the architecture, we show that the very simple 3L-SSNet overcomes UNet and NAFNet in every test where the noise on test images exceeds the noise on the training set, combining the desired characteristics of execution speed and high stability. The choice of the stabilizer is also important. Indeed, we showed that the FiNN increases the stability of the NN-based restoration, but the restored images appear too smooth and a few small details are lost somewhere. On the other side, the Tikhonov StNN proposal, exploiting a model-based formulation of the underlying imaging process, achieves the highest SSIM values in the most challenging out-of-domain cases, confirming its great theory-grounded potential. It represents, indeed, a good compromise between stability and accuracy. We finally remark that the proposed approach can be simply extended to other imaging applications modeled as an inverse problem, such as super-resolution, denoising, or tomography, where the neural networks learning the map

from the input to the ground truth image cannot efficiently handle noise in the input data.

Chapter 3

RISING: an unsupervised and stable data-driven approach for Sparse Computed Tomography

Combining healthy protocols with high-quality images is one of the most important components of medical imaging and a crucial target for researchers involved in minimal invasive Computed Tomography (CT). Radiologists, manufacturers, and medical physicists have implemented many examination protocols as well as software and hardware modifications to reduce the harmful ionizing radiations and pave the way to X-ray imaging for screening tests, pediatric cases, or pre-surgical exams. There are two main techniques allowing for a significant reduction of the total radiation exposure per patient. The first one consists of reducing the X-ray tube current at each scan (*Low-Dose CT*), without changing the traditionally used CT full geometry. The resulting measured data is very noisy, due to excessive quantum noise. The second practical way to lower the radiation per person consists of reducing the number of X-ray projections (*SparseCT*), which leads to incomplete tomographic data, but very fast examinations.

In this Chapter, we focus on SparseCT images, but we reasonably argue that the proposed algorithm could be also applied with success to low-dose CT. In the case of sub-sampled data, the image reconstruction is tricky and conventional Filtered Back-Projection (FBP) algorithms, widely exploited in classical CT, do not provide stable reconstructions as the recovered images typically suffer from severe striking artifacts, as already remarked in Section 1.3.1. The already introduced model-based iterative methods (see Section 1.3.2) represent a widely used alternative approach. They model the image reconstruction as a mathematical linear inverse problem which is solved, in the discrete setting, by minimizing a constrained or unconstrained function combining a data-fitting term and a regularizer. Due to the lack of projections, the SparseCT operator is non-injective and the associated inverse problem has

infinite possible solutions [22]. The embedding of a sparsifying regularizer mitigates the lack of many projection views, according to the Compressed Sensing theory [35]. An exhaustive review of model-based reconstruction methods can be found in [94].
Recently, Deep Learning (DL) based methods have emerged over fully conventional and variational approaches for few-view tomographic reconstruction [95]. A widely used strategy is the so-called Learnt Post Processing (LPP) approach, introduced in Section 1.3.3: a two-step scheme where first a low-quality image with artifacts and noise is reconstructed with a fast method (typically an FBP) and then a neural network suppresses the artifacts. Usually, the network learns from a set of *ground truth* images, reconstructed from full dose acquisitions in a long offline phase. The pioneering works by Han in 2016 and 2018 demonstrated the superiority of LPP strategies over some model-based iterative algorithms for SparseCT images [59, 96].

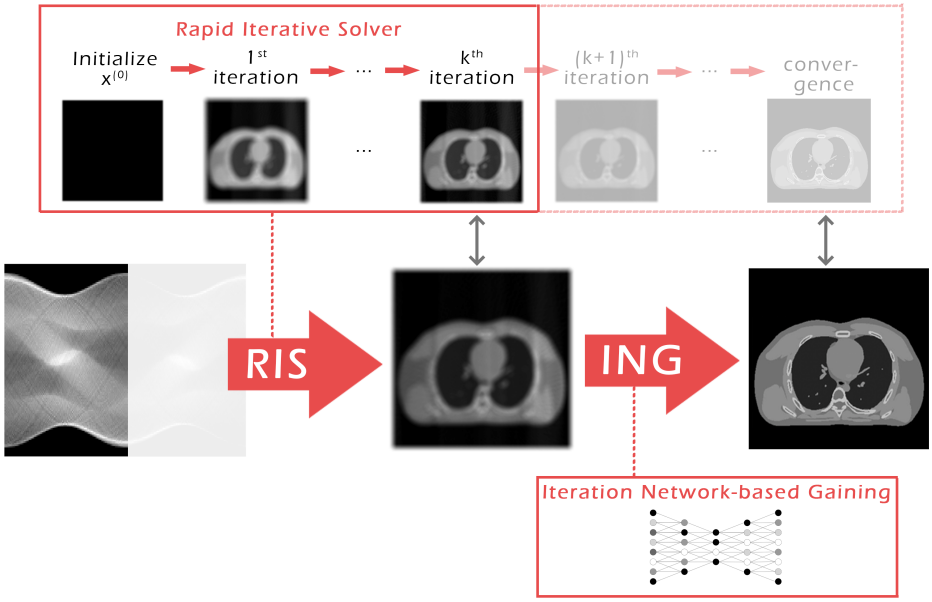


Figure 3.1: Graphical draft of the proposed two-step RISING workflow for tomographic reconstruction from few-view data.

The main disadvantage of model-based iterative reconstruction algorithms is their high computational cost since they typically need several iterations to achieve high-quality results. In real systems, in order to fulfill the clinical requirements of time per exam, only very few iterations of the algorithms can be performed, leading to a solution that is far from being the best achievable one. However, by considering fast iterative methods, the images reconstructed in a few iterations already contain many details of interest of the scanned object [97, 28].
Focusing on LPP-based approaches, the work by Sidky et al. [98] claims and demonstrates that these schemes do not compute the solution of the CT inverse problem and they can introduce, in the reconstruction, structures not belonging to the scanned objects. On the

contrary, numerical evidence shows that model-based methods compute a good solution to the inverse problem, according to the mathematical sense that will be defined in the next Section. In addition, it is worth noticing that neural networks need to be trained on task-specific data sets to properly learn both the degradation effects to be removed and the anatomical details to be preserved, which characterize each typology of medical imaging; hence a further relevant challenge for medical applications is the lack of precise training data [95].

Strong on this awareness, in this Chapter we propose to extend the StReNN framework, introduced in Chapter 2, to the resolution of the SparseCT inverse problem. This method is particularly promising in this scenario since it does not require any full-dose target image, attenuating the difficulties related to data acquisition in the medical setup, cited in the last paragraph. We refer to our proposal as the *RISING* (Rapid Iterative Solver with Iteration Network-based Gaining) framework, whose graphical draft is depicted in Figure 3.1. RISING is conceived as one reconstructing procedure, described by the two following steps, executed in sequence:

- Starting from the sub-sampled projection data, a rapid model-based iterative algorithm produces a preliminary coarse reconstruction, in a few iterations. The execution of only a few iterations fits realistic time constraints. This step acts as a Stabilizer as described in Section 2.4. Note that in this particular application, the pre-processing step is necessary not only to enhance the method's stability by mitigating the influence of noise, similar to its role in image deblurring inverse problems, but also to incorporate the prior information regarding the solution through the regularization term in the initial reconstruction.

- The computed rough reconstruction is post-processed by a pre-trained convolutional neural network, providing the RISING solution. By completing the iterations up to the iterative method convergence, we obtain images used as targets for the neural network training, as in the Regularized Neural Network (ReNN) scheme (see Section 2.3). This ensures a fully consistent training set with respect to the system geometry, and it is applicable to all types of medical images and few-view protocols.

Contributions This work has a dual purpose. On one side, we aim to combine the fast execution of LPP methods with the stability of model-based iterative algorithms used in sparse CT reconstruction. The variational robustness can face, in fact, the lack of projection data and the presence of noise on the projections, thanks to the use of suitable regularization functions acting as priors onto the solution. As the execution of a neural network is very fast, the use of DL greatly speeds up the whole reconstruction.
On the other side, we intend to remedy the lack of task-specific CT data sets by using, for network training, images created by the same system and under the same geometry used for

the reconstruction. This is attainable for every CT system and makes the RISING framework suitable for real use.

This Chapter is based on my publication [7]. My personal contributions to the cited work were the development of the methods, the implementation of the code to test the experiments and the experimental setup. The code to replicate the experiments is available at `https://github.com/loibo/RISING`.

Structure of the Chapter The Chapter is organized as follows. In Section 3.1 we describe the proposed RISING framework. In Section 3.2 we introduce the experimental settings considered to achieve the results reported and discussed in Section 3.3. At last, final conclusions are drawn in Section 3.4.

3.1 The RISING framework

In this Section we describe the proposed RISING framework by focusing on the Rapid Iterative Solver (RIS) phase in Section 3.1.1 and on the Iteration Network-based Gaining (ING) phase in Section 3.1.2. For simplicity, we consider in this paper only the two-dimensional case. The extension to three dimensions is straightforward.

3.1.1 Rapid Iterative Solver

In the discrete setting, the CT process of X-ray absorption is expressed as:

$$\boldsymbol{y} = \boldsymbol{K}\boldsymbol{x}^{gt} \tag{3.1}$$

where the unknown vector $\boldsymbol{x}^{gt} \in \mathbb{R}^n$ denotes the image to reconstruct, the right-hand side term $\boldsymbol{y}$ is an m-dimensional vector containing the noisy projection measurements and the $m \times n$ system matrix $\boldsymbol{K}$ is the discretization of the X-ray physical process projecting an image onto the detector. We say that a system of linear equations is *solvable* in a subset $\mathcal{X}$ of $\mathbb{R}^n$ if it admits a unique solution in $\mathcal{X}$. In case of SparseCT protocols, Equation (3.1) is not solvable in $\mathbb{R}^n$ since $m < n$, and the under-determined linear system admits infinite solutions. According to the Compressed Sensing theory [35], if the desired solution $\boldsymbol{x}^{gt}$ of (3.1) is sparse in some transform $\boldsymbol{T} \in \mathbb{R}^{s \times n}$, then Equation (3.1) is solvable in the subset $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n | \; ||\boldsymbol{T}\boldsymbol{x}||_0 \leq s\}$ for $s \ll n$, where $\|\cdot\|_0$ is the ℓ_0 semi-norm counting the non-zero elements of the vector argument. Hence the CT inverse problem can be reformulated as the following minimization:

$$\arg\min_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{T}\boldsymbol{x}\|_0 \text{ s.t. } \boldsymbol{K}\boldsymbol{x} = \boldsymbol{y}. \tag{3.2}$$

which admits a unique solution if $\boldsymbol{K}$ is full-rank.

Equation (3.2) is usually relaxed in the following, more easily solvable, form:

$$\arg\min_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{T}\boldsymbol{x}\|_1 \text{ s.t. } \boldsymbol{K}\boldsymbol{x} = \boldsymbol{y}, \tag{3.3}$$

where $\|\cdot\|_1$ is the ℓ_1 norm [99]. An unconstrained formulation of (3.3) can be stated as:

$$\arg\min_{\boldsymbol{x}} \|\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{T}\boldsymbol{x}\|_1 \tag{3.4}$$

where λ is a suitable positive parameter (also called *regularization parameter*). Convergent iterative schemes, such as the lagged diffusivity fixed point algorithm [26] or FISTA [100], are commonly exploited to solve (3.4). Sometimes, the constraint $\boldsymbol{x} \geq \boldsymbol{0}$ is added to (3.3) or (3.4), to preserve the physical non-negativity property of the attenuation coefficients [28, 101, 102].

In the following we will consider the particular minimization problem:

$$\arg\min_{\boldsymbol{x}\geq\boldsymbol{0}} \|\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \mathrm{TV}_\beta(\boldsymbol{x}). \tag{3.5}$$

where TV_β is a smoothed differentiable version of the Total Variation [103], obtained by setting $\boldsymbol{T}$ as the image gradient discretization, to exploit sparsity in the image gradient domain. It is defined as:

$$\mathrm{TV}_\beta(\boldsymbol{x}) = \sum_{i=1}^{n} \sqrt{(\boldsymbol{D}_h\boldsymbol{x})_i^2 + (\boldsymbol{D}_v\boldsymbol{x})_i^2 + \beta^2} \tag{3.6}$$

with β fixed small positive parameter.

Among the wide class of iterative solvers for the problem (3.5), we select the Scaled Gradient Projection (SGP) algorithm, proposed in 2008 for image deblurring [43]. The SGP has been successfully applied with some acceleration techniques for SparseCT reconstructions in the papers [25, 97, 28], showing that the objects of interest are already distinguishable in the image computed after very few iterations of the method. Regarding the convergence, it is proved in [46] that the theoretical convergence rate of the SGP at the unique minimum of (3.5) is $\mathcal{O}(1/\mathrm{k})$, even if empirical tests reveal that the SGP performances are closely comparable to the convergence rate of optimal algorithms.

The first step of the RISING framework performs a predefined (and relatively small) number M of iterations of the considered Rapid Iterative Solver (RIS). We remark that we stop the SGP far before its convergence to meet the constraints imposed by clinical settings, where very short computational times are admitted for the reconstruction process. The early solution $\boldsymbol{x}^{(M)}$ is denoted as $\boldsymbol{x}_{RIS}$ in the following and it represents the input of the neural network which enhances the image in a new perspective.

3.1.2 Iteration Network-based Gaining

The second step of the RISING framework implements the Iteration Network-based Gaining (ING) task. From our previous works [25, 97] we know that, due to the empirical rapidity of

the SGP, in the achieved coarse image reconstruction the anatomical structures are present but typically more blurred and less visible than on the $\boldsymbol{x}_{IS}$ image computed at the iterative solver convergence. Hence, we train a CNN to learn the transformation mapping the early solution $\boldsymbol{x}_{RIS}$ to the corresponding $\boldsymbol{x}_{IS}$.
Formally, we denote with $\mathcal{T}_k$ the function describing the action of the j-th iteration of the solver:

$$\boldsymbol{x}^{(k+1)} = \mathcal{T}_k(\boldsymbol{x}^{(k)}; \boldsymbol{y}) \quad \forall k \geq 0.$$

Thus, the whole iterative process can be expressed as the concatenation of the following g_M and f_M functions:

$$g_M := \mathcal{T}_{M-1} \circ \cdots \circ \mathcal{T}_0 \tag{3.7}$$

and

$$f_M := \mathcal{T}_{M^*} \circ \cdots \circ \mathcal{T}_{M+1} \circ \mathcal{T}_M. \tag{3.8}$$

where M^* is the number iterations necessary to achieve the $\boldsymbol{x}_{IS}$ solution. In fact, it holds:

$$f_M(g_M(\boldsymbol{x}^{(0)}; \boldsymbol{y})) = \boldsymbol{x}_{IS}. \tag{3.9}$$

Since we compute $\boldsymbol{x}_{RIS} = g_M(\boldsymbol{x}^{(0)}; \boldsymbol{y})$ in the RIS step, in the ING phase we approximate f_M by learning the map from $\boldsymbol{x}_{RIS}$ to $\boldsymbol{x}_{IS}$ with a CNN. Ideally, if the network would perfectly learn f_M, its output $\boldsymbol{x}_{RISING}$ should be equal to $\boldsymbol{x}_{IS}$. It is known that this is not provable in practice; nevertheless, we show numerical evidence through very accurate RISING reconstructions on simulations, in Section 3.3.

3.2 Experimental design and implementation notes

To numerically verify the feasibility of the RISING workflow, we develop numerical simulations on a data set of images with geometric elements, where measures of merits can be computed, as well as real on a data set of medical images, to properly understand the potential of RISING for clinical applications and settings. We generate the sinograms by projecting the images of the data sets according to 2D fan-beam geometries, and by adding zero-mean Gaussian noise to the projections.

This Section describes the experimental design, whereas the results are reported and discussed in Section 3.3.

3.2.1 Data set of synthetic images

To fully exploit the full-reference image quality assessment metrics and validate our experiments, we consider the Constrasted Overlapping Uniform Lines and Ellipses (COULE) dataset, described in Section 1.4.1. The left image of Figure 3.2 shows one image of the dataset as an example.

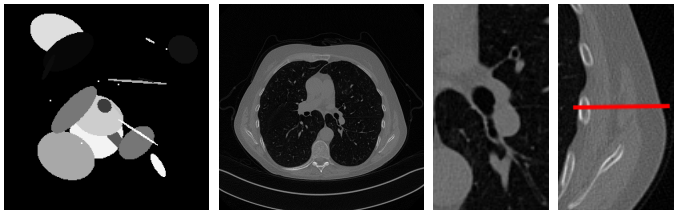


Figure 3.2: Ground-truth images from the COULE (on the left) and the Low Dose Mayo (center) data sets, with two zoomed crops (on the right) on regions with different anatomical structures.

To address sparse-view CT reconstructions, we considered different protocols: the first one is a full angular acquisition with 1-degree spaced projections (we call it $P_{360,360}$ in the following); in the others, we reduce the number of acquired projections to 180 ($P_{360,180}$) and 60 ($P_{360,60}$).

3.2.2 Data set of real medical images

As real patient images, we have downloaded the widely used AAPM Low Dose CT Grand Challenge data set by the Mayo Clinic [75], described in Section 1.4.1. In Figure 3.2 we depict one image with two zooms-in highlighting areas with different anatomical structures, such as pulmonary details, Sections of ribs, and low-contrast inter-costal muscles. On this data set, we test the $P_{360,360}$ protocol and a very sparse geometry with only 60 projections in 180 degrees scanning trajectory ($P_{180,60}$). As observable from Figure 3.2, real full-dose medical images still present little noise and slightly visible streaking artifacts. This makes it quite difficult to compute reliable metrics on the reconstructions by referencing them as ground truth.

3.2.3 Network architecture and training

As CNN architecture we use the state-of-the-art ResU-Net architecture as described in Section 1.3.3. Given the data set $\{(\boldsymbol{x}_{RIS,i}, \boldsymbol{x}_{IS,i})\}_{i=1,\dots,N}$, we denote by $F_{\Theta^*}(\boldsymbol{x}_{RIS,i})$ the action of the neural network on the input $\boldsymbol{x}_{RIS,i}$ and by $\boldsymbol{x}_{RISING,i} = F_{\Theta^*}(\boldsymbol{x}_{RIS,i})$ the network output. We estimate the parameters Θ^* by network training, where the loss function is set as $\ell(\boldsymbol{x}_{IS,i}, \boldsymbol{x}_{RISING,i}) = ||\boldsymbol{x}_{RISING,i} - \boldsymbol{x}_{IS,i}||_2^2$, so that:

$$\Theta^* = \arg\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(\boldsymbol{x}_{IS,i}, \boldsymbol{x}_{RISING,i}). \tag{3.10}$$

The training is performed by running Adam for 100 and 50 epochs for the COULE and Mayo datasets, respectively. The batch size is fixed equal to 8 in all the experiments (this is the largest batch size usable in our Nvidia RTX A4000 GPU). The step size for

the optimization algorithm decreases with polynomial decay, going from 10^{-3} to 10^{-5}. To increase the stability over the first iterations, we clip the gradient to 5.

3.2.4 Implementation notes

As already presented in Section 3.1.1, we use SGP as the iterative solver. In each iteration, we use the class OpTomo of ASTRA toolbox [104, 105] to compute the matrix-vector products $\boldsymbol{K}\boldsymbol{x}$ and $\boldsymbol{K}^T\boldsymbol{y}$, where $\boldsymbol{K}$ is the discretization of the fan-beam Radon transform as in equation (3.1).
In our experiments, we set the smoothing parameter for the TV regularizer to $\beta = 10^{-3}$, the regularization parameter as $\lambda = 10^{-5}$ for the Mayo data set and $\lambda = 4 \cdot 10^{-5}$ for the COULE data set.

To generate the images $\boldsymbol{x}_{IS}$, we consider the following stopping conditions:

$$\begin{cases} ||\nabla \mathcal{J}(\boldsymbol{x}^{(k)})||_2 < \tau_1 ||\nabla \mathcal{J}(\boldsymbol{x}^{(0)})||_2 \\ ||\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}||_2 < \tau_2 ||\boldsymbol{x}^{(k-1)}||_2 \end{cases} \tag{3.11}$$

where $\nabla \mathcal{J}$ is the gradient of our objective function (3.5) and $\tau_1 = \tau_2 = 10^{-6}$ in the experiments. All the SGP inner parameters are taken from [25].

3.3 Experimental results and discussion

In this Section, we report and discuss the more representative numerical experiments performed using the proposed workflow for tomographic image reconstruction from few-view data.

3.3.1 Robustness of RISING with respect to data perturbation

We first analyze the robustness of the proposed RISING framework with respect to noise on the sinogram. We compare it with an LPP-based traditional algorithm (called FBP-LPP in the following), where the same U-Net architecture used in RISING is trained to map low-quality FBP reconstructions to ground truth images [6]. We obtained a noisy projection vector $\boldsymbol{y}^\delta$ from the non-noisy sinogram $\boldsymbol{y}$ as:

$$\boldsymbol{y}^\delta = \boldsymbol{y} + \delta \frac{||\boldsymbol{y}||_2}{||\boldsymbol{e}||_2} \boldsymbol{e}, \quad \delta = 0.01, \tag{3.12}$$

where $\boldsymbol{e}$ is AWGN with standard deviation equal to one. We trained the networks on reconstructions computed from the data $\boldsymbol{y}^\delta$ and, to verify the methods' robustness, we tested them both on sinograms $\boldsymbol{y}^\delta$ and $\boldsymbol{y}^{\delta'}$, corrupted by further noise:

$$\boldsymbol{y}^{\delta'} = \boldsymbol{y}^\delta + \delta' \frac{||\boldsymbol{y}^\delta||_2}{||\boldsymbol{e}||_2} \boldsymbol{e}. \tag{3.13}$$

Table 3.1: Results on the COULE synthetic test data set, under the $P_{360,360}$ CT protocol, reporting mean and standard deviation (StdDev) for different reconstructions obtained with low noise data $\boldsymbol{y}^{\delta}$ (Equation (3.12)) and high noise data $\boldsymbol{y}^{\delta'}$ (Equation (3.13)).

		$\boldsymbol{x}_{FBP}$		$\boldsymbol{x}_{RIS}$		$\boldsymbol{x}_{FBP-LPP}$		$\boldsymbol{x}_{RISING}$	
		Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
RE	$\boldsymbol{y}^{\delta}$	0.1027	0.0089	0.2501	0.0319	0.0292	0.0097	0.0574	0.0869
	$\boldsymbol{y}^{\delta'}$	0.2554	0.0132	0.2542	0.0302	0.1294	0.0333	0.0989	0.0128
SSIM	$\boldsymbol{y}^{\delta}$	0.8926	0.0252	0.8113	0.0686	0.9969	0.0019	0.9801	0.0093
	$\boldsymbol{y}^{\delta'}$	0.4891	0.0822	0.7916	0.0697	0.7475	0.1537	0.8982	0.0475

Figure 3.3 shows the boxplots of Relative Error (RE) and SSIM values drawn from the results on the COULE data set, with the $P_{360,360}$ geometry and $\delta = 0.05$. They show that the quality of the images by the FBP-LPP method highly decreases (especially for the SSIM metric) when we consider the noisy data $\boldsymbol{y}^{\delta'}$, from the best to the worst values. The results obtained with RISING framework are much more stable: the RE and SSIM values moderately worsen, and the final scores outperform the FBP-LPP. Moreover, the RISING boxplot length in the presence of noise is quite small, showing that the method performance does not have extreme values on the whole test set. Table 3.1 reports the mean and standard deviation (StdDev in the table) computed on the FBP reconstruction $\boldsymbol{x}_{FBP}$, the $\boldsymbol{x}_{RIS}$ low-quality image and on the two final outputs $\boldsymbol{x}_{FBP-LPP}$ and $\boldsymbol{x}_{RISING}$. It confirms that RISING is much more robust than FBP-LPP with respect to noise on the data.
The same information on the Mayo test set with geometry $P_{180,60}$ and $\delta' = 0.04$ is reported in Table 3.2. We underline that the SSIM values decrease by about 17% for FBP-LPP and only 9.8% for RISING, when applied to high noise data $\boldsymbol{y}^{\delta'}$. This provides medical images with the same outcomes obtained on synthetic data.

The previous findings confirm that the widely used FBP-LPP approach gives very good results when applied to images corrupted by the same noise degradating the training samples: in this case, it exploits at its best the information provided by the available ground truth images and it outperforms RISING. On the contrary, when the sinograms are affected by unseen noise (as it is common in real applications), the quality of the RISING reconstructions is way better.

3.3.2 Results on synthetic images

In this paragraph, we focus on the synthetic COULE data set.

As first in-depth analysis, we set the geometry $P_{360,360}$ and we compute the $\boldsymbol{x}_{RIS}$ early solution for $M = \{3, 5, 10\}$. Figure 3.4 shows the $\boldsymbol{x}_{RIS}$ starting images in the top row and

Table 3.2: Results on the Mayo test data set, under the $P_{180,60}$ CT protocol, reporting mean and standard deviation (StdDev) for different reconstructions obtained with low noise data $\boldsymbol{y}^{\delta}$ (Equation (3.12)) and high noise data $\boldsymbol{y}^{\delta'}$ (Equation (3.13)).

		$\boldsymbol{x}_{FBP}$		$\boldsymbol{x}_{RIS}$		$\boldsymbol{x}_{FBP-LPP}$		$\boldsymbol{x}_{RISING}$	
		Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
RE	$\boldsymbol{y}^{\delta}$	0.4518	0.0264	0.1803	0.0249	0.1004	0.0164	0.1236	0.0294
	$\boldsymbol{y}^{\delta'}$	1.300	0.0609	0.2326	0.0251	0.2026	0.0113	0.2041	0.0229
SSIM	$\boldsymbol{y}^{\delta}$	0.3672	0.0620	0.8730	0.0283	0.9265	0.0148	0.9145	0.0175
	$\boldsymbol{y}^{\delta'}$	0.085	0.0305	0.8016	0.0389	0.7689	0.0681	0.8231	0.0255

Table 3.3: Mean and standard deviation values of the quality metrics, evaluated on the COULE test set for different RIS reconstructions under the $P_{360,360}$ protocol.

		M=3		M=5		M=10		convergence	
		Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
RE	$\boldsymbol{x}_{RIS}$	0.6152	0.0366	0.4996	0.0548	0.2700	0.0239	-	-
	$\boldsymbol{x}_{RISING}$	0.0681	0.0098	0.0784	0.0844	0.0781	0.0122	-	-
	$\boldsymbol{x}_{IS}$	-	-	-	-	-	-	0.0207	0.0051
SSIM	$\boldsymbol{x}_{RIS}$	0.2264	0.0262	0.3139	0.0919	0.7844	0.0481	-	-
	$\boldsymbol{x}_{RISING}$	0.9630	0.0133	0.9267	0.0389	0.9674	0.0140	-	-
	$\boldsymbol{x}_{IS}$	-	-	-	-	-	-	0.9980	0.0010

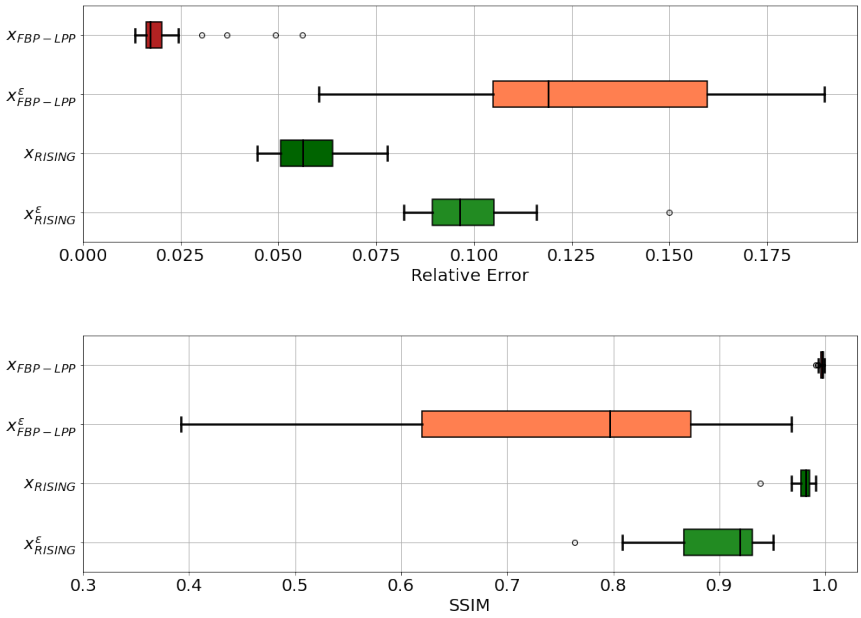


Figure 3.3: Results on the COULE synthetic test data set, under the $P_{360,360}$ CT protocol. On the x-axis the values of the metrics RE (top) and SSIM (bottom). On the y-axis the framework used from low noise data $\boldsymbol{y}^{\delta}$ (Equation (3.12)) and high noise data $\boldsymbol{y}^{\delta'}$ (Equation (3.13))

.

the $\boldsymbol{x}_{RISING}$ final reconstructions in the bottom row (relative to the example image of Figure 3.2). They show that the network is able to almost perfectly restore all three input images. From Table 3.3 we observe that the $\boldsymbol{x}_{RISING}$ images are very accurate, for $M = \{3, 5, 10\}$, highlighting that the number M of starting iterations seems not to notably influence the final results. Moreover, the table shows that the $\boldsymbol{x}_{IS}$ solution is a reliable target for the network training since it approximates $\boldsymbol{x}^{gt}$ very well.

To understand the behavior of our approach at increasing sparsity in the CT protocol, we also test RISING at different geometric settings such as $P_{360,360}$, $P_{360,180}$ and $P_{360,60}$. The number of iterations used to generate $\boldsymbol{x}_{RIS}$ is $M = 10$ and the M^* iterations needed for convergence is in $[150, 300]$ for all the test images. In Table 3.4 we report the quality indices evaluated on the test set in terms of mean and standard deviation. We first interestingly observe that the values of all the $\boldsymbol{x}_{RIS}$ outputs are very similar, independently from the geometry. Considering the final reconstructions $\boldsymbol{x}_{RISING}$, we see that halving the number of angles in $P_{360,180}$ does not affect the mean values of the metrics; when the geometry is very sparse in $P_{360,60}$ the errors slightly increase. However, the values of the standard deviations are small and very similar in all the tests, showing that the network has a stable behavior. As before, the $\boldsymbol{x}_{IS}$ solutions have excellent metrics, justifying the use of the RISING approach even in the hardest case with only 60 angles. This shows that the considered compressed sensing-based model (3.5) properly describes the reconstruction process independently of the

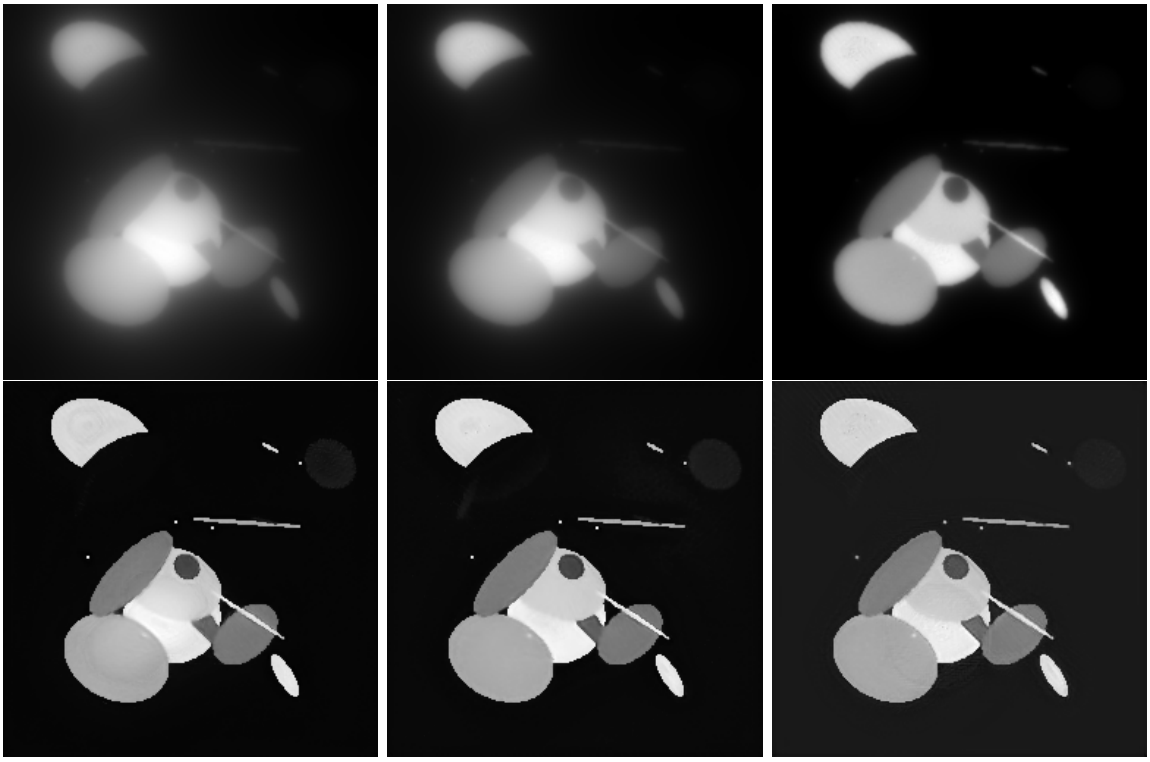


Figure 3.4: Results on a test image from the COULE synthetic data set, under the $P_{360,360}$ CT protocol. First row, from left to right: $\boldsymbol{x}_{RIS}$ with $M = 3$, $M = 5$ and $M = 10$ respectively; second row, from left to right: the corresponding $\boldsymbol{x}_{RISING}$.

sparsity of the geometry (at least for the considered ones).

3.3.3 Results on real medical images

Now we present the results of RISING applied to the Mayo data set. The results obtained with two sparse-view CT geometries, namely $P_{360,360}$ and $P_{180,60}$, are shown in Figure 3.5. The left images are the $\boldsymbol{x}_{RIS} = \boldsymbol{x}^{(15)}$ reconstructions. Even if only a small number of iterations are performed, the main structures of the abdomen are visible; however, the images are still blurry and few streaking artifacts are visible in the bottom image.
Our solutions $\boldsymbol{x}_{RISING}$ are reported on the right. They both appear very accurate, the low-contrast regions are correctly preserved and the noise is not visible. In the case of the more challenging protocol $P_{180,60}$, the thin details are less neat but still present and discernable. Figure 3.6 plots the intensity profiles taken over the red line on the crop of Figure 3.2. In particular, we compare the unseen target profile (black line) from $\boldsymbol{x}_{IS}$ with the network input $\boldsymbol{x}_{RIS}$ curve (blue line) and the output $\boldsymbol{x}_{RISING}$ curve (red line) for both the geometries. In the left image ($P_{360,360}$) we observe that starting from the input profile, quite far from the black one, we obtain a result faithful to the target, mirroring that the CNN has accurately learnt a map well approximating f_M of (3.8). Also in the right image ($P_{180,60}$) the line

Table 3.4: Mean and standard deviation values of the quality metrics, evaluated on the COULE test set for different geometries.

		$P_{360,360}$		$P_{360,180}$		$P_{360,60}$	
		Mean	StdDev	Mean	StdDev	Mean	StdDev
RE	$\boldsymbol{x}_{RIS}$	0.2700	0.0239	0.2706	0.0240	0.2834	0.0282
	$\boldsymbol{x}_{RISING}$	0.0781	0.0122	0.0676	0.0123	0.1113	0.0223
	$\boldsymbol{x}_{IS}$	0.0207	0.0051	0.0302	0.0077	0.0668	0.0147
SSIM	$\boldsymbol{x}_{RIS}$	0.7844	0.0481	0.7831	0.0485	0.7472	0.0643
	$\boldsymbol{x}_{RISING}$	0.9674	0.0140	0.9741	0.0115	0.9493	0.0117
	$\boldsymbol{x}_{IS}$	0.9980	0.0010	0.9951	0.0030	0.9753	0.0141

corresponding to the RISING solution almost overlaps the target one. At last, we underline that the solutions $\boldsymbol{x}_{IS}$ of the regularized model are very similar in the case of $P_{360,360}$ and $P_{180,60}$ geometries.

3.4 Conclusions

We have proposed a novel framework, called RISING, for the reconstruction of a CT image from few-views. RISING is ground truth-free, fast, and robust with respect to the noise, hence it is a suitable framework for clinical real usage. The numerical experiments performed both on synthetic and real medical images show that the RISING reconstructions are visually accurate, even in a very sparse geometry with only 60 views in $[0, 180]$ degrees. At last, we underline that the RISING scheme is flexible since it can be set on different model-based iterative reconstruction methods (not limited to the model in (3.5) and/or to the SGP algorithm).

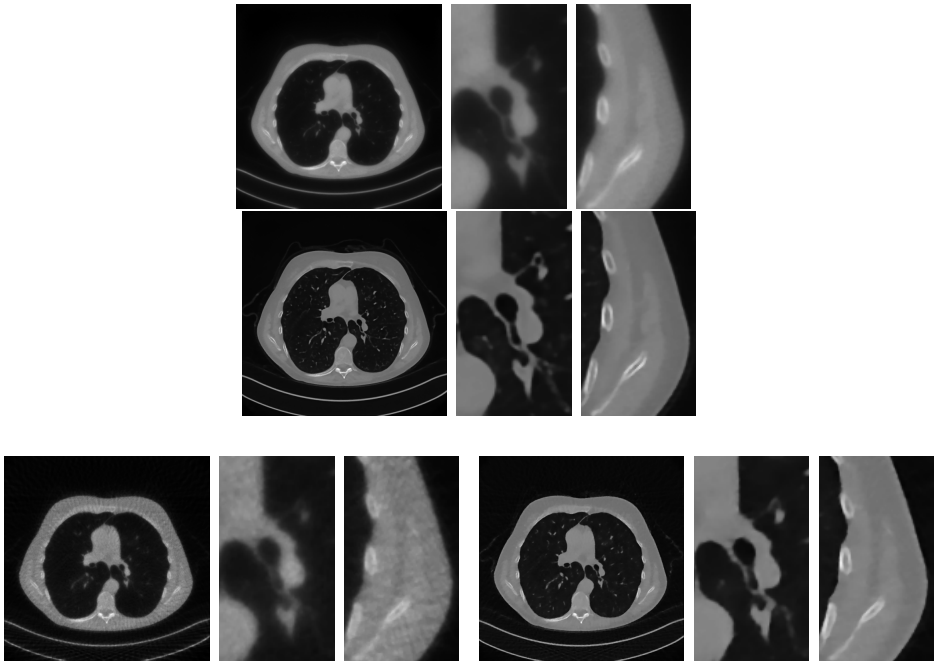


Figure 3.5: Results on a test image from the Mayo data set, under the $P_{360,360}$ (upper row) and the $P_{180,60}$ (bottom row) CT protocol. Left: $\boldsymbol{x}_{RIS}$; right: $\boldsymbol{x}_{RISING}$.

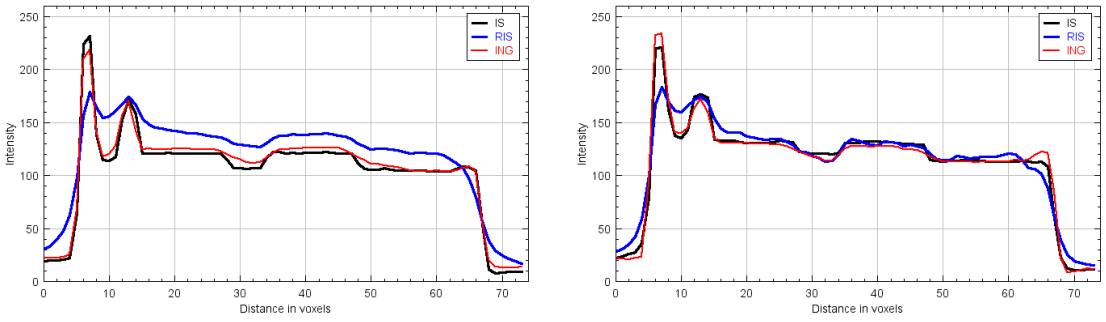


Figure 3.6: Intensity profiles taken on the horizontal red line depicted in 3.2, on the reconstructions in Figure 3.5. The black, blue and red lines are the profile relative to $\boldsymbol{x}_{IS}$, $\boldsymbol{x}_{RIS}$ and $\boldsymbol{x}_{RISING}$, respectively. Left $P_{360,360}$; right: $P_{180,60}$.

Chapter 4

Robust non-convex model-based approach for deep learning-based image processing

Gradient sparsity is a commonly employed tool in imaging applications, such as image reconstruction and image processing. In particular, sparse gradients play a crucial role in compressed sensing, enabling the reconstruction of high-quality images from a limited amount of data. Forcing gradient sparsity is extremely useful in fields such as medical imaging, where preserving boundaries of low-contrast objects is essential for accurate diagnosis and treatment. In model-based approaches, the sparsity of a gradient is usually embedded as a prior $\mathcal{R}$ in a minimization problem of the form:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta) + \lambda\mathcal{R}(\boldsymbol{x}) \tag{4.1}$$

where $\boldsymbol{x}$ is an image in $\mathbb{R}^n$, $\boldsymbol{y}^\delta$ is the acquired noisy sinogram, and $\mathcal{J}$ is a data fitting operator.

It is well known that the ℓ_0 quasi-norm, measuring the cardinality of its argument, is the best possible sparsifying function, but its minimization is computationally very expensive. Luckily, the signal recovery is still possible if one substitutes the ℓ_1 norm to the ℓ_0 quasi-norm under suitable hypotheses [35, 99]. The resulting Total Variation (TV) prior (corresponding to the ℓ_1 norm of the image gradient magnitude) has been widely used in image processing for more than two decades [106, 107], above all for tomographic applications [28, 40, 41, 108]. However, it also has certain drawbacks that need to be considered, as it typically over-smoothes regions with fine details or textures.

In order to better approximate the ℓ_0 quasi-norm, the ℓ_p, $0 < p < 1$, the non-convex sparsifying norm can be adopted. For a fixed p, the isotropic Total p-norm Variation (TpV)

of an image $\boldsymbol{x} \in \mathbb{R}^n$ reads as:

$$\mathrm{T}p\mathrm{V}(\boldsymbol{x}) := ||\mathbf{D}\boldsymbol{x}||_p^p = \sum_{i=1}^{n} \Big(\sqrt{(\boldsymbol{D}_h \boldsymbol{x})_i^2 + (\boldsymbol{D}_v \boldsymbol{x})_i^2} \Big)^p \tag{4.2}$$

where $(\mathbf{D}\boldsymbol{x})_i = \sqrt{(\boldsymbol{D}_h \boldsymbol{x})_i^2 + (\boldsymbol{D}_v \boldsymbol{x})_i^2}$ reflects the magnitude of the gradient image and $\boldsymbol{D}_h$ and $\boldsymbol{D}_v$ denote the differences among two adjacent pixels in the horizontal and vertical directions, respectively, as described in Section 1.3.2. TpV has already been successfully employed in a number of imaging applications such as image enhancement [109, 110] and image reconstruction [111, 112, 113]. The drawback of non-convex minimization is the difficulty in solving the optimization problem, admitting possible multiple local minima. Many efforts have been made to propose efficient strategies or adapt the algorithms considered for convex optimization to the non-convex case. In [114], the authors presented a proof of asymptotic convergence of ℓ_p towards ℓ_0 and demonstrated that even if the global minimum cannot be guaranteed in this case, finding a local minimizer can produce an exact reconstruction of sparse signals with many fewer measurements than in case of $p = 1$. Furthermore, the TpV practical interest is limited by the difficulty of the computational stage. Non-convex iterative minimization is often more computationally intensive and time-consuming than the convex one, as it often requires more iterations to converge to a solution and may require solving multiple sub-problems at each iteration. In addition, the dependence of the convergence solution on the starting iterative guess makes the non-convex framework particularly difficult to tackle.

Beyond a shadow of a doubt, a new wave of innovation is fundamentally changing the image-processing sector and is represented by deep learning. While these models can achieve high accuracy when used to enhance images as black boxes (even in a post-processing phase), from an optimization perspective it is not even clear whether or not we are still addressing the original associated and mathematically grounded imaging problem [98] and the reliability of the final solutions must be discussed [6]. The lack of interpretability, or the inability to understand how the model arrived at its decision, has motivated the design of hybrid schemes as described in Section 1.3.3, where deep learning is used to foster the performances of optimization schemes. We briefly recap the two main classes of algorithms as introduced in Section 1.3.3 citing only a few of the several papers on this topic: (*i*) plug-and-play schemes, where a denoising network is embedded in an iterative algorithm as regularizer [68, 115], and (*ii*) unrolling methods, where a deep network architecture implements an optimization solver and learns model parameters [63]. To those, we add the class of multi-step frameworks [7, 116], that also contains the methods described in the previous Chapters, i.e. RISING and StReNN.

To our knowledge, no one has already considered non-convex regularization in the last two hybrid approaches. The only exception is [117] where a non-convex and non-smooth prior is learned for image reconstruction and used in a descent algorithm.

Contributions In this Chapter, we describe a new two-step learnable optimization framework for solving non-convex image reconstruction problems, extending the RISING approach described in the last Chapter to the non-convex optimization problem deriving by the TpV regularizer. Specifically, in the first step, we apply a robust convolutional neural network to input data to provide a coarse image with sparse gradients. We remark that neural networks are generally non-stable with respect to noise when solving an ill-posed inverse problem. We face this problem by applying a stabilizing strategy based on the TpV prior. Then, the output of the network is used as the starting guess of an iterative solver tackling the TpV-regularized imaging task. We denote our proposal as (TpV)^2, i.e. TpV-squared, since we force the TpV prior in both the two steps. We remark that we fully preserve convergence properties and the mathematical characterization of the final solution, despite the use of a neural network. In addition, we have performed numerical experiments for tomographic imaging, and the results demonstrate that our (TpV)^2 achieves good solutions but also inherits robustness from the data-driven step outperforming iterative non-convex algorithms.

This Chapter is based on my publication [12]. My personal contributions to the cited works were the implementation of the code to test the experiments, partial development of the method, and setting up the experiments. The code to replicate the experiments can be found at: `https://github.com/devangelista2/TpV_RISING`.

Structure of the chapter The Chapter is organized as follows. In Section 4.1 we describe the proposed (TpV)^2 approach. In Section 4.2 we numerically test the proposed method on a synthetic dataset, while in Section 4.3 we describe possible extensions and future works related to this topic. At last, final conclusions are drawn in Section 4.4.

4.1 The T_pV approach

In this paper, we consider imaging tasks stated as under-determined linear inverse problems of forms:

$$\boldsymbol{y}^\delta = \boldsymbol{K}\boldsymbol{x} + \boldsymbol{e}, \quad \boldsymbol{e} \sim \mathcal{N}(0, \sigma^2 I) \tag{4.3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the grey-scale image we want to reconstruct from data $\boldsymbol{y}^\delta \in \mathbb{R}^m$ affected by white gaussian noise $\boldsymbol{e}$ of $\sigma^2 I$ variance (being $I \in \mathbb{R}^{m\times m}$ the identity matrix), $\boldsymbol{K} \in \mathbb{R}^{m\times n}$ is the linear forward imaging operator, and $n > m$. Exemplar imaging applications are super-resolution, where $\boldsymbol{K}$ denotes the downsampling operator and $\boldsymbol{y}^\delta$ the low-resolution image, and tomographic image reconstruction from sparse measurements, where $\boldsymbol{K}$ models the X-ray projections and $\boldsymbol{y}^\delta$ represents the sinogram data. Due also to the typical ill-posedness of the inverse problems, it is convenient to introduce a regularization item and reformulate the imaging task as the regularized optimization problem (4.1). In this work, we consider

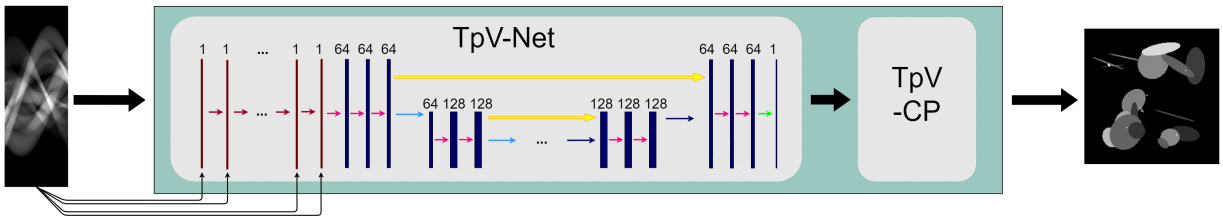


Figure 4.1: Graphical representation of the proposed (TpV)^2 procedure, applied to tomographic image reconstruction.

the following problem statement:

$$\min_{\boldsymbol{x}} \left(\delta_{B_2(\epsilon)}(\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta) + \lambda ||\mathbf{D}\boldsymbol{x}||_p^p \right) \tag{4.4}$$

where $\lambda > 0$ is the regularization parameter and the data fitting is expressed through the indicator function $\delta_{B_2(\epsilon)}$ of the 2-norm ball $B_2(\epsilon)$, which is defined as $B_2(\epsilon) = \{\boldsymbol{x} \in \mathbb{R}^n : ||\boldsymbol{x}||_2 \leq \epsilon\}$ for a suitable $\epsilon > 0$.

In this Section, we illustrate our (TpV)^2 procedure to solve the TpV-regularized optimization problem (4.4). It is also graphically represented in Figure 4.1, for tomographic reconstruction from sub-sampled sinograms. As anticipated, it is designed in two steps, and the TpV gradient sparsity prior plays an important role in both.

4.1.1 The TpV Chambolle-Pock algorithm

To solve the TpV-regularized problem (4.4), we use the reweighted Chambolle-Pock (CP) algorithm illustrated in [111]. The CP algorithm has been first proposed for image processing applications [42] and then efficiently used for convex TV-based image reconstruction from sub-sampled tomographic data [118, 28].
Exploiting the reweighting strategy illustrated in [119], the CP has been applied to a reformulation of the TpV non-convex minimization into a convex weighted TV problem. Indeed, we can alter the objective function in (4.4) as:

$$\min_{\boldsymbol{x}} \left(\delta_{B_2(\epsilon)}(\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta) + \lambda || \ \mathbf{w} \odot \mathbf{D}\boldsymbol{x} \ ||_1 \right) \tag{4.5}$$

introducing the element-wise product with the weights $\mathbf{w} \in \mathbb{R}^n$, defined by:

$$\mathbf{w} = \left(\frac{\sqrt{\eta^2 + |\boldsymbol{D}\boldsymbol{x}|^2}}{\eta} \right)^{p-1}$$

with a predefined $\eta > 0$. It is now possible to address problem (4.5) with a slightly modified version of the CP algorithm, which we will denote in the following as TpV-CP. A pseudocode of the TpV-CP algorithm is reported on Algorithm 7. We recall it is an iterative procedure solving the primal-dual problem up to a convergence criterium [111]. Typically, hundreds of iterations are required, hence the computational effort can be significant.

Algorithm 7 Chambolle-Pock T_pV (CP-T_pV)

input: $\boldsymbol{K}, \boldsymbol{y}^\delta, \boldsymbol{x}^{(0)}, \tilde{\boldsymbol{x}}, \lambda, p, \texttt{maxit}, \texttt{tolf}, \texttt{tolx}$

define: $\boldsymbol{M} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{D} \end{bmatrix}$, $\Gamma = ||\boldsymbol{M}||_2$, $\nu = \frac{||\boldsymbol{K}||_2}{||\boldsymbol{D}||_2}$, $\eta = 2 \cdot 10^{-3}$

initialize: $\tau = \sigma = \Gamma^{-1}$, $\theta \in [0, 1]$

initialize: primal variables $\overline{\boldsymbol{x}}^{(0)} = \boldsymbol{x}^{(0)} \in \mathbb{R}^n$ and dual variables $\boldsymbol{s}^{(0)} \in \mathbb{R}^m$, $\boldsymbol{q}^{(0)} \in \mathbb{R}^{2n}$ as zeros vectors

`% Compute the re-weighting vector`

$\overline{\boldsymbol{w}} = \left(\frac{\sqrt{\eta^2 + |\boldsymbol{D}\tilde{\boldsymbol{x}}|^2}}{\eta} \right)^{p-1}$, $\boldsymbol{w} = \begin{bmatrix} \overline{\boldsymbol{w}} \\ \overline{\boldsymbol{w}} \end{bmatrix}$

for $k <$ `maxit` **do**

`% Update the dual variables`

$\boldsymbol{q}^{(k+1)} = \frac{\boldsymbol{q}^{(k)} + \sigma(\boldsymbol{K}\overline{\boldsymbol{x}} - \boldsymbol{y}^\delta)}{1 + \sigma\lambda}$

$\overline{\boldsymbol{s}}^{(k+1)} = \boldsymbol{s} + \sigma \boldsymbol{D}\overline{\boldsymbol{x}}$

$\boldsymbol{s}^{(k+1)} = \frac{\lambda \boldsymbol{w} \odot \overline{\boldsymbol{s}}^{(k+1)}}{\max\{\lambda \boldsymbol{w}, |\overline{\boldsymbol{s}}^{(k+1)}|}\}$

`% Update primal variable and project to the positive axis`

$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \tau(\boldsymbol{K}^T \boldsymbol{q}^{(k+1)} + \nu \boldsymbol{D}^T \boldsymbol{s}^{(k+1)})$

$\boldsymbol{x}^{(k+1)} = \mathcal{P}_+(\boldsymbol{x}^{(k+1)})$

$\overline{\boldsymbol{x}}^{(k+1)} = \boldsymbol{x}^{(k+1)} + \theta(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)})$

`% Stopping criteria`

if $\frac{||\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}||_2}{||\boldsymbol{x}^{(k)}||_2 + 10^{-6}} <$ `tolx` **or** $\frac{||\boldsymbol{K}\boldsymbol{x}^{(k)} - \boldsymbol{y}^\delta||_2}{\sqrt{m}||\boldsymbol{y}^\delta||_\infty} <$ `tolf` **then**

return $\boldsymbol{x}^{(k+1)}$

end if

end for

return $\boldsymbol{x}^{(k+1)}$

4.1.2 The T*p*V-Net preprocessing

As the numerical solution of a non-convex problem depends on the initial guess $\boldsymbol{x}^{(0)}$ initialized in the iterative solver, we want to find a good starting point effortless. This would also possibly accelerate the convergence, reducing the number of iterations to achieve a good minimum. In this regard, we believe neural networks can be of help. In fact, once pre-trained, the forward processing of a NN is characterized by a very short computational time. For this reason, before applying the TpV-CP algorithm, we use a network to preprocess the data $\boldsymbol{y}^\delta$ and obtain a good $\boldsymbol{x}^{(0)}$.

Precisely, we have designed an architecture that takes inspiration from a classical residual UNet, similar to [7]. In the case of $n > m$, it is known that a fully convolutional neural network is unable to recover the n-dimensional solution $\boldsymbol{x}$ of an inverse problem from the m-dimensional data, due to the lack of translation-equivariance. Hence, we transform the

input $\boldsymbol{y}^\delta$ into a n-dimensional image (approximating the output) by applying a sequence of M frozen, untrained layers. Such layers act as the first M iterations of an iterative method, solving the TpV imaging problem from a pre-fixed starting guess $\boldsymbol{x}^{(0)}$. We always set $\boldsymbol{x}^{(0)}$ as the zero constant image. In this scenario, we opt for the TpV-CP algorithm as the underlying solver. As visible in the graphical draft of Figure 4.1, the input $\boldsymbol{y}^\delta$ is fed to each, subsequent M layers by adding skip-connections. The result of this procedure is then processed by a standard U-Net architecture as described in Section 1.3.3, to efficiently produce an approximation of the target image. Because of the initial frozen layers, our network embeds the TpV prior strongly, hence we denote it as TpV-Net.
We highlight that the U-Net structure needs supervised training. We can use ground truth images, if available. Alternatively, we can use the images generated by the TpV-CP at convergence, applying it to a set of available data $\{\boldsymbol{y}_k^\delta\}_k$ to create a specific training set. In this case, the network can properly learn the sparsifying pattern of the TpV-CP solutions and replicate it onto its outputs.

4.2 Numerical Results

In this Section, we report and discuss results achieved with the proposed (TpV)^2. It has been applied to reconstruct images from synthetic tomographic measurements. We have used the COULE dataset described in Section 1.4.1, whose 256×256 gray-level images contain low- and high-contrast objects, such as overlapping ellipses of uniform intensities, and dots.

Our tomographic simulation considers $n_\alpha = 180$ scans in a 180-degree angular range. The detector has $n_d = 358$ recording units. The additive noise $\boldsymbol{e}$ has a noise level of 0.003.

In the imaging model, we set $p = 0.5$ and $\lambda = 0.006$ as constants, whereas $\epsilon = \epsilon_0\sqrt{m}||\boldsymbol{y}^\delta||_\infty$, with $\epsilon_0 = 0.001$ differs for every image. In the TpV-Net, we set $M = 10$. The network has been trained to minimize the mean squared error between the prediction of the network and the corresponding label, with Adam optimizer and a step size of 0.001, for a total of 50 epochs. We always use the null image as the starting iterate of the TpV-CP solver, whereas we stopped it at the k-th iteration if $||x^{(k)} - x^{(k-1)}||_2 < 10^{-3}$. Indicatively, it takes 180-300 iterations to achieve precise (numerical) solutions, in this setting.

In Table 4.1, we consider a couple of significant test images that have been processed with the same setup but exhibit different results. In the table, we report the root mean square error (RMSE) as described in Section 1.4.2 and the normalized TV value $nTV = \frac{1}{n}\sum_{i=1}^{2n}(\mathbf{D}\boldsymbol{x})_i$ to compare the Ground Truth (GT) images to the convergence solutions. In the case of test image number 10, the TpV-CP convergence image has the smallest error, and all the nTV values are close to the corresponding GT one. Independently from the training settings, the TpV-Net images already have sparse gradients but the second step of the (TpV)^2 is necessary to strongly reduce the RMSE. As visible in Figure 4.2, when we

				Training to GT		Training to TpV-CP	
	Image	GT	TpV-CP	TpV-Net	$(\mathrm{T}p\mathrm{V})^2$	TpV-Net	$(\mathrm{T}p\mathrm{V})^2$
RMSE	10	-	0.0082	0.0649	0.0127	0.0654	0.0117
	14	-	0.0212	0.0461	0.0081	0.0493	0.0080
nTV	10	0.0136	0.0148	0.0175	0.0150	0.0181	0.0148
	14	0.0170	0.0262	0.0197	0.0179	0.0204	0.0180

Table 4.1: Quantitative metrics for image quality assessment, on test images.

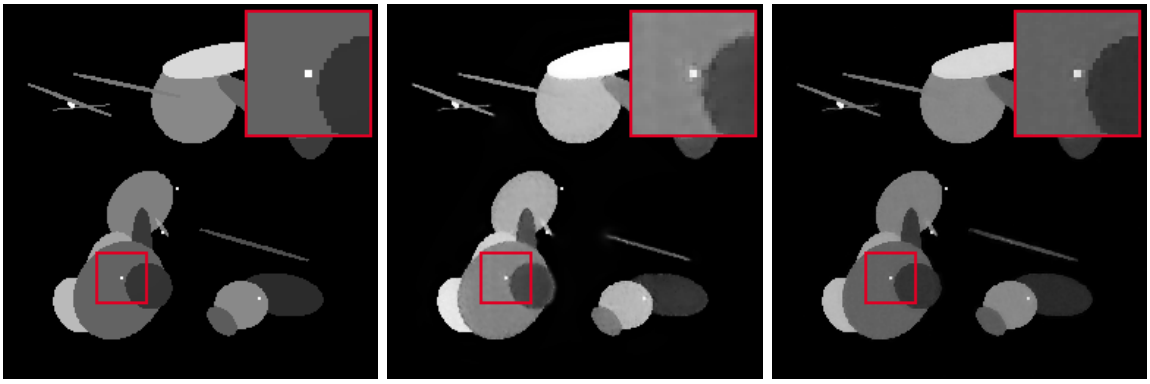


Figure 4.2: Results on the test image number 10, by the $(\mathrm{T}p\mathrm{V})^2$ method trained with convergence images as target. *Left:* GT; *Center:* TpV-Net; *Right:* $(\mathrm{T}p\mathrm{V})^2$. The zoomed red square highlights a region of interest.

train our network with the TpV-CP images as target (thus avoiding GT images at all), the mere TpV-Net method already provides a good image with thin details. However, the final solution is much closer to the GT image, with almost perfect edges and great uniformity. We highlight that it has been achieved in only 33 iterations of the TpV-CP algorithm, hence the preprocessing has been effective in finding a good starting iterate.

Differently, the 14-th test image has been poorly reconstructed by the TpV-CP procedure, in 470 iterations: the image appears noisy in Figure 4.3, its RMSE is higher than 2% and its nTV reflects a high gradient magnitude. On the contrary, the $(\mathrm{T}p\mathrm{V})^2$ is not affected by the instability issue of non-convex optimization, as the two reported images look homogeneous and precise, and they only required about 65 iterations of TpV-CP in both cases. Also the quality assessment metrics confirm this observation.

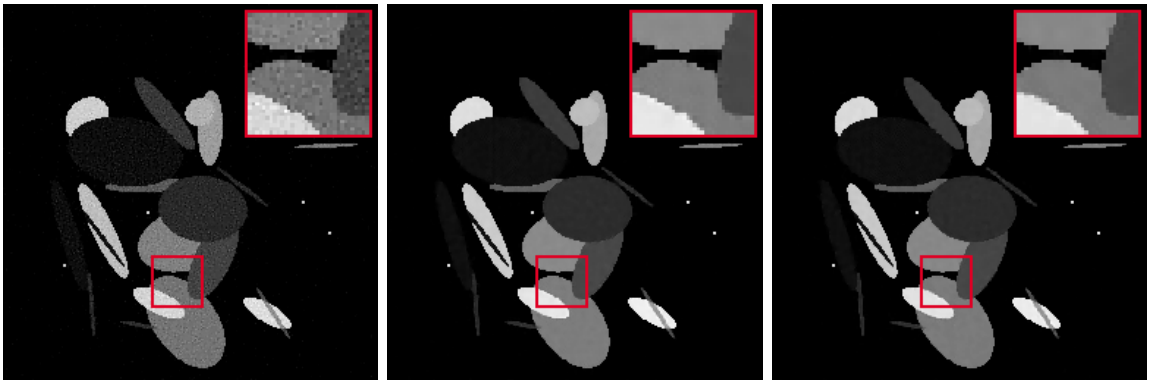


Figure 4.3: Results on the test image number 14. *Left:* TpV-CP at convergence; *Center:* (TpV)^2 trained with convergence images as target; *Right:* (TpV)^2 trained with GT images as target. The zoomed red square highlights a region of interest.

4.3 Extensions and Future Works

The (TpV)^2 approach is a preliminary idea that can be further extended to a general method that can be used to reliably find a stationary point to the ℓ_0-norm optimization problem:

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta) + \lambda||\boldsymbol{D}\boldsymbol{x}||_0. \tag{4.6}$$

To this aim, consider a generic TpV optimization problem, where the sparsity parameter p and the regularization parameter λ are functions of a non-negative variable $h \in \mathbb{N}$. Define:

$$\boldsymbol{x}^{(h)} \in \arg\min_{\boldsymbol{x}\geq\mathbf{0}} \mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y}^\delta) + \lambda(h)||\boldsymbol{D}\boldsymbol{x}||_{p(h)}. \tag{4.7}$$

Note that, if $p(h) \to 0$ as $h \to \infty$, then the sequence $\boldsymbol{x}^{(h)}$ converges to a stationary point of (4.6) as $h \to \infty$. Let $f_h(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}})$ be an iterative algorithm solving the non-convex optimization problem (4.7) such as TpV-CP, starting from the initial guess $\tilde{\boldsymbol{x}}$. The class of Iterated methods consider the sequence $\boldsymbol{x}^{(h)}$ defined as:

$$\boldsymbol{x}^{(h+1)} = f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)}). \tag{4.8}$$

It is possible to show [120] that if $p(h)$ and $\lambda(h)$ satisfy some simple conditions, then the iteration $\boldsymbol{x}^{(h)}$ converges to a stationary point of the limiting problem (4.6).

In an extension of [12], we consider an iterated version of the TpV-CP algorithm where the sparsity parameter $p(h)$ and the regularization parameter $\lambda(h)$ are automatically updated with the updating rule in [120] that guarantees convergence to a local minima of (4.6). The resulting method, presented in detail in Algorithm 8, shows promising results in preliminary experiments.

We can then consider a sequence $\{f_h^\theta(\boldsymbol{y}^\delta), \cdot\}_{h\in\mathbb{N}}$ of neural networks, each representing a TpV-Net model solving the optimization problem (4.7), and compute

$$\tilde{\boldsymbol{x}}^{(h+1)} = f_h^\theta(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(h)}) \tag{4.9}$$

Algorithm 8 Iterated Chambolle-Pock TpV (iTpV-CP)

input: $\boldsymbol{K}, \boldsymbol{y}^\delta, \lambda(0) > 0, \epsilon(0) > 0, \alpha \in (0,1), H \in \mathbb{N}$
initialize: $\boldsymbol{x}^{(0)} = \boldsymbol{0}$, $p(0) = 1$
for $h < H$ **do**
compute:

$$\boldsymbol{x}^{(h+1)} = f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})$$

using Algorithm 7 with starting point and $\tilde{\boldsymbol{x}}$ equal to $\boldsymbol{x}^{(h)}$, parameters $\lambda(h)$, $p(h)$ and `maxit` $= k(h)$
compute: the value of the objective function

$$\mathcal{F}^{[h]} = \frac{1}{2}||\boldsymbol{K}\boldsymbol{x}^{(h)} - \boldsymbol{y}^\delta||_2^2 + \lambda(h)||\boldsymbol{w} \odot |\boldsymbol{D}\boldsymbol{x}^{(h)}|\,||_1$$

`% Update parameters`
update: $\lambda(h+1)$ by:
if $h = 0$ **then**
$\lambda(h+1) = \lambda(h)/2$
else
$\lambda(h+1) = \lambda(h) \cdot \frac{\mathcal{F}^{[h]}}{\mathcal{F}^{[h-1]}}$
end if
update: $p(h+1) = p(h) \cdot \alpha$
end for
return $x^{(H)}$

with $\tilde{\boldsymbol{x}}^{(0)} = 0$. As a result, the sequence $\{\tilde{\boldsymbol{x}}^{(h)}\}_{h\in\mathbb{N}}$ will approximately converge to a stationary point of (4.6). In particular, if we define the training error at step h as

$$\rho_h = \sup_{\substack{\boldsymbol{y}^\delta = \boldsymbol{K}\boldsymbol{x}^\dagger + \boldsymbol{e} \\ \boldsymbol{x}^\dagger \in \mathcal{X}}} ||f_h^\theta(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)}) - f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})||_2, \tag{4.10}$$

and the Lipschitz constant of $f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})$ with respect to $\boldsymbol{x}^{(h)}$ as

$$L_h = \sup_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n} \frac{||f_h(\boldsymbol{y}^\delta, \boldsymbol{x}_1) - f_h(\boldsymbol{y}^\delta, \boldsymbol{x}_2)||_2}{||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2}, \tag{4.11}$$

it holds:

Theorem 4.1. *For any $h > 0$, let $\{f_h^\theta(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})\}_{h\in\mathbb{N}}$ be a sequence of neural networks approximating the iterated Chambolle-Pock algorithm $f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})$. If ρ_h is the training error at step h and L_h is the Lipschitz constant of $f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})$ with respect to $\boldsymbol{x}^{(h)}$, then if $\boldsymbol{x}^{(0)} = \tilde{\boldsymbol{x}}^{(0)}$,*

$$||\tilde{\boldsymbol{x}}^{(h)} - \boldsymbol{x}^{(h)}||_2 \leq \sum_{i=0}^{h-1} \rho_i \prod_{j=i+1}^{h-1} L_j, \tag{4.12}$$

where the product is defined to be 1 for $i = h - 1$.

Proof. We proceed by induction over $h > 0$. For $h = 1$, it holds,

$$\begin{aligned} ||\tilde{\boldsymbol{x}}^{(1)} - \boldsymbol{x}^{(1)}||_2 &= ||f_0^\theta(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(0)}) - f_0(\boldsymbol{y}^\delta, \boldsymbol{x}^{(0)})||_2 \\ &= ||f_0^\theta(\boldsymbol{y}^\delta, \boldsymbol{x}^{(0)}) - f_0(\boldsymbol{y}^\delta, \boldsymbol{x}^{(0)})||_2 \leq \rho_0. \end{aligned} \tag{4.13}$$

Similarly, for $h = 2$,

$$\begin{aligned} ||\tilde{\boldsymbol{x}}^{(2)} - \boldsymbol{x}^{(2)}||_2 &= ||f_1^\theta(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(1)}) - f_1(\boldsymbol{y}^\delta, \boldsymbol{x}^{(1)})||_2 \\ &\leq \underbrace{||f_1^\theta(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(1)}) - f_1(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(1)})||_2}_{\leq \rho_1} + \underbrace{||f_1(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(1)}) - f_1(\boldsymbol{y}^\delta, \boldsymbol{x}^{(1)})||_2}_{\leq L_1 ||\tilde{\boldsymbol{x}}^{(1)} - \boldsymbol{x}^{(1)}||_2} \\ &\leq \rho_1 + L_1 ||\tilde{\boldsymbol{x}}^{(1)} - \boldsymbol{x}^{(1)}||_2 \leq \rho_1 + L_1 \rho_0. \end{aligned} \tag{4.14}$$

Now, we assume that (4.12) holds for h and we want to show that it holds for $h + 1$.

$$\begin{aligned} ||\tilde{\boldsymbol{x}}^{(h+1)} - \boldsymbol{x}^{(h+1)}||_2 &= ||f_h^\theta(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(h)}) - f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})||_2 \\ &\leq ||f_h^\theta(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(h)}) - f_h(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(h)})||_2 + ||f_h(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(h)}) - f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})||_2 \end{aligned} \tag{4.15}$$

but $||f_h^\theta(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(h)}) - f_h(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(h)})||_2 \leq \rho_h$ by definition, while

$$||f_h(\boldsymbol{y}^\delta, \tilde{\boldsymbol{x}}^{(h)}) - f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})||_2 \leq L_h ||\tilde{\boldsymbol{x}}^{(h)} - \boldsymbol{x}^{(h)}||_2 \leq L_h \sum_{i=0}^{h-1} \rho_i \prod_{j=i+1}^{h-1} L_j \tag{4.16}$$

by inductive hypothesis. Consequently,

$$\begin{aligned} ||\tilde{\boldsymbol{x}}^{(h+1)} - \boldsymbol{x}^{(h+1)}||_2 &\leq \rho_h + L_h \sum_{i=0}^{h-1} \rho_i \prod_{j=i+1}^{h-1} L_j = \rho_h + \sum_{i=0}^{h-1} \rho_i \prod_{j=i+1}^{h} L_j \\ &= \rho_h \prod_{j=h+1}^{h} L_j + \sum_{i=0}^{h-1} \rho_i \prod_{j=i+1}^{h} L_j = \sum_{i=0}^{h} \rho_i \prod_{j=i+1}^{h} L_j, \end{aligned} \tag{4.17}$$

which proves the Theorem. □

A deeper understanding on the estimate of the Theorem above can be obtained by assuming both $\{\rho_h\}_h$ and $\{L_h\}_h$ to be limited. Note that this is not a restrictive condition in practice since $\{\rho_h\}_h$ is intuitively bounded by the training, while $\{L_h\}_h$ approaches 0 as $h \to \infty$, since it is known the in a converging iterative algorithm, the last iterations are closer to the identity. When this is the case, let $\rho = \sup_h \rho_h$ and $L = \sup_h L_h$. Then,

$$||\tilde{\boldsymbol{x}}^{(h)} - \boldsymbol{x}^{(h)}||_2 \leq \sum_{i=0}^{h-1} \rho_i \prod_{j=i+1}^{h-1} L_j \leq \sum_{i=0}^{h-1} \rho \prod_{j=i+1}^{h-1} L = \rho \sum_{i=0}^{h-1} L^i \tag{4.18}$$

which is the sum of a geometric sequence. Thus,

$$||\tilde{\boldsymbol{x}}^{(h)} - \boldsymbol{x}^{(h)}||_2 \leq \rho \sum_{i=0}^{h-1} L^i = \rho \frac{L^h - 1}{L - 1} \to \frac{\rho}{1 - L}, \tag{4.19}$$

if $L < 1$. On the other side, if $L > 1$, the above relationship is uninformative since the right-hand side diverges. An alternative estimate can be obtained by considering the following Corollary of the above Theorem.

Corollary 4.2. *Under the same assumptions of Theorem 4.1, if L_h^θ is the Lipschitz constant of $f_h^\theta(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})$ with respect to $\boldsymbol{x}^{(h)}$, it holds*

$$||\tilde{\boldsymbol{x}}^{(h)} - \boldsymbol{x}^{(h)}||_2 \leq \sum_{i=0}^{h-1} \rho_i \prod_{j=i+1}^{h-1} L_j^\theta, \tag{4.20}$$

where the product is defined to be 1 for $i = h - 1$.

Proof. Same as the proof of Theorem 4.1, by changing the roles of $f_h(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})$ and $f_h^\theta(\boldsymbol{y}^\delta, \boldsymbol{x}^{(h)})$. □

An interesting consequence of the Corollary above is the following: if each step of the iterative Chambolle-Pock algorithm has Lipschitz constant which is bounded by a constant $L < 1$, then training a network that minimizes the training error ρ leads to a solution whose distance to the optimum is bounded by $\frac{\rho}{1-L}$, which is small if the network is expressive enough. If instead, the Lipschitz constant of the iterative Chambolle-Pock algorithm is greater than 1, then it is recommended to constraint the network architecture such that $L_h^\theta < 1$ for any h, which can be obtained e.g. by techniques such as spectral normalization or gradient penalty. Indeed, if the supremum L^θ of L_h^θ is bounded by 1, then by the Corollary 4.2,

$$||\tilde{\boldsymbol{x}}^{(h)} - \boldsymbol{x}^{(h)}||_2 \leq \rho \sum_{i=0}^{h-1} (L^\theta)^i = \rho \frac{(L^\theta)^h - 1}{L^\theta - 1} \to \frac{\rho}{1 - L^\theta}. \tag{4.21}$$

Clearly, by constraining the network architecture, the training error ρ will be bigger, but the insight above shows that this will cause just a linear increase to the error of the model, while the increase would be exponential if L^θ would be greater than 1. Note that this is coherent with the results obtained in [8, 10], where limiting the Lipscitz constant to be lower than 1 was a sufficient condition that implies stability.

Numerical results showing the effectiveness of this method and a complete understanding of the way it works are still a work in progress.

4.4 Conclusion

Deep learning and optimization are two different approaches to solve imaging tasks, each with its own strengths and weaknesses. We have proposed an algorithm exploiting all their strengths, to solve a non-convex minimization problem with a data-driven speed-up enforcing the convergence to a reliable minimum. Results achieved on tomographic reconstructions

have confirmed the effectiveness of our (TpV)^2 algorithm, outperforming the robustness of mere iterative solvers for non-convex optimization, with respect to noise and algorithmic parameters.

Part II

Deep Generative Models for image generation

Chapter 5

A probabilistic approach to imaging: generative models in a GreenAI perspective

So far, we've dealt with a scenario in which the variables $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$, representing the unknown image and the corresponding data, are deterministic, and the operator mapping one to the other is also deterministic. In this context, the only stochastic element is introduced by the noise, which affects the mathematical representation of the fidelity term $\mathcal{J}(\boldsymbol{K}\boldsymbol{x}, \boldsymbol{y})$ in the optimization problem discussed in Section 1.3.2. In this Chapter, which introduces the second part of my thesis, we intend to change the perspective and shift to a probabilistic approach. In this new framework, we consider $\mathcal{X}$ and $\mathcal{Y}$ as sub-manifolds of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. More specifically, we assume that two probability distributions, denoted as μ_{gt} and μ_Y, exist in such a way that $\mathcal{X} = supp(\mu_{gt})$ and $\mathcal{Y} = supp(\mu_Y)$.

To simplify the computation, we make the assumption that both μ_{gt} and μ_Y are absolutely continuous. This implies the existence of probability density functions $p_{gt}(\boldsymbol{x})$ and $p_Y(\boldsymbol{y})$, such that for any measurable set $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^m$, it holds:

$$\begin{aligned} \mu_{gt}(A) &= \int_A p_{gt}(\boldsymbol{x}) d\boldsymbol{x}, \\ \mu_Y(B) &= \int_B p_Y(\boldsymbol{y}) d\boldsymbol{y}. \end{aligned} \tag{5.1}$$

The introduction of probability density functions allows us to reframe the image reconstruction process in terms of the conditional probabilities of $\boldsymbol{x}$ and $\boldsymbol{y}$.

In particular, consider the probabilistic version of the inverse problem introduced in Section 1.3, i.e. given a matrix $\boldsymbol{K} \in \mathbb{R}^{m \times n}$, a noise realization $\boldsymbol{e} \in \mathbb{R}^m$ and

$$\boldsymbol{y}^\delta = \boldsymbol{K}\boldsymbol{x}^{gt} + \boldsymbol{e}, \qquad \boldsymbol{x}^{gt} \sim p_{gt}(\boldsymbol{x}), \tag{5.2}$$

recover an approximation of $\boldsymbol{x}^{gt}$. To solve this problem, the probabilistic counterpart of the direct methods introduced in Section 1.3.1 is the so-called *Maximum Likelihood Estimation (MLE)*. The idea of MLE is that, given the observed data $\boldsymbol{y}^\delta$ and the corruption process $\boldsymbol{K}$, the reconstruction $\boldsymbol{x}^*$ is defined as the value that maximizes the conditional probability of observing $\boldsymbol{y}^\delta$ given $\boldsymbol{x}$. In formula:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{y}|\boldsymbol{x}), \tag{5.3}$$

where the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ is commonly referred to as *likelihood*. For example, when the noise $\boldsymbol{e}$ is a sample from a zero-mean Gaussian distribution, i.e. $\boldsymbol{e} \sim \mathcal{N}(0, \sigma^2 I)$, then

$$p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{K}\boldsymbol{x}, \sigma^2 I), \tag{5.4}$$

and an MLE solution can be computed by solving the optimization problem obtained by taking the negative logarithm of the likelihood distribution in (5.3):

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} -\log p(\boldsymbol{y}|\boldsymbol{x}) = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{2\sigma^2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta||_2^2. \tag{5.5}$$

Clearly, when $\boldsymbol{K}$ is ill-conditioned, the MLE suffers from the same instability observed in the direct solution approach to inverse problems. A solution is to consider the *Maximum A-Posteriori (MAP)* approach, where the a-posteriori distribution $p(\boldsymbol{x}|\boldsymbol{y})$ is maximized instead of the likelihood. As $p(\boldsymbol{x}|\boldsymbol{y})$ is generally unknown, it is common practice to estimate it by employing Bayes' Theorem:

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})p_{gt}(\boldsymbol{x})}{p(\boldsymbol{y})}, \tag{5.6}$$

whose maximum can be obtained by taking the negative logarithm, similar to what was done in (5.5). The resulting optimization problem can be expressed as follows:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} -\log p(\boldsymbol{x}|\boldsymbol{y}) = \arg\min_{\boldsymbol{x}\in\mathcal{X}} -\log \underbrace{p(\boldsymbol{y}|\boldsymbol{x})}_{likelihood} - \log \underbrace{p_{gt}(\boldsymbol{x})}_{prior}, \tag{5.7}$$

where the likelihood is dependent on the model $\boldsymbol{K}$, as in (5.5), while the prior distribution $p_{gt}(\boldsymbol{x})$ relies on the structure of $\mathcal{X}$, given that, according to our assumption, $\mathcal{X} = supp(\mu_{gt})$, and $p_{gt}(\boldsymbol{x})$ represents the density of μ_{gt}. Note that (5.7) is similar to the regularized formulation described in Section 1.3.2, where the likelihood represents the fidelity term and is related to the corruption process, while the prior represents the regularaization functional, related with $\mathcal{X}$.

Since the likelihood $p(\boldsymbol{y}|\boldsymbol{x})$ is given when the statistics of the noise are known, a significant challenge is to find an approximation for the unknown distribution $p_{gt}(\boldsymbol{x})$. Part of my PhD research focused on the analysis of neural network techniques to approximate this distribution using a dataset $\mathcal{S} \subseteq \mathcal{X}$ consisting of *independent identically distributed (i.i.d.)* samples from $p_{gt}(\boldsymbol{x})$. These methods, used to approximate $p_{gt}(\boldsymbol{x})$, belong to the category of *Generative Models* [121].

5.1 Generative Models

As already remarked, generative models are mathematical models used to approximate the distribution $p_{gt}(\boldsymbol{x})$ of the data. In particular, we assume that we have access to a set of N independent identically distributed points $\boldsymbol{x}^{(i)} \sim p_{gt}(\boldsymbol{x})$ for $i = 1, \dots, N$. The set of these points is indicated as $\mathcal{S} \subseteq \mathbb{R}^n$ in the following. To find an appropriate approximation for $p_{gt}(\boldsymbol{x})$, we consider a family of probability densities $\mathcal{P}_\Theta = \{p_\Theta : \mathbb{R}^n \to \mathbb{R}; \Theta \in \mathbb{R}^s\}$ and find the optimal distribution $p_{\Theta^*} \in \mathcal{P}_\Theta$ such that $p_\Theta \approx p_{gt}$. To do that, it is common to consider the optimization problem:

$$p_{\Theta^*} = \arg\min_{p_\Theta \in \mathcal{P}_\Theta} D_{KL}(p_{gt}||p_\Theta), \tag{5.8}$$

where $D_{KL}(p_{gt}||p_\Theta)$ is the Kullback-Leibler divergence [122] between p and p_Θ, defined as:

$$D_{KL}(p_{gt}||p_\Theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})} \left[\log p_{gt}(\boldsymbol{x}) - \log p_\Theta(\boldsymbol{x})\right]. \tag{5.9}$$

Note that a solution of (5.8) effectively leads to a distribution $p_{\Theta^*} \approx p_{gt}$, since it is easy to prove [122] that for any distributions p, q,

$$\begin{aligned} &D_{KL}(p||q) \geq 0, \\ &D_{KL}(p||q) = 0 \iff p = q, \end{aligned} \tag{5.10}$$

which implies that the distribution p_{Θ^*} that minimizes $D_{KL}(p_{gt}||p_\Theta)$ is the *closest* element to p_{gt} in $\mathcal{P}_\Theta$.

The optimization problem (5.8) can be further simplified by noting that by the definition of Kullback-Leibler divergence,

$$\begin{aligned} \arg\min_{p_\Theta \in \mathcal{P}_\Theta} D_{KL}(p_{gt}||p_\Theta) &= \arg\min_{p_\Theta \in \mathcal{P}_\Theta} \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})} \left[\log p_{gt}(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})} \left[\log p_\Theta(\boldsymbol{x})\right] \\ &= \arg\min_{p_\Theta \in \mathcal{P}_\Theta} \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})} \left[-\log p_\Theta(\boldsymbol{x})\right], \end{aligned} \tag{5.11}$$

which is the Maximum Likelihood Estimation of the distribution $p_\Theta(\boldsymbol{x})$. For this reason, Generative Models that are trained to minimize the Kullback-Leibler divergence with the unknown distribution p_{gt} are referred to as *MLE-based Generative Models*.

Among the wide variety of MLE-based Generative Models, this thesis focuses on a specific category that has gained popularity in recent years due to its impressive results – the class of *Deep Generative Models (DGM)*. A Generative Model is considered a DGM if $p_\Theta(\boldsymbol{x})$ is parameterized, at least in part, by a neural network. For instance, one can define $p_\Theta(\boldsymbol{x}) = \mathcal{N}(\mu_\Theta(\boldsymbol{z}), \Sigma_\Theta(\boldsymbol{z}))$, where $\boldsymbol{z}$ is a variable containining information about $\boldsymbol{x}$, and μ_Θ and Σ_Θ are the neural networks that parameterize the distribution p_Θ. The specific form of p_Θ and the assumptions used to solve (5.8) characterize the DGM method. Based on this, we can create a taxonomy of the various approaches to DGM.

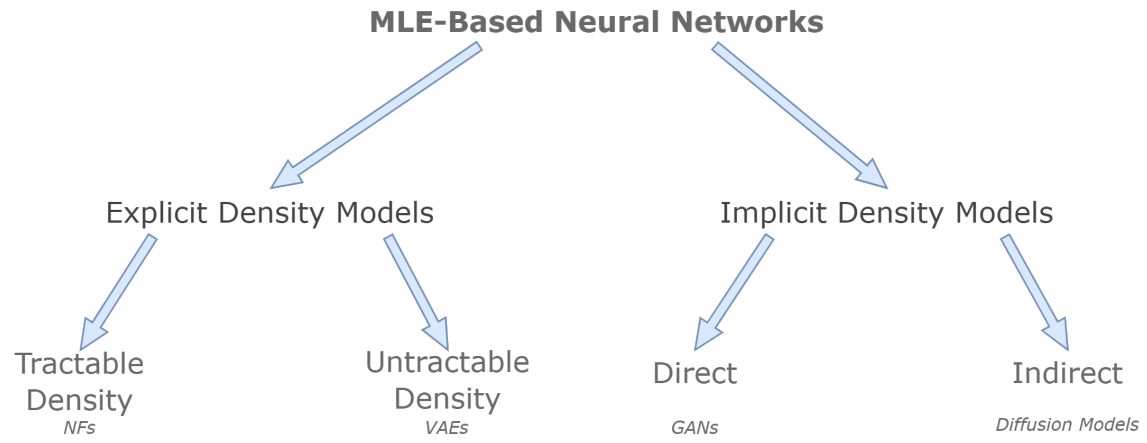


Figure 5.1: A taxonomy of the MLE-based DGMs.

5.1.1 A Taxonomy of Deep Generative Models

Figure 5.1 shows a diagram describing the interrelationship between the main categories in which MLE-based DGM are divided, which takes inspiration from [123]. In particular, at the first level of the diagram, there are the *Explicit Density* and the *Implicit Density* models.

Explicit Density Models In Explicit Density Models, we explicitly define a functional form for the family of distribution $\mathcal{P}_\Theta = \{p_\Theta(\boldsymbol{x}); \Theta \in \mathbb{R}^s\}$ and, given a loss function $\mathcal{L}(\Theta)$, we optimize:

$$\Theta^* = \arg\min_{\Theta \in \mathbb{R}^s} \mathcal{L}(\Theta). \tag{5.12}$$

The resulting distribution is *explicit* in the sense that, after training, it is possible to access $p_{\Theta^*}(\boldsymbol{x})$ directly. It can be used not only to get samples from it but also to get statistics that, if $\mathcal{P}_\Theta$ is expressive enough, are strongly related to the statistics of $p_{gt}(\boldsymbol{x})$.

A drawback of Explicit Density models is that, since the functional form of $\mathcal{P}_\Theta$ has to be fixed a priori, they usually lack in expressiveness, and they perform worse when compared to Implicit Density models, especially if the distribution $p_{gt}(\boldsymbol{x})$ is extremely complex, such as when $p_{gt}(\boldsymbol{x})$ is the distribution of natural high-dimensional images.

In the category of Explicit Density models, we can distinguish two classes of methods. The first one, usually referred to as *tractable* models, assumes that the distribution $p_\Theta(\boldsymbol{x})$ has a tractable form and that $\log p_\Theta(\boldsymbol{x})$ can be computed efficiently. Consequently, the loss function can be defined as:

$$\mathcal{L}(\Theta) = \min_{\Theta \in \mathbb{R}^s} \mathbb{E}_{\boldsymbol{x} \sim p_{gt}(\boldsymbol{x})} \left[-\log p_\Theta(\boldsymbol{x}) \right]. \tag{5.13}$$

We remark that minimizing $\mathcal{L}(\Theta)$ is equivalent to minimize the Kullback-Leibler divergence between $p_{gt}(\boldsymbol{x})$ and $p_\Theta(\boldsymbol{x})$. While the Tractable approach is theoretically the most

desirable, the assumption that $p_\Theta(\boldsymbol{x})$ is tractable proves to be overly restrictive in most high-dimensional scenarios. Consequently, models resulting from this approach lack expressiveness, leading to a learned distribution p_{Θ^*} which is significantly different from $p_{gt}(\boldsymbol{x})$. In this category, the most successful approach, known as Normalizing Flows (NF) [124], excels in low-dimensional settings but struggles to scale effectively to high dimensions, where other types of DGMs outperform them.

To enhance the expressiveness of the resulting distribution, untractable Explicit Density models do not rely on the assumption that $p_\Theta(\boldsymbol{x})$ is tractable. Consequently, the MLE in Equation (5.11) cannot be explicitly computed and has to be approximated. In particular, this class of models employs a tractable upper bound for the MLE estimation as their loss function, i.e.,

$$\mathcal{L}(\Theta) \geq \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})} \left[-\log p_\Theta(\boldsymbol{x}) \right]. \tag{5.14}$$

As a result, a minimizer of $\mathcal{L}(\Theta)$ is approximately a minimizer of $\mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})} \left[-\log p_\Theta(\boldsymbol{x}) \right]$, and the approximation error depends on the *approximation gap*, defined as

$$AG(\Theta) = \mathcal{L}(\Theta) - \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})} \left[-\log p_\Theta(\boldsymbol{x}) \right], \text{ for any } \Theta \in \mathbb{R}^s. \tag{5.15}$$

The representative models in this class are the Variational Autoencoders (VAEs) [4, 125, 126]. Since understanding all the advantages and disadvantages of VAEs is a part of my PhD project, these models will be described in depth in the next Chapter.

Implicit Density Models The Implicit Density models are characterized by the fact that no explicit functional form for $p_\Theta(\boldsymbol{x})$ is provided by the trained network, which is only used to build an algorithm to *sample* from the learned distribution $p_\Theta(\boldsymbol{x})$, which is approximately equal to $p_{gt}(\boldsymbol{x})$. Note that while we could potentially compute an explicit form for the distribution by repeatedly sampling from $p_\Theta(\boldsymbol{x})$, the number of samples required to obtain a sufficiently good approximation is impractical in practice, due to the *curse of dimensionality*.

The different classes of Implicit Density models are distinguished by the way the sample from $p_\Theta(\boldsymbol{x})$ is obtained. In particular, an Implicit Density model is *indirect* if the sample is obtained by an iterative algorithm, such as a Markov chain, designed in a way that $p_{gt}(\boldsymbol{x})$ is the limiting distribution. The principal drawback of Indirect Implicit Density models is that, due to their iterative nature, it is unclear how many iterations are required to reach convergence and consequently, the computation required to obtain a single sample is usually extensive. However, indirect Implicit Density Models have recently gained popularity, primarily due to the success of Diffusion Models [127], which are described in detail in Chapter 7. These models are now considered state-of-the-art in the field of Deep Generative Models due to their ability to generate high-quality samples. Modern image generation software,

such as DALL·E [128], are based on Diffusion Models.

On the other side, an Implicit Density model is *direct* if the sampling is generated directly by the neural network, which is trained to map a random input into a point distributed as $p_{gt}(\boldsymbol{x})$. A representative method in this class is the Generative Adversarial Network (GAN) [123, 129, 130], whose introduction in 2016 marked a turning point in the neural network literature. Indeed, the incredible quality of the generation of GANs showcased the potential of Deep Generative Models, leading to an interest in the field that still continues today.

5.1.2 Deep Latent Variable Models

An interesting aspect regarding Deep Generative Models is their exceptional ability to approximate very high-dimensional distributions $p_{gt}(\boldsymbol{x})$ as it is the case, for example, when $\mathcal{X}$ is the set of natural images. To be able to achieve such a result, a general assumption in DGM literature is that, any *observable* variable $\boldsymbol{x} \in \mathcal{X}$ is associated with a *latent* variable, which cannot be measured directly, containing high-level information on $\boldsymbol{x}$. From now on, let $\boldsymbol{z} \in \mathbb{R}^s$ represent a vector or latent variables. Then,

$$p_\Theta(\boldsymbol{x}) = \int_{\boldsymbol{z}\in\mathcal{Z}} p_\Theta(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}, \tag{5.16}$$

where $\mathcal{Z} \subseteq \mathbb{R}^s$ is the set of admissible latent variables, i.e. the support of the prior distribution $p(\boldsymbol{z})$. Since $\boldsymbol{z}$ is, by assumption, a vector containing information on the observable variables $\boldsymbol{x}$, we can identify two tasks related to $p(\boldsymbol{x}, \boldsymbol{z})$, namely the *inference*, where given $\boldsymbol{x} \sim p_{gt}(\boldsymbol{x})$, one has to find a latent variable $\boldsymbol{z}$ associated with $\boldsymbol{x}$, i.e. $\boldsymbol{z} \sim p(\boldsymbol{z}|\boldsymbol{x})$, and the *generation*, where a sampled $\boldsymbol{z} \sim p(\boldsymbol{z})$ is used to sample $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z})$. This thesis primarily focuses on the second application, where the high-dimensionality of $\boldsymbol{x}$ is addressed by introducing a low-dimensional latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$. Subsequently, $\boldsymbol{x}$ is sampled from a *learned* approximation $p_\Theta(\boldsymbol{x}|\boldsymbol{z})$ of $p(\boldsymbol{x}|\boldsymbol{z})$. Note that this description naturally leads to the decomposition:

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z}), \tag{5.17}$$

from which:

$$p_{gt}(\boldsymbol{x}) = \int_{\boldsymbol{z}\in\mathcal{Z}} p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} = \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[p(\boldsymbol{x}|\boldsymbol{z})\right]. \tag{5.18}$$

This decomposition, together with the expression in Equation (5.16), suggests a simple way to learn a model $p_\Theta(\boldsymbol{x})$ approximating $p_{gt}(\boldsymbol{x})$. Indeed, if we assume $p(\boldsymbol{z})$ is known, as it is commonly done in literature, then

$$p_\Theta(\boldsymbol{x}) \approx p_{gt}(\boldsymbol{x}) \iff p_\Theta(\boldsymbol{x}|\boldsymbol{z}) \approx p(\boldsymbol{x}|\boldsymbol{z}). \tag{5.19}$$

Since in general $p(\boldsymbol{x}|\boldsymbol{z})$ has a simpler form than $p_{gt}(\boldsymbol{x})$, the problem of learning an approximation of $p(\boldsymbol{x}|\boldsymbol{z})$ is better conditioned with respect to the original DGM problem while leading to the same result. Models that are trained to approximate $p(\boldsymbol{x}|\boldsymbol{z})$ are called *Latent Variable Models (LVM)*. In particular, when the distribution $p_\Theta(\boldsymbol{x}|\boldsymbol{z})$ is defined as a neural network, the resulting model is named *Deep Latent Variable Model (DLVM)* [131]. The most representative models of each class of DGM described in the last Section, i.e. the VAEs, GANs, NFs, and Diffusion Models, are DLVMs. In the remaining part of this Section, we will describe in detail the four models, as a reference for the subsequent Chapters.

VAEs A Variational Autoencoder is an Explicit Density Model, where a neural network is trained to explicitly compute the generative distribution $p_\Theta(\boldsymbol{x}|\boldsymbol{z})$, mapping the latent to the observable variables, leading to a distribution $p_\Theta(\boldsymbol{x})$ whose computation relies on the decomposition $p_\Theta(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[p_\Theta(\boldsymbol{x}|\boldsymbol{z})\right]$. In particular, in Variational Autoencoders, we utilize a pair of networks. The first one, known as the *encoder*, is trained to infer latent variables $\boldsymbol{z}$ from an input datum $\boldsymbol{x} \sim p_{gt}(\boldsymbol{x})$. For each input $\boldsymbol{x}$, we establish an encoder distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\phi(\boldsymbol{x}), \sigma^2_\phi(\boldsymbol{x}))$, where the mean $\mu_\phi(\boldsymbol{x})$ and variance $\sigma^2_\phi(\boldsymbol{x})$ are the outputs of a neural network with $\boldsymbol{x}$ as input. We sample a latent variable $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$, and with high probability, it will correspond to a latent variable associated with $\boldsymbol{x}$.

The second network, referred to as the *decoder*, is trained to generate a new sample $\boldsymbol{x} \sim p_\Theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mu_\Theta(\boldsymbol{z}), \sigma^2_{\boldsymbol{x}})$ from the latent variable $\boldsymbol{z}$. Here, $\mu_\Theta(\boldsymbol{z})$ is the output of the neural network with input $\boldsymbol{z}$, and $\sigma^2_{\boldsymbol{x}}$ is a hyperparameter that determines the expressiveness of the resulting model.

These two models are jointly trained as follows: for any sample $\boldsymbol{x} \sim p_{gt}(\boldsymbol{x})$ drawn from $\mathcal{S}$, the encoder network samples a latent variable $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$ associated with $\boldsymbol{x}$. This $\boldsymbol{z}$ is then processed by the second network, which returns a sample $\tilde{\boldsymbol{x}} \sim p_\Theta(\boldsymbol{x}|\boldsymbol{z})$. The loss function is designed to minimize the reconstruction error between the input $\boldsymbol{x}$ and the final output $\tilde{\boldsymbol{x}}$, while also trying to align the distribution of the encoded variables $q_\phi(\boldsymbol{z}) = \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}\left[q_\phi(\boldsymbol{z}|\boldsymbol{x})\right]$ with the prior distribution $p(\boldsymbol{z})$ for $\boldsymbol{z}$. Specifically, the loss function of Variational Autoencoders, commonly referred to as the *Evidence Lower Bound (ELBO)* [125], is defined as:

$$ELBO(\Theta, \phi) = \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}\left[\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[-\log p_\Theta(\boldsymbol{x}|\boldsymbol{z})\right] + D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))\right]. \tag{5.20}$$

It can be shown [125] that $ELBO(\Theta, \phi)$ provides an upper bound for the Maximum Likelihood Estimation (MLE) of $p_\Theta(\boldsymbol{x})$ for any Θ, ϕ, i.e.:

$$ELBO(\Theta, \phi) \geq \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}\left[-\log p_\Theta(\boldsymbol{x})\right], \tag{5.21}$$

implying that the parameters Θ^*, ϕ^* that minimize $ELBO(\Theta, \phi)$ are an approximation of the MLE solution. In practice, $p(\boldsymbol{z})$ is often assumed to be a normal distribution, with $p(\boldsymbol{z}) = \mathcal{N}(0, I)$, which makes optimizing $ELBO(\Theta, \phi)$ computationally efficient.

After the model is trained, new samples $\boldsymbol{x}$ can be generated by first sampling $\boldsymbol{z} \sim p(\boldsymbol{z}) = \mathcal{N}(0, I)$, and then $\boldsymbol{x} \sim p_\Theta(\boldsymbol{x}|\boldsymbol{z})$. For more in-depth details on how Variational Autoencoders operate, refer to the next Chapter, where they are explained in greater detail.

GANs Generative Adversarial Networks are Implicit Density Models, whose working mechanism is similar to that of VAEs. In particular, given a latent variable $\boldsymbol{z}$ sampled from a known distribution $p(\boldsymbol{z})$, usually defined to be $\mathcal{N}(0, I)$, a *generative network* is considered to generate a sample $\boldsymbol{x} = G_\Theta(\boldsymbol{z})$. The quality of the generation is assessed by another network, called *discriminator*, which is trained to distinguish between real images and fake images generated by G_Θ. A generation $\boldsymbol{x} = G_\Theta(\boldsymbol{z})$ is good if it is able to fool the discriminator into being classified as a real image. The two models are jointly trained, so that while the ability of the generator to fool the discriminator increases, the accuracy of the discriminator increases as well, requiring the first model to further improve.

From a mathematical point of view, the learned distribution $p_\Theta(\boldsymbol{x})$ is the push-forward of the latent distribution $p(\boldsymbol{z})$, i.e.

$$p_\Theta(\boldsymbol{x}) = (G_\Theta)_{\#}\, p(\boldsymbol{z}). \tag{5.22}$$

Due to the complexity provided by the mapping G_Θ, the push-forward distribution cannot be explicitly computed, for which the GANs are implicit density models.

NFs Normalizing Flows are Explicit Density Models, whose structure permits an explicit minimization of the Kullback-Leibler divergence between the ground-truth distribution $p_{gt}(\boldsymbol{x})$ and the learned distribution $p_\Theta(\boldsymbol{x})$. In particular, let $\boldsymbol{z} \sim p(\boldsymbol{z}) = \mathcal{N}(0, I)$ be a vector of latent variables, and let $G_\Theta(\boldsymbol{z})$ be a continuous, invertible function mapping the latent variable $\boldsymbol{z}$ to an $\boldsymbol{x} \in \mathcal{X}$. In Normalizing Flows, we consider the change of variable formula for probability distributions to explicitly compute the distribution of the generated $\boldsymbol{x} \sim p_{gt}(\boldsymbol{x})$ from the known distribution $p(\boldsymbol{z})$ of the latent variables, i.e.

$$p_\Theta(\boldsymbol{x}) = p(G_\Theta^{-1}(\boldsymbol{x}))\det\left(\frac{\partial G_\Theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right). \tag{5.23}$$

Consequently, the Kullback-Leibler divergence between $p_{gt}(\boldsymbol{x})$ and $p_\Theta(\boldsymbol{x})$ can be explicitly computed as

$$D_{KL}(p_{gt}(\boldsymbol{x})||p_\Theta(\boldsymbol{x})) \propto -\log p_\Theta(\boldsymbol{x}) = -\log p(G_\Theta^{-1}(\boldsymbol{x})) - \log\det\left(\frac{\partial G_\Theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right). \tag{5.24}$$

To define a model $G_\Theta(\boldsymbol{z})$ which is continuous, invertible, and expressive enough to be able to approximately fit the high-dimensional ground-truth distribution $p_{gt}(\boldsymbol{x})$, in Normalizing Flows we assume that $G_\Theta(\boldsymbol{z})$ is defined by L simple transformations $G_\Theta^1, G_\Theta^2, \ldots, G_\Theta^L$, each

being an invertible neural network, whose composition defines the transformation G_Θ. Note that, if $G_\Theta(\boldsymbol{z}) = (G_\Theta^L \circ G_\Theta^{L-1} \circ \cdots \circ G_\Theta^2 \circ G_\Theta^1)(\boldsymbol{z})$, then its inverse G_Θ^{-1} can be computed as:

$$G_\Theta^{-1}(\boldsymbol{x}) = \left(\left(G_\Theta^{-1}\right)^1 \circ \left(G_\Theta^{-1}\right)^2 \circ \cdots \circ \left(G_\Theta^{-1}\right)^{L-1} \circ \left(G_\Theta^{-1}\right)^L\right)(\boldsymbol{x})\,. \tag{5.25}$$

Given that, (5.24) becomes:

$$-\log p_\Theta(\boldsymbol{x}) = -\log p(G_\Theta^{-1}(\boldsymbol{x})) - \sum_{l=1}^{L} \log\det\left(\frac{\partial\left(G_\Theta^{-1}\right)^l(\boldsymbol{x})}{\partial \boldsymbol{x}}\right), \tag{5.26}$$

which can be easily computed if the derivative of $G_\Theta^l(\boldsymbol{x})$ is simple for any l.

It's important to note that, while the composition of multiple, simpler invertible functions offers flexibility, the constraint that the resulting model $G_\Theta(\boldsymbol{z})$ must be invertible limits its expressiveness. This limitation makes the method impractical for very high-dimensional scenarios, such as when $\mathcal{X}$ represents the set of natural images. Nevertheless, the advantage of being able to explicitly compute the optimal distribution in terms of the Kullback-Leibler divergence makes this method highly effective when the ground-truth distribution $p_{gt}(\boldsymbol{x})$ is relatively simple to approximate, as is the case in low-dimensional generative problems.

Diffusion Models Diffusion Models are Implicit Density Models with Indirect Sampling Method. In particular, let $\boldsymbol{x}_0 \sim p_{gt}(\boldsymbol{x})$ be a sample from the ground-truth distribution, and define a Markov Chain, called *diffusion process*, that increasingly corrupts $\boldsymbol{x}_0$ to pure noise. In particular, let

$$p_{gt}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\sqrt{\beta_t}\boldsymbol{x}_{t-1}, (1-\beta_t)I\right) \tag{5.27}$$

be the transition probability from time $t-1$ to time t, where $\{\beta_t\}_{t\geq 1}$ is a sequence of real numbers in the $(0,1)$ range. It can be shown that,

$$p_{gt}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathbb{E}_{(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_{t-1})\sim p_{gt}(\boldsymbol{x}_{1:t-1})}\left[\prod_{s=1}^{t} p_{gt}(\boldsymbol{x}_s|\boldsymbol{x}_{s-1})\right] = \mathcal{N}\left(\sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)I\right), \tag{5.28}$$

where $\alpha_t = \prod_{s=1}^{t}\beta_t$ for any $t \geq 1$, and $\alpha_t \to 0$ as $t \to \infty$. Note that this implies that, as t increases to infinity, then $p_{gt}(\boldsymbol{x}_t|\boldsymbol{x}_0)$ approaches a normal distribution $\mathcal{N}(0, I)$. If T is sufficiently large so that $\boldsymbol{x}_T$ is close to a normal distribution, then $\boldsymbol{x}_T$ is referred to as *latent variable* in the field of diffusion models, and the stochastic process modeling the transformation from the initial distribution $p_{gt}(\boldsymbol{x}_0)$ to the normal distribution is the inference process.

To define the corresponding generative process, mapping the normally distributed latent variables $\boldsymbol{x}_T$ back to samples $\boldsymbol{x}_0$ from the ground-truth distribution, the Markov Chain $p_{gt}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ can be inverted, and the resulting process $p_{gt}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ can be approximated, step-by-step, by a distribution $p_\Theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ parameterized by a neural network. This is

usually done by training a denoising network $f_\Theta(\boldsymbol{x}_t)$ to approximately recover the image $\boldsymbol{x}_0 \in \mathcal{X}$ such that $\boldsymbol{x}_t \sim p_{gt}(\boldsymbol{x}_t|\boldsymbol{x}_0)$. Then, the distribution $p_\Theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ can be defined as $p_\Theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = p_{gt}(\boldsymbol{x}_{t-1}|\tilde{\boldsymbol{x}}_0)$, where $\tilde{\boldsymbol{x}}_0 = f_\Theta(\boldsymbol{x}_t)$ is the denoised version of $\boldsymbol{x}_t$ by the network f_Θ.

After training, a new sample is generated by sampling $\boldsymbol{x}_T \sim \mathcal{N}(0, I)$ and then by iteratively compute $\boldsymbol{x}_{t-1} \sim p_\Theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ for $t = T, \dots, 1$. We remark that since the new sample $\boldsymbol{x}_0$ is obtained by a Markov Chain, the learned distribution $p_\Theta(\boldsymbol{x}_0)$ cannot be explicitly computed. However, the quality of samples generated by Diffusion Models is remarkable. Moreover, modern Diffusion Model algorithms allow the parameter T that determines how close $\boldsymbol{x}_T$ is to a Gaussian distribution to be chosen relatively small. This makes Diffusion Models a viable option for complex and high-dimensional generative tasks. In Chapter 7, this kind of method will be analyzed and described in detail.

5.2 Datasets and Metrics

In this Section, we will introduce the most common datasets used to test DGM methods and the metrics that can be used to assess the quality of the generated results. We remark that, to compare the ability of Generative Models in approximating the distribution $p_{gt}(\boldsymbol{x})$, we consider the task of image generation. This task is relatively simple to describe: after the model $p_\Theta(\boldsymbol{x})$ has been trained, we employ it to sample a set of realizations from $p_\Theta(\boldsymbol{x}) \approx p_{gt}(\boldsymbol{x})$. Then, we compare the similarity of the generated samples with a reference test set. The closer the generated images are to the test set, the better the model approximates $p_{gt}(\boldsymbol{x})$.

5.2.1 Datasets

We present here four datasets that will be used for the experiments in the following Chapters: MNIST, a dataset of low-dimensional images representing hand-written digits, CIFAR10, a dataset containing natural images of different categories, Oxford Flower, containing RGB images of flowers, and a modified version of CelebA, the famous dataset containing a large number of human faces.

MNIST MNIST [132] is a widely used benchmark dataset for image generation tasks. It comprises 60,000 grayscale images with a resolution of 28×28 pixels, each representing handwritten digits from 0 to 9. The images have a consistent black background and exhibit a variety of handwriting styles, adding diversity to the dataset.

CIFAR10 CIFAR10 [133], like MNIST, is a benchmark dataset used for image generation. It consists of 60,000 RGB images with a resolution of 32×32 pixels and is categorized

into ten classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Generating CIFAR10 images is considered a challenging task due to their low resolution and the complexity of the content.

Oxford Flowers The Oxford Flowers dataset [134] comprises 1,020 RGB images with various resolutions, depicting different types of flowers. In our experiments, the data has been pre-processed to a standardized shape of 64×64 pixels.

CelebA CelebA [135] is a large-scale dataset containing 202,599 RGB images of celebrities' faces. The images have a resolution of 178×218 pixels and depict faces with different poses and expressions. In our experiments, we pre-processed the dataset by center-cropping each image to a dimension of 64×64 pixels, preserving the essential features.

5.2.2 Metrics

Assessing the quality of generated images poses a significant challenge because there is no *ground-truth* to serve as a reference for comparison. Traditional image quality metrics like RMSE, PSNR, and SSIM, which rely on comparing generated images to ground-truth images, cannot be used in this context. Therefore, there is a need to develop perception-based metrics that measure the similarity of generated images to a reference dataset. These metrics often leverage the activations of layers in a pre-trained classification network. One widely used metric of this type is the Frechét Inception Distance (FID), which will be explained in the following Section.

FID The idea of Frechét Inception Distance (FID) [136] is to evaluate the quality of generated data by comparing the probability distribution of generated versus real images. However, the dimension of $\mathcal{X}$ is typically too large to allow a direct, significant comparison. Moreover, adjacent pixels are highly correlated, reducing their statistical relevance. To solve this issue, Frechét Inception Distance considers, instead of raw data, their internal representations generated by some third-party, agnostic network. In particular, the Inception v3 network [137] trained on Imagenet is used for this purpose; Inception is usually preferred over other models due to the limited amount of pre-processing performed on input images (images are rescaled in the interval $[-1, 1]$, sample wise). The activations that are traditionally used are those relative to the last pooling layer, with a dimension of 2048 features.

Given the activations $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, relative to real and generated images, and called $\mu_i, i = 1, 2$ and $\boldsymbol{C}_i, i = 1, 2$ their empirical mean and covariance matrix, respectively, the Fréchet Distance between $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ is defined as:

$$FID(\boldsymbol{a}_1, \boldsymbol{a}_2) = ||\mu_1 - \mu_2||^2 + Tr(\boldsymbol{C}_1 + \boldsymbol{C}_2 - 2(\boldsymbol{C}_1\boldsymbol{C}_2)^{\frac{1}{2}}), \tag{5.29}$$

where we indicate with Tr the *trace* of a matrix.

A problem of FID, is that it is extremely sensible to a number of different factors such as: the weights of the Inception network (which has been trained on ImageNet), the dimension of the datasets of real/generated images to be compared, the dimension of input images fed to Inception (which has to be rescaled), and the resizing algorithm. Moreover, articles in the literature are not always fully transparent on the previous points, which may explain some discrepancies and the difficulty one frequently faces in replicating results. All our experiments have been conducted with "defaults" values: the standard Inception checkpoint `inception_v3_2016_08_28/inception_v3.ckpt`, 10,000 images of dimension 299×299, rescaled by means of bilinear interpolation.

5.3 Structure of the part II of the thesis

In the following Chapter, we will describe the results of a set of experiments done during my PhD, with the intent of deeply understanding the properties of generative models. In particular, in Chapter 6, we compare multiple variants of the classical Variational Autoencoder in terms of their ability to generate new images. For each architecture, we also measure the number of Floating Points Operations (FLOPs) required to generate a single sample, since it is known to be a metric that correlates with the computational cost, as suggested in [2, 5]. In Chapter 7, we describe the deterministic version of the classical Diffusion Models, namely the *Denoising Diffusion Implicit Models (DDIM)* [138] and we investigate some properties of their latent space.

Chapter 6

A survey on Variational Autoencoders from a GreenAI perspective

Data generation, that is the task of generating new realistic samples given a set of training data, is a fascinating problem of AI, with many relevant applications in different areas, spanning from computer vision to natural language processing and medicine. Due to the curse of dimensionality, the problem was practically hopeless to solve, until Deep Neural Networks enabled the scalability of the required techniques via learned approximators. In recent years, deep generative models have gained a lot of attention in the deep learning community, not just for their amazing applications, but also for the fundamental insight they provide into the encoding mechanisms of Neural Networks, the extraction of deep features, and the latent representation of data.

In spite of the successful results, deep generative modeling remains one of the most complex and expensive tasks in AI. Training a complex generative model typically requires a lot of time and computational resources. To make a couple of examples, the hyper-realistic Generative Adversarial Network for face generation in [139] required training on 8 Tesla V100 GPUs for 4 days; the training of BERT [140], a well-known generative model for NLP, takes about 96 hours on 64 TPU2 chips.

As remarked in [2], this computational cost has huge implications, both from the ecological point of view and for the increasing difficulties for academics, students, and researchers, in particular those from emerging economies, to do competitive, state-of-the-art research. As a good practice in Deep Learning, one should give detailed reports about the financial cost of training and running models, in such a way as to promote the investigation of increasingly efficient methods.

In this chapter, we offer a comparative evaluation of some recent generative models. In particular, our focus is on the so-called Variational Autoencoders [141, 142] (VAEs), introduced in Section 5.1.2.

Variational Autoencoders are very popular inside the scientific community [143, 144],

both due to their strong probabilistic foundation, which has been introduced in Section 5.1.2 and will be recalled in Section 6.1, and the precious insight on the latent representation of data. However, in spite of the remarkable achievements, the behavior of Variational Autoencoders is still far from satisfactory; there are a number of well-known theoretical and practical challenges that still hinder this generative paradigm (see Section 6.2), and whose solution drove the recent research on this topic.

Contributions This chapter is based on my paper [4]. The contributions of this article are twofold. First of all, we provide a benchmark of the most important variants of the classical Variational Autoencoder, showing that most of the known challenges regarding VAEs are still present in these variants. In particular, we focus on a restricted subset of recent architectures that, in our opinion, deserve a deeper investigation, for their paradigmatic nature, the elegance of the underlying theory, or some key architectural insight. The three categories of models that we shall compare are the Two-stage model [145], the Regularized Autoencoder [146], and some versions of Hierarchical Autoencoders, such as the recent Nouveau VAE [147]. Secondly, we discuss the relationship of the discussed VAEs variants with the GreenAI paradigm, showing that often, increased generation quality leads to increased ecological impact.

My personal contributions to this work were to write the code and the experimental setup. A running implementation in TensorFlow 2 for this chapter is available on our GitHub repository https://github.com/devangelista2/GreenVAE.

Structure of the Chapter This Chapter is meant to offer a self-contained introduction to the topic of Variational Autoencoders, just assuming the basic knowledge of neural networks already described in the previous Chapter. In Section 6.1 we start with the theoretical background, discussing the strong and appealing probabilistic foundation of this class of generative models. In Section 6.2 we address the way theory is translated into a vanilla neural net implementation, and introduce the many issues arising from this operation: balancing problems in the loss function (Section 6.2.1), posterior collapse (Section 6.2.2), aggregate posterior vs. prior mismatch (Section 6.2.3), blurriness (Section 6.2.4) and disentanglement (Section 6.2.5).

In the next three Sections we give a detailed mathematical introduction to the three classes of models for which we provide a deeper investigation, namely the Two-Stage approach in Section 6.3, the regularized VAE in Section 6.4 and hierarchical models in Section 6.5. Section 6.6 is devoted to describing our experimental setting: we discuss the metrics used for the comparison, and provide a detailed description of the neural network architectures. In Section 6.7 we provide the results of our experimentation, making a critical discussion. In the conclusive Section 6.8 we summarize the content of the article and draw

a few considerations on the future of this field, and the challenges ahead.

6.1 Theoretical Background

In this Section, we extend the introduction to Variational Autoencoders (VAEs) from Section 5.1.2, describing theoretical details and deriving the so-called Evidence Lower Bound (ELBO) adopted as a learning objective for this class of models.

We recall that to deal with the problem of generating realistic data points, we consider a dataset $\mathcal{S} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}\} \subset \mathcal{X}$, where $\mathcal{X}$ is the manifold of *good* data, and we assume that there exists a ground-truth distribution μ_{gt} supported on $\mathcal{X}$, absolutely continuous with density $p_{gt}(\boldsymbol{x})$. We also assume that the distribution $p_{gt}(\boldsymbol{x})$ factorize as:

$$p_{gt}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[p_{gt}(\boldsymbol{x}|\boldsymbol{z})], \tag{6.1}$$

where $\boldsymbol{z} \in \mathbb{R}^s$ is the latent variable associated with $\boldsymbol{x}$, distributed with a simple distribution $p(\boldsymbol{z})$, as we did in Section 5.1.2.

We also recall that in DLVMs, one usually considers a family of probability distributions $\{p_\Theta(\boldsymbol{x}|\boldsymbol{z}); \Theta \in \mathbb{R}^s\}$, parameterized by a neural network, and optimizes Θ to minimize the Kullback-Leibler distance between $p_{gt}(\boldsymbol{x})$ and $p_\Theta(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[p_\Theta(\boldsymbol{x}|\boldsymbol{z})]$, which is equivalent to compute the Maximum Likelihood Estimation of $p_\Theta(\boldsymbol{x})$:

$$p_{\Theta^*} = \arg\max_{\Theta\in\mathbb{R}^s} \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})} \left[\log p_\Theta(\boldsymbol{x})\right]. \tag{6.2}$$

Unfortunately, (6.2) is usually computationally infeasible. For this reason, VAEs define another probability distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ named *encoder distribution* which describes the relationship between a data point $\boldsymbol{x} \in \mathcal{X}$ and its latent variable $\boldsymbol{z} \in \mathbb{R}^s$ and optimizes ϕ and Θ such that:

$$\Theta^*, \phi^* = \arg\min_{\Theta,\phi} \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\Theta(\boldsymbol{z}|\boldsymbol{x}))] \tag{6.3}$$

where $D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\Theta(\boldsymbol{z}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p_\Theta(\boldsymbol{z}|\boldsymbol{x})]$ is the Kullback-Leibler divergence between $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p_\Theta(\boldsymbol{z}|\boldsymbol{x})$.

But

$$\begin{aligned} D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\Theta(\boldsymbol{z}|\boldsymbol{x})) &= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p_\Theta(\boldsymbol{z}|\boldsymbol{x})] \\ &= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}) - \log p_\Theta(\boldsymbol{x}|\boldsymbol{z}) - \log p_\Theta(\boldsymbol{z}) + \log p_\Theta(\boldsymbol{x})] \\ &= D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) - \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\Theta(\boldsymbol{x}|\boldsymbol{z})] + \log p_\Theta(\boldsymbol{x}). \end{aligned} \tag{6.4}$$

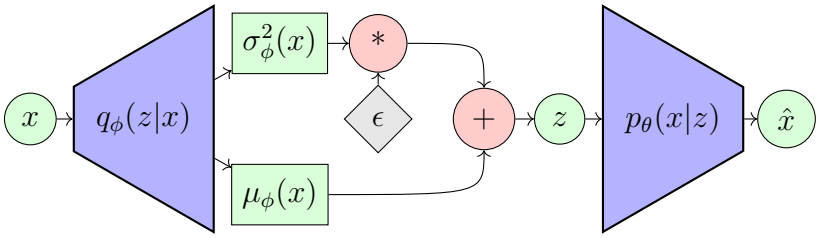


Figure 6.1: A diagram representing the VAE architecture. The stochastic component ϵ in the gray diamond is sampled from $\mathcal{N}(0, I)$.

Thus:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\Theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) &= \log p_\Theta(\boldsymbol{x}) - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\Theta(\boldsymbol{z}|\boldsymbol{x})) \\ &\leq \log p_\Theta(\boldsymbol{x}),\end{aligned} \tag{6.5}$$

where we used that $D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\Theta(\boldsymbol{z}|\boldsymbol{x})) \geq 0$. Equation (6.5) implies that the left-hand side $\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\Theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$ is a lower bound for the logarithm of $p_\Theta(\boldsymbol{x})$. For this reason, it is usually called *ELBO* (Evidence Lower BOund).

Since ELBO is more tractable than MLE, it is used as the cost function for the training of neural networks in order to optimize both Θ and ϕ:

$$\mathcal{L}_{\Theta,\phi}(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\Theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})), \tag{6.6}$$

$$\mathcal{L}_{\Theta,\phi} := \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[\mathcal{L}_{\Theta,\phi}(\boldsymbol{x})]. \tag{6.7}$$

It is worth to remark that ELBO has a form resembling an autoencoder, where the term $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ maps the input $\boldsymbol{x}$ to its latent representation $\boldsymbol{z}$, and $p_\Theta(\boldsymbol{x}|\boldsymbol{z})$ decodes $\boldsymbol{z}$ back to $\boldsymbol{x}$. Figure 6.1 shows a diagram representing the basic VAE structure.

For generative sampling, we forget the encoder and just exploit the decoder, sampling the latent variables according to the prior distribution $p(\boldsymbol{z})$ (that must be known).

6.2 The vanilla VAE and its problems

In this section, we explain how the theoretical form of the ELBO (eq. 6.6) can be translated into a numerical loss function exploitable for the training of neural networks. This will allow us to point out some of the typical problems that affect this architecture and whose solution drove the design of the variants discussed in the sequel.

In the vanilla VAE, we assume $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\phi(\boldsymbol{x}), \sigma^2_\phi(\boldsymbol{x}))$, so that learning $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ amounts to learning its two first moments. Similarly, we assume $p_\Theta(\boldsymbol{x}|\boldsymbol{z})$ has a Gaussian distribution around a decoder function $\mu_\Theta(\boldsymbol{z})$. The functions $\mu_\phi(\boldsymbol{x})$, $\sigma^2_\phi(\boldsymbol{x})$ and $\mu_\Theta(\boldsymbol{z})$ are modelled by deep neural networks.

If the model approximating the decoder function $\mu_\Theta(\boldsymbol{z})$ is sufficiently expressive (as it is the case for deep neural networks), the shape of the prior distribution $p(\boldsymbol{z})$ does not really

matter, and for simplicity, it is assumed to be a normal distribution $p(\boldsymbol{z}) = \mathcal{N}(0, I)$. The term $D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$ is hence the KL-divergence between two Gaussian distributions $\mathcal{N}(\mu_\phi(\boldsymbol{x}), \sigma^2_\phi(\boldsymbol{x}))$ and $\mathcal{N}(0, I)$ and it can be computed in closed form as:

$$D_{KL}(\mathcal{N}(\mu_\phi(\boldsymbol{x}), \sigma_\phi(\boldsymbol{x})), \mathcal{N}(0, I)) = \frac{1}{2}\sum_{i=1}^{s} \mu_\phi(\boldsymbol{x})_i^2 + \sigma^2_\phi(\boldsymbol{x})_i - \log(\sigma^2_\phi(\boldsymbol{x})_i) - 1, \tag{6.8}$$

where s is the dimension of the latent space. The previous equation has an intuitive explanation as a cost function. By minimizing $\mu_\phi(\boldsymbol{x})$ for $\boldsymbol{x}$ varying on the whole dataset, we are centering the latent space around the origin (i.e. the mean of the prior). The other component is preventing the variance $\sigma^2_\phi(\boldsymbol{x})$ from dropping to zero, implicitly enforcing a better coverage of the latent space.

Coming to the reconstruction loss $\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\Theta(\boldsymbol{x}|\boldsymbol{z})]$, under the Gaussian assumption, we already remarked that the logarithm of $p_\Theta(\boldsymbol{x}|\boldsymbol{z})$ is the quadratic distance between $\boldsymbol{x}$ and its reconstruction $\mu_\Theta(\boldsymbol{z})$. The variance of this Gaussian distribution can be understood as a parameter balancing the relative importance between reconstruction error and KL-divergence [131].

The problem of integrating sampling with backpropagation during training is solved by the well-known reparametrization trick proposed in [141, 142], where the sample is performed using a standard distribution (outside of the backpropagation flow), and this value is rescaled with $\mu_\phi(\boldsymbol{x})$ and $\sigma_\phi(\boldsymbol{x})$.

The basic model of the Vanilla VAE that we just outlined is unfortunately hindered by several known theoretical and practical challenges. In the next Sections, we give a short list of important topics that have been investigated in the literature, along with a short discussion of the main works addressing them.

6.2.1 The balancing issue

The VAE loss function is the sum of two distinct components, with somehow contrasting effects

$$\mathcal{L}_{\Theta,\phi}(\boldsymbol{x}) := \underbrace{\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\Theta(\boldsymbol{x}|\boldsymbol{z})]}_{\text{log-likelihood}} - \gamma \underbrace{D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))}_{\text{KL-divergence}}. \tag{6.9}$$

The log-likelihood loss is just meant to improve the quality of reconstruction, while the Kullback-Leibler component is acting as a regularizer, pushing the aggregate inference distribution $q_\phi(\boldsymbol{z}) = \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[q_\phi(\boldsymbol{z}|\boldsymbol{x})]$ towards the desired prior $p(\boldsymbol{z})$.

Log-likelihood and KL-divergence are frequently balanced by a suitable parameter, allowing tuning of their mutual relevance. The parameter is called γ, in this context, and it is considered as a normalizing factor for the reconstruction loss.

Privileging log-likelihood will improve the quality of reconstruction, neglecting the shape of the latent space (with ominous effects on generation). Privileging KL-divergence typically

results in a smoother and normalized latent space, and more disentangled features [148, 149]; this usually comes at the cost of a more noisy encoding, finally resulting in more blurriness in generated images. [150]. Discovering a good balance between these components is a crucial aspect for the effective training of VAEs.

Several techniques for the calibration of γ have been investigated in the literature, comprising an annealed optimization schedule [151] or a policy enforcing minimum KL contribution from subsets of latent units [152]. These schemes typically require hand-tuning and, as observed in [153], they easily risk interfering with the principled regularization scheme that is at the core of VAEs.

An alternative possibility, investigated in [145], consists in *learning* the correct value for the balancing parameter during training, which also allows its automatic calibration along the training process.

In [154] it is observed that considering the objective function used in [145] in order to learn γ, the optimal γ parameter is in fact proportional to the current reconstruction error; so learning can be replaced by a mere computation, using e.g. a running average. This has a simple and intuitive explanation: what matters is to try to maintain a *fixed* balance between the two components during training: if the reconstruction error decreases, we must proportionally decrease the KL component that could otherwise prevail, preventing further improvements. The technique in [154] is simple and effective: we shall implicitly adopt it in all our VAE models unless explicitly stated differently. A similar technique has been recently investigated in [155], where the KL-divergence is used as feedback during model training for dynamically tuning the balance of the two components.

6.2.2 Variable collapse phenomenon

The KL-divergence component of the VAE loss function typically induces a parsimonious use of latent variables, some of which may be altogether neglected by the decoder, possibly resulting in an under-exploitation of the network capacity; if this is a beneficial side effect or regularization [145, 156] or an issue to be solved [153, 157, 158, 159], it is still debated.

The variable collapse phenomenon has a quite intuitive explanation. If, during training, a latent variable gives a modest contribution to the reconstruction of the input (in comparison with other variables), then the Kullback-Leibler divergence may prevail, pushing the mean towards 0 and the standard deviation towards 1. This will make the latent variable even more noisy, in a vicious cycle that will eventually induce the network to completely ignore the latent variable (see Figure 6.2, Left).

As described in [160], one can easily get empirical evidence of the phenomenon by adding some artificial noise to a variable and monitoring its evolution during training (Figure 6.2, Right). The contribution of a latent variable to reconstruction is computed as the difference between the reconstruction loss when the variable is masked with respect to the case when

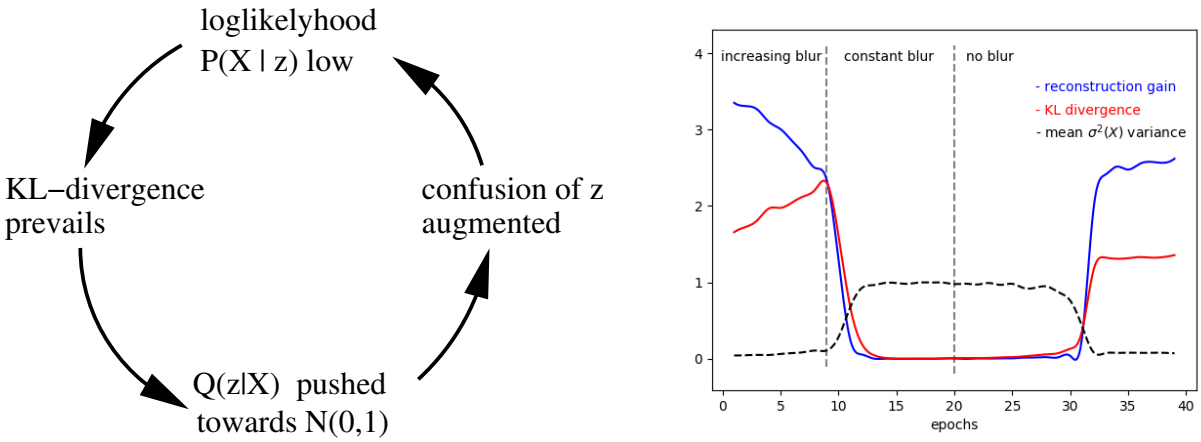


Figure 6.2: (Left) The vicious cycle leading to the variable collapse. (Right) An empirical demonstration of the phenomenon: we apply a progressive noise to a latent variable, reducing its contribution to reconstruction; at some point, KL-divergence prevails, enlarging the sampling variance of the variable and making it even more noisy; the phenomenon has a catastrophic nature, leading to a complete collapse of the variable. If we remove the artificial noise, the variable gets reactivated. Pictures borrowed from [160].

it is normally taken into account; we call this information *reconstruction gain*.

When the reconstruction gain of the variable is less than the KL-divergence, the variable gets ignored by the network: its correspondent mean value will collapse to 0 (independently from $\boldsymbol{x}$) and its sampling variance is pushed to 1. Sampling has no impact on the network, precisely because the variable is ignored by the decoder.

The variable collapse phenomenon is, to some extent, reversible. However, reactivating a collapsed variable is not a completely trivial operation for a network, probably due to saturation effects and vanishing gradients.

6.2.3 Aggregate posterior vs. expected prior mismatch

The crucial point of VAEs is to learn an encoder to produce an *aggregate posterior distribution* $q_\phi(\boldsymbol{z}) = \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[q_\phi(\boldsymbol{z}|\boldsymbol{x})]$ close to the prior $p(\boldsymbol{z})$. If this objective is not achieved, the generation is doomed to fail.

Before investigating ways to check the intended behavior, let us discuss how the Kullback-Leibler divergence term in (6.9) acts on the distance between $q_\phi(\boldsymbol{z})$ and $p(\boldsymbol{z})$. So, let us

average over $\boldsymbol{x}$:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})|p(\boldsymbol{z}))] \\
&= -\mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[\mathcal{H}(q(\boldsymbol{z}|\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[\mathcal{H}(q(\boldsymbol{z}|\boldsymbol{x}), p(\boldsymbol{z}))] && \text{by def. of KL} \\
&= -\mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[\mathcal{H}(q(\boldsymbol{z}|\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{z})]] && \text{by def. of entropy} \\
&= -\mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[\mathcal{H}(q(\boldsymbol{z}|\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log p(\boldsymbol{z})] && \text{by marginalization} \\
&= -\underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x})}[\mathcal{H}(q(\boldsymbol{z}|\boldsymbol{x}))]}_{\substack{\text{Avg. Entropy} \\ \text{of } q(\boldsymbol{z}|\boldsymbol{x})}} + \underbrace{\mathcal{H}(q(\boldsymbol{z}), p(\boldsymbol{z}))}_{\substack{\text{Cross-entropy of} \\ q(\boldsymbol{x}) \text{ VS } p(\boldsymbol{z})}} && \text{by def. of entropy}
\end{aligned} \tag{6.10}$$

By minimizing the cross-entropy between $q(\boldsymbol{z})$ and $p(\boldsymbol{z})$ we are pushing one towards the other. Jointly, we try to augment the entropy of $q(\boldsymbol{z}|\boldsymbol{x})$; under the assumption that $q(\boldsymbol{z}|\boldsymbol{x})$ is Gaussian, its entropy is $\frac{1}{2}\log(e\pi\sigma^2)$: we are thus enlarging the (mean) variance, further improving the coverage of the latent space, essential for generative sampling.

As a simple sanity check, one should always monitor the moments of the aggregate posterior distribution $q(\boldsymbol{z})$ during training: the mean should be 0, and the variance 1. Since collapsed variables could invalidate this computation (both mean and variance are close to 0), it is better to use an alternative rule [161]: if we look at $q(\boldsymbol{z}) = \mathbb{E}_{p_{gt}(\boldsymbol{x})}[q(\boldsymbol{z}|\boldsymbol{x})]$ as a Gaussian Mixture Model (GMM), its variance σ^2_{GMM} is given by the sum of the variances of the means $\mathbb{E}_{p_{gt}(\boldsymbol{x})}[\mu_\phi(\boldsymbol{x})^2]$ and the mean of the variances $\mathbb{E}_{p_{gt}(\boldsymbol{x})}[\sigma^2_\phi(\boldsymbol{x})]$ of the components (supposing that $\mathbb{E}_{p_{gt}(\boldsymbol{x})}[\mu_\phi(\boldsymbol{x})]$=0):

$$\sigma^2_{GMM} = \mathbb{E}_{p_{gt}(\boldsymbol{x})}[\mu_\phi(\boldsymbol{x})^2] + \mathbb{E}_{p_{gt}(\boldsymbol{x})}[\sigma^2_\phi(\boldsymbol{x})] = 1 \tag{6.11}$$

where in this case $\mu_\phi(\boldsymbol{x})$ and $\sigma^2_\phi(\boldsymbol{x})$ are the values computed by the encoder.

This is called *variance law* in [161] and can be used to verify that the regularization effect of the KL-divergence is properly working.

The big problem is that even if the two first moments of $q(\boldsymbol{z})$ are 0 and 1, this does not imply that it should look like a Normal (meaning that the KL-divergence got lost in some local minimum, contenting itself with adjusting the first moments of the distributions).

The potential mismatch between $q(\boldsymbol{z})$ and the expected prior $p(\boldsymbol{z})$ is a problematic aspect of VAEs that, as observed by many authors [161, 162, 163], could seriously compromise the whole generative framework. Attempts to solve this issue have been made both by acting on the loss function [164] or by exploiting more complex priors [152, 165, 166].

An interesting possibility, that has been recently deployed in the Hyperspherical VAE [167], consists in replacing the Gaussian Distribution with the von Mises-Fisher (vMF) distribution [168], that is a continuous distribution on the N-dimensional sphere in use in *directional statistics*.

An orthogonal, drastic alternative consists in renouncing to work in the comfortable setting of continuous latent variables, passing instead in the discrete domain. This approach

is at the core of the Vector Quantized VAE [169] (VQ-VAE): each latent variable is forced to occupy a position in a finitely sampled space so that we can treat each latent variable as a s-dimensional vector in a space of dimension n. This discrete encoding is exploited during sampling, where the prior is learned via a suitable autoregressive technique.

Clustering, GMM, and Two-stage

In case input data are divided into subcategories (as in the case of MNIST and Cifar10), or have macroscopic attributes like, say, a different color for hairs in the case of CelebA, we could naturally expect to observe this information in the latent encoding of data [170]. In other words, we could imagine the latent space to be organized in *clusters*, (possibly) reflecting macroscopic features of data.

To make an example, Figure 6.3 describes the latent encoding of MNIST digits, with a different color for each class in the range 0-9.

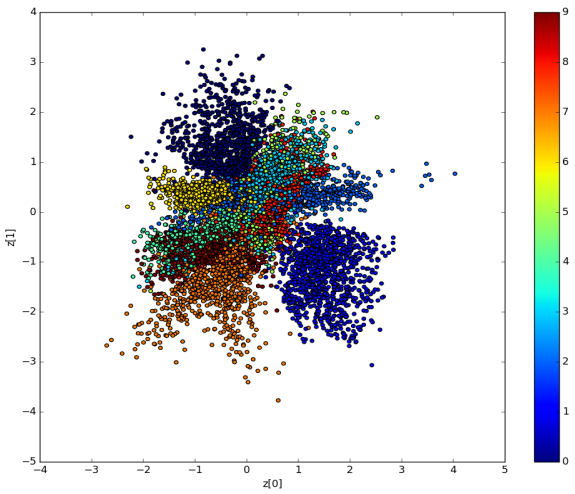


Figure 6.3: Latent encoding of MNIST digits in a latent space of dimension 2. Digits in different categories are represented with a different color. Observe: (1) the overall (rough) Gaussian-like disposition of all digits and (2) the typical organization in clusters, in contrast with the uni-modal objective of KL-regularization.

We can clearly observe that different digits naturally organize themselves in separate clusters. While the overall distribution still has a Gaussian-like shape, the presence of clusters may obviously contrast with the required smoothness of the internal encoding, introducing regions with higher/lower probability densities. Observe e.g. the gaps between some of the clusters: sampling in such a region will eventually result in a poor generative output. In other words, clustering could be one of the main sources for the mismatch between the prior and the aggregate posterior.

While the phenomenon is evident in a low-dimensional setting, it is more difficult to observe and testify it in higher dimensions. Remember that one of the VAE assumptions is that, as far as you have a sufficiently expressive decoder, the prior does not really matter since the decoder will be able to turn each distribution into the desired one [131].

Still, it makes sense to try to exploit clustering and a natural approach consists in using a GMM model. Several works have been done in this direction. The simplest approach, followed in [146], is to superimpose a GMM of fixed dimension on the latent space via ex-post estimation using standard machine learning techniques (this is also the approach we shall follow in some of our tests). Alternatively, the GMM model can be *learned*. In the Variational Deep Embedding approach [170] (VaDE), which essentially provides an unsupervised clustering model, the relevant statistics of the GMM are estimated via Maximum Likelihood Estimation, in a way similar to the Vanilla case (see also [171] for a similar, slightly more sophisticated approach).

In the so-called Two-Stage model [145], a second VAE is trained to learn an accurate approximation of $q(\boldsymbol{z})$; samples from a Normal distribution are first used to generate samples of $q(\boldsymbol{z})$, passed to the actual generator of data points. We shall give an extensive discussion of to the Two-Stage approach in Section 6.3.

In [172], it is proposed to give an ex-post estimation of $q(\boldsymbol{z})$, e.g. imposing a distribution with a sufficient complexity (they consider a combination of 10 Gaussians, reflecting the ten categories of MNIST and Cifar10). A suitable regularization technique alternative to KL is used to induce the desirable smoothness of the latent space. A deeper analysis of this approach is done in Section 6.4.

An additional and interesting issue of the Two-Stage model concerns the similarity measure to use as a loss function in the second stage. In [145], the traditional mean squared error and categorical cross entropy are considered. However, we discovered that *cosine distance* works amazingly better. We did not get to cosine distance by trial and error, but by a long and deep investigation on latent representations. These results will be the object of a forthcoming article.

6.2.4 Blurriness

Variational Autoencoders (VAEs), in comparison with alternative generative techniques, usually produce images with a characteristic and annoying blurriness. The phenomenon can also be observed in terms of the mean-variance of pixels in generated images, which is significantly lower than that for data in the training set [173].

The source of the problem is not easy to identify, but it is likely due to *averaging*, implicitly underlying the VAE frameworks (and, more generally, the whole autoencoder approach). In the presence of multimodal output, a loglikelihood objective typically results in averaging and hence blurriness [129].

Variational Autoencoders are intrinsically multimodal, both due to dimensionality reduction and to the sampling process during training.

Several attempts to solve the issue by acting on the reconstruction metrics have been made. Structural similarity (frequently used for deblurring purposes) does not seem to be effective [174]. Better results can be obtained by considering deep hidden features extracted from a pre-trained image classification model, like e.g. VGG19 [175]. In models of the VAE-GAN family [176, 177, 178], the reconstruction loss is altogether replaced by a discriminator trying to distinguish real images from generated ones. The use of a discriminator, assessing the quality of generated data and acting on the density of the prior, is also a basic component of the recent VAEPP model (VAEs with a pullback prior) [179].

The most promising approaches are however based on iterative/hierarchical approaches [180, 181, 147]. In these architectures, following the idea of latent Gaussian models [182], the vector of latent variables $\boldsymbol{z}$ is split into L groups of latent variables $\boldsymbol{z}_l, l = 1, ..., L$ and the density over the variable of interest is constructed sequentially, in terms of latent variables of lower indices. For instance, the prior $p(\boldsymbol{z})$ would be written as an autoregressive density of the following kind:

$$p(\boldsymbol{z}) = \prod_{l=1}^{L} p_l(\boldsymbol{z}_l|\boldsymbol{z}_{<l}). \tag{6.12}$$

.

Similarly, the inference probability would be decomposed as:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \prod_{l=1}^{L} q_\phi^{(l)}(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l}), \tag{6.13}$$

where $q_\phi^{(l)}(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l})$ is the encoder density of the l-th group. Suitable (iterative) neural networks modules are used to sequentially compute the relevant statistics of these distributions, in terms of previous outputs.

As an example of these architectures, the structure of NVAE will be detailed in Section 6.6.2.

The advantage of this approach is that it usually allows to work with a larger number of latent variables, responsible for small and progressive adjustments of generated samples.

6.2.5 Disentanglement

Besides the task of generating new images, [148] and [149] noticed that VAEs can also be used to learn an efficient way to represent the data, with important applications in transfer learning and classification.

To understand this phenomenon, suppose that there exists a set of *true* generative factors $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_S) \in \mathbb{R}^S$ such that $p_{gt}(\mathbf{v}|\boldsymbol{x}) = \prod_{i=1}^{S} p_{gt}(\mathbf{v}_i|\boldsymbol{x})$ (i.e. $\mathbf{v}$ are conditionally independent given $\boldsymbol{x}$) and that each $\mathbf{v}_i$ encodes a meaningful feature of the data point $\boldsymbol{x}$

generated by it. Under the assumption that $s \geq S$, the latent variables $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k)$ learnt during the training are a redundant representation of $\mathbf{v}$ in a basis where the features are not disentangled. To learn an optimal latent representation of the input image $\boldsymbol{x}$, it is necessary to train the network in such a way that S coordinates of $\boldsymbol{z}$ are related to $\mathbf{v}$, while the other $s - S$ coordinates can be used to improve the reconstruction of $\boldsymbol{x}$, recovering the high frequency components that are missing in $\mathbf{v}$.

In β-VAE [149, 148], this constraint is imposed by noting that in the ELBO function the prior distribution $p(\boldsymbol{z}) = \mathcal{N}(0, I)$ forces the decoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to learn a vector $\boldsymbol{z}$ where each variable is independent of each other. To improve disentanglement, we should hence induce the D_{KL} term to be as small as possible, which can be achieved by augmenting the decoder variance γ to be greater than 1. Unfortunately, since :

$$\mathbb{E}_{p_{gt}(\boldsymbol{x})}[D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))] = D_{KL}(q_\phi(\boldsymbol{z})||p(\boldsymbol{z})) + I_{q_\phi}(\mathbf{X}; \mathbf{Z}),$$

where $I_{q_\phi}(\mathbf{X}; \mathbf{Z})$ is the mutual information between $\mathbf{X}$ and $\mathbf{Z}$ with respect to the joint distribution $q_\phi(\boldsymbol{x}, \boldsymbol{z}) = q_\phi(\boldsymbol{z}|\boldsymbol{x})p_{gt}(\boldsymbol{x})$, by pushing $D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$ to zero, the mutual information between $\mathbf{X}$ and $\mathbf{Z}$ is also minimized, reducing the reconstruction efficiency of the network. This problem is addressed in [183, 184] where the ELBO is modified by adding more parameters with the intent to improve disentanglement without losing too much performance.

6.3 Two-Stage VAE

To address the mismatch of aggregate posterior versus the expected prior, Bin Dai and David Wipf in [145], introduced the Two-Stage VAEs.

The idea behind this model is to train two different VAEs sequentially. The first VAE is used to learn a good representation $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ of the data in the latent space without guaranteeing exactly $q(\boldsymbol{z}) = p(\boldsymbol{z})$, whereas the second VAE should learn to sample from the true $q(\boldsymbol{z})$ without using the prior distribution $p(\boldsymbol{z})$. A scheme of the implementation follows (a detailed architectural description is given in Section 6.6.2):

- Given a data set $\mathcal{S} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$, train a VAE with a fixed latent dimension s, possibly small.
- Generate latent samples $\mathcal{Z} = \{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(N)}\}$ via $\boldsymbol{z}^{(i)} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})$, $i = 1, \ldots N$. By design, these samples are distributed as $q_\phi(\boldsymbol{z}) = \mathbb{E}_{p_{gt}(\boldsymbol{x})}[q_\phi(\boldsymbol{z}|\boldsymbol{x})]$, but likely not as $p(\boldsymbol{z}) = \mathcal{N}(0, I)$.
- Train a second VAE with parameters (Θ', ϕ') and latent variable $\mathbf{u} \sim p(\mathbf{u}) = \mathcal{N}(0, I)$ of dimension s to learn the distribution $q_\phi(\boldsymbol{z})$ with $\mathcal{Z}$ as the dataset.

- Sample new images by ancestral sampling, i.e. by first sampling $\mathbf{u} \sim p(\mathbf{u})$, then generate a $\boldsymbol{z}$ value by $p_{\Theta'}(\boldsymbol{z}|\mathbf{u})$ and finally $\boldsymbol{x} \sim p_{\Theta}(\boldsymbol{x}|\boldsymbol{z})$.

The theoretical foundation of the Two-Stage VAE algorithm is well presented in [145]. We summarize here the main results. The two VAEs aim at separating the components of the ELBO loss function (6.9), by suitably using the decoder variance γ. Remarking that $p_{gt}(\boldsymbol{x})$ is the unknown data distribution which we desire to learn and that $p_{\Theta}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim q_{\phi}(\boldsymbol{z})}[p_{\Theta}(\boldsymbol{x}|\boldsymbol{z})]$ is the learned distribution, we hope that $p_{\Theta}(\boldsymbol{x}) \approx p_{gt}(\boldsymbol{x})\ \forall \boldsymbol{x}$.

Unfortunately, this is not always possible. In fact, there is a critical distinction between the cases where the dimension of the data n and the latent space dimension s are equal, and the case where $n > s$.

As a matter of fact, in the first case, it is possible to prove that, under suitable assumptions, for the optimal choice of the parameters (Θ^*, ϕ^*) it holds that $p_{\Theta^*}(\boldsymbol{x}) = p_{gt}(\boldsymbol{x})$ almost everywhere (i.e. VAEs strongly converges to the true distribution $p_{gt}(\boldsymbol{x})$). In the second case, only weak convergence, in the sense that $\int_A p_{\Theta^*}(\boldsymbol{x})d\boldsymbol{x} = \int_A p_{gt}(\boldsymbol{x})d\boldsymbol{x}$ where A is an open subset of $\mathbb{R}^n$, can be proved (see Theorems 1 and 2 in [145]).

In the first stage, since the ambient dimension is obviously greater than the latent space dimension (i.e. $n > s$), for the previous results only a weak convergence is guaranteed; the parameter γ is chosen in this case in order to get a good reconstruction (Theorem 4 in [145]). In the second stage by construction, the data variable $\boldsymbol{z}$ and its correspondent latent variable $\boldsymbol{u}$ have the same dimension, hence the unknown distribution $q_{\phi}(\boldsymbol{z})$ is exactly identified by the VAE. As a consequence it is possible to sample directly from $q_{\phi}(\boldsymbol{z})$, without using the prior $p(\boldsymbol{z})$, thus bypassing the problem of mismatch between the aggregate posterior and the prior distributions.

6.4 Regularized VAE (RAE)

One of the most interesting variations of vanilla VAE is the work of Partha Ghosh and Mehdi S. M. Sajjadi [172], where the authors tried to solve all the problems related to the classical VAE by completely changing the way of approaching the problem. They pointed out that, in their typical implementation, VAEs can be seen as a regularized Autoencoder with Additive Gaussian Noise on the decoder input. In their work, the authors argued that noise injection in decoder input can be seen as a form of regularization since it implicitly helps to smooth the function learned by the network.

To get a new insight into this problem, they took into consideration the distinct components of ELBO already introduced in (6.9):

$$\mathcal{L}_{\Theta,\phi}(\boldsymbol{x}) := \underbrace{\mathbb{E}_{\boldsymbol{z}\sim q_{\phi}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\Theta}(\boldsymbol{x}|\boldsymbol{z})]}_{:=\mathcal{L}_{REC}(\Theta,\phi)} - \gamma \underbrace{D_{KL}(q_{\phi}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))}_{:=\mathcal{L}_{KL}(\phi)}, \tag{6.14}$$

where $\mathcal{L}_{REC}$ is a term that measures the distance between the input and the reconstruction, whereas $\mathcal{L}_{KL}$ is a regularization term that enforces the aggregate posterior to follow the prior distribution.

To show how $\mathcal{L}_{KL}(\phi)$ regularizes the loss, in [172] the Constant-Variance VAEs (CV-VAEs) [172] have been investigated, where the encoder variance $\sigma^2_\phi(\boldsymbol{x})$ is fixed for every $\boldsymbol{x} \in \mathcal{S}$ and thus treated as a hyperparameter σ^2. In this situation,

$$\mathcal{L}_{REC}(\Theta, \phi) = -\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{1}{2}||\boldsymbol{x} - \mu_\Theta(\boldsymbol{z})||_2^2\right] \tag{6.15}$$

$$\mathcal{L}_{KL}(\phi) = D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) = ||\mu_\phi(\boldsymbol{x})||_2^2 + C \tag{6.16}$$

$$\mathcal{L}_{\Theta,\phi}(\boldsymbol{x}) = -\mathbb{E}_{p_{gt}}\left[\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{1}{2}||\boldsymbol{x} - \mu_\Theta(\boldsymbol{z})||_2^2\right] - \gamma||\mu_\phi(\boldsymbol{x})||_2^2\right]. \tag{6.17}$$

We observe that the expression in (6.17) is a Mean Squared Error (MSE) with L_2 regularization on $\mu_\phi(\boldsymbol{x})$.

The authors proposed to substitute noise injection in the decoder input with an explicit regularization scheme in a classical CV-VAE. This is done by modifying the cost function $\mathcal{L}_{\Theta,\phi} = \mathbb{E}_{p_{gt}(\boldsymbol{x})}[\mathcal{L}_{REC}(\Theta, \phi) - \gamma\mathcal{L}_{KL}(\phi) - \lambda\mathcal{L}_{REG}(\Theta)]$ where $\mathcal{L}_{REG}(\Theta)$ is a regularizer for the decoder weights, while $\gamma, \lambda \geq 0$ are regularization parameters.

Whereas $\mathcal{L}_{REC}(\Theta, \phi) = -\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\frac{1}{2}||\boldsymbol{x} - \mu_\Theta(\boldsymbol{z})||_2^2]$ and $\mathcal{L}_{KL}(\phi) = \frac{1}{2}||\boldsymbol{z}||_2^2$ are fixed a priori by the CV-VAE architecture, $\mathcal{L}_{REG}(\Theta)$ needs to be defined. The choice for $\mathcal{L}_{REG}(\Theta)$ identifies the specific kind of network. Ghosh and Sajjadi proposed three possible choices for $\mathcal{L}_{REG}(\Theta)$:

- *L_2-Regularization*, where $\mathcal{L}_{REG}(\Theta) = ||\Theta||_2^2$ is simply the weight decay on the decoder parameters.
- *Gradient Penalty*, where $\mathcal{L}_{REG}(\Theta) = ||\nabla\mu_\Theta(\mu_\phi(\boldsymbol{x}))||_2^2$ bounds the gradient norm of the decoder with respect to its input, enforcing smoothness.
- *Spectral Normalization*, where each weight matrix Θ_l in the decoder is normalized by an estimate of its largest singular value: $\Theta_l^{SN} = \frac{\Theta_l}{s(\Theta_l)}$ (the estimate $s(\Theta_l)$ can be easily obtained with one iteration of the power method).

Moreover, they argued that removing noise injection from the decoder input prevents from knowing the distribution of latent variables, thus losing the generative ability of the network. They solved this problem by proposing an *ex-post density estimation*, where the distribution of the latent variables is learned a posteriori, by fitting $\mathcal{Z} = \{\boldsymbol{z}^i; \boldsymbol{z}^i = \mu_\phi(\boldsymbol{x}^i)\}$ with a GMM model $q_\delta(\boldsymbol{z})$ with a fixed number of components and then sampling $\boldsymbol{z}$ from $q_\delta(\boldsymbol{z})$ to generate new samples from $p_\Theta(\boldsymbol{x}|\boldsymbol{z})$. The generative model defined in this way is called *Regularized Autoencoder (RAE)*.

6.5 Hierarchical Variational Autoencoder

To improve the quality of the generation in Variational Autoencoders, Kingma et al. [152] strengthened the inference network $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ with the powerful Normalizing Flows [124] introduced by Rezende and Mohamed in 2015. The idea of Normalizing Flows is to begin with a latent variable $\boldsymbol{z}_0$ sampled by a simple distribution $q_\phi(\boldsymbol{z}_0|\boldsymbol{x})$, and to iteratively construct more complex variables by applying transformations $\boldsymbol{z}_t = f_t(\boldsymbol{z}_{t-1})$ for $t = 1, \dots, T$. By observing that the D_{KL} expression is:

$$D_{KL}(q_\phi(\boldsymbol{z}_T|\boldsymbol{z}_{<T}, \boldsymbol{x})||p(\boldsymbol{z}_T)) = \mathbb{E}_{\boldsymbol{z}_T \sim q_\phi(\boldsymbol{z}_T|\boldsymbol{z}_{<T}, \boldsymbol{x})}\Big[\log q_\phi(\boldsymbol{z}_T|\boldsymbol{z}_{<T}, \boldsymbol{x}) - \log p(\boldsymbol{z}_T)\Big], \tag{6.18}$$

its implementation requires the computation of the logarithm of $q_\phi(\boldsymbol{z}_T|, \boldsymbol{z}_{<T}, \boldsymbol{x})$. If the functions $f_t(\cdot)$ are simple enough, it is possible to efficiently use them to compute $\log q_\phi(\boldsymbol{z}_T|\boldsymbol{z}_{<T}, \boldsymbol{x})$ as:

$$\log q_\phi(\boldsymbol{z}_T|\boldsymbol{z}_{<T}, \boldsymbol{x}) = \log q_\phi(\boldsymbol{z}_0|\boldsymbol{x}) - \sum_{t=1}^{T} \log \det \Big| \frac{\partial f_t}{\partial \boldsymbol{z}_{t-1}} \Big|, \tag{6.19}$$

where $\frac{\partial f_t}{\partial \boldsymbol{z}_{t-1}}$ is the Jacobian matrix of $f_t(\boldsymbol{z}_{t-1})$ computed by repeatedly applying the well-known change of variable theorem to the multi-variate random variable z_T defined as:

$$\boldsymbol{z}_T = f_T(f_{T-1}(\dots(f_1(\boldsymbol{z}_0))\dots)). \tag{6.20}$$

An interesting aspect concerning Normalizing Flows is that, under suitable assumptions, they are provably universal, in the sense defined in [185]. As already mentioned, the first successful integration of Normalizing Flows in VAEs was by Kingma et al. in [152], where they introduced Inverse Autoregressive Flows (IAF). The idea was to define $f_t(\boldsymbol{z}_{t-1})$ as a simple affine function of the form:

$$\boldsymbol{z}_t = f_t(\boldsymbol{z}_{t-1}) = \mu_t + \sigma_t \odot \boldsymbol{z}_{t-1}, \quad \forall t = 1, \dots, T, \tag{6.21}$$

where $\boldsymbol{z}_0 \sim q_\phi(\boldsymbol{z}_0|\boldsymbol{x}) = \mathcal{N}(\mu_\phi(\boldsymbol{x}), \sigma^2_\phi(\boldsymbol{x}))$.
Figure 6.4 schematically represents the unrolling of equation (6.21).

We highlight that the IAF introduces a natural order in the latent variables. For this reason, we will refer to this kind of model as Hierarchical Variational Autoencoder (HVAE). In this paradigm, we will refer to each $\boldsymbol{z}_t$ as a group of latent variables, and we will collect the set of all groups in a vector $\boldsymbol{z} = (\boldsymbol{z}_0, \dots, \boldsymbol{z}_T)$ where the variables are written in the order defined above.
If we distinguish between the encoder (inference) network $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the decoder (generative) network, we need to choose if the ordering of latent variables is the same in the two parts of the network (*bottom-up inference*), or if it is reversed (*bidirectional inference*) as shown in Figure 6.5.

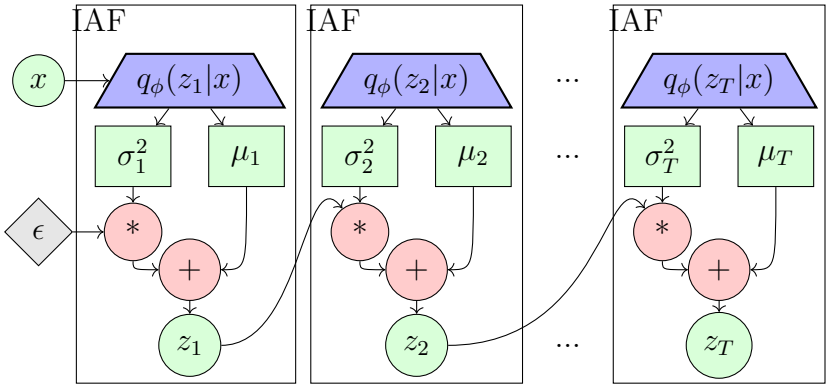
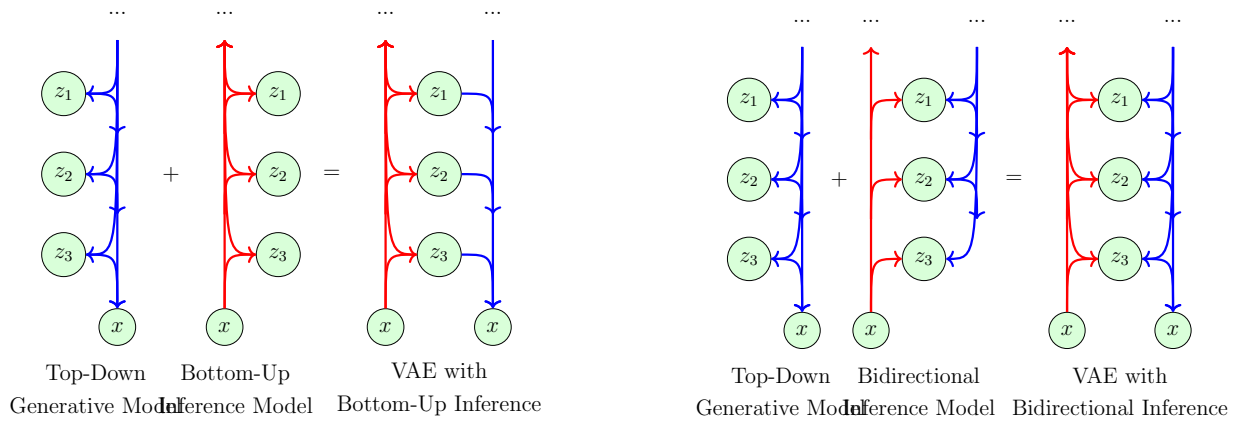


Figure 6.4: A scheme of Inverse Autoregressive Flow. Each white box represents one iteration of equation (6.21), where μ_t, σ_t^2 are generated by the encoder $q_\phi(\boldsymbol{z}_t|\boldsymbol{x})$.



(a) (b)

Figure 6.5: Diagrams that schematically represent Hierarchical VAE in two different configurations: Bottom-Up Inference (a) and Bidirectional Inference (b).

As it is clear from Figure 6.5, in bottom-up inference the image $\boldsymbol{x} \in \mathbb{R}^n$ is encoded to $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T)$ independently from the prior $p(\boldsymbol{z}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t|\boldsymbol{z}_{<t})$; in the generative phase the image is reconstructed by taking $\boldsymbol{z}_T$ as the final output of the encoder, and then sampling each $\boldsymbol{z}_t$, $t = T-1, \ldots, 0$ from the prior distribution independently from $q_\phi(\boldsymbol{z}_t|\boldsymbol{x})$ (i.e. the encoder and decoder are independent of each other). We underline that this fact makes the bottom-up inference training unstable.

Conversely, in bidirectional inference, the process of generating latent variables is shared between the two parts of the network, which makes the training easier, although the design of the network is a bit more difficult.

Since the results of vanilla IAF are not competitive with the state-of-art, we will not use them in our future analysis (see the original paper for more information), whereas we will focus our experimental results on two powerful variants of IAF, making use of bidirectional inference and residual blocks to generate high-quality images.

6.6 Experimental setting

For each variant of Variational Autoencoder discussed in the previous section, we provide detailed benchmarks on some traditional datasets listed in Section 5.2.1, that is MNIST, CIFAR10, and CelebA. The specific architectures which have been tested are described in the following. All models have been compared using standard metrics, assessing both their energy consumption through the number of Floating Point Operations (FLOPS) (see Section 6.6.1), and their performance via the so-called Frechèt Inception Distance [136], introduced in Section 5.2.2. Numerical results are given in Section 6.7, along with examples of reconstructed and generated images.

6.6.1 Green AI and FLOPS

The paradigm of Green AI [2] is meant to raise attention to the computational efficiency of neural models, encouraging a reduction in the amount of resources required for their training and deployment. This concept is not as trivial as it seems; in fact, most traditional AI research (referred to as Red AI) targets accuracy rather than efficiency, exploiting massive computational power, and resulting in rapidly escalating costs; this trend is not sustainable for various reasons, it is environmentally unfriendly [186], socially not inclusive and inefficient.

The computation of floating point operations (FLOPS) was advocated in [2] as a measure of the efficiency of models; the main advantages of this measure are that it is hardware-independent and has a direct (even if not precise) correlation with the running time of the model [187]. There are also known problems related to FLOPs, mostly related to the fact that memory access time can be a more dominant factor in real implementations (see the "Trap of FLOPs" discussion in [188]).

So, while we shall adopt FLOPS for our comparison, we shall also investigate performance through more traditional tools, like Tensorboard, in order to gain confidence in the reliability of FLOPs-based assessments.

6.6.2 Architectures overview

In this section, we provide detailed descriptions of the several different neural network architectures we have been dealing with, each one inspired by a different article. For each of them, different possible configurations have been investigated, varying the number and dimension of layers, as well as the learning objectives. Moreover, since some of the techniques considered are not dependent on the encoder/decoder structure, we also tested a mix of different architectures, hyperparameter configurations, and optimization objectives.

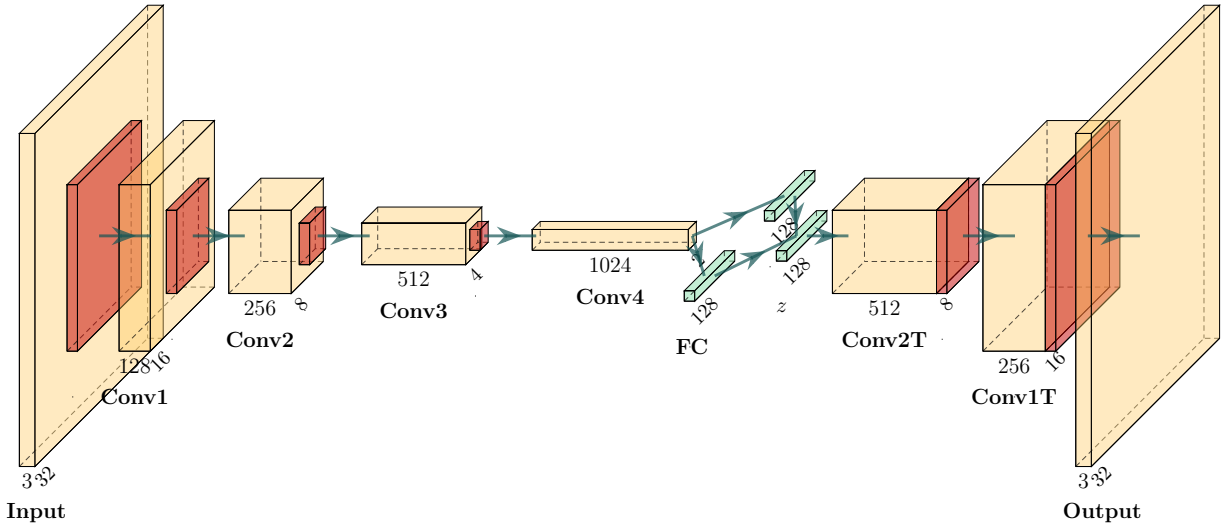


Figure 6.6: Graphical representation of the Vanilla VAE architecture. The yellow, orange, and green boxes represent convolutional, downsampling and dense layers, respectively.

Vanilla Convolutional VAE

In our first experiment, we followed the same structure of [172], which is a simple CNN architecture where we doubled the number of channels for each Convolution, and we downsampled the spatial dimension by 2 (see Figure 6.6).
The encoder is structured as follows. In the first layer, the input image of dimension $(n, n, 3)$ (where $n = 32$ and $n = 64$ in CIFAR10 and CelebA, respectively) was passed through a convolutional layer with 128 channels and stride equals 2, to obtain 128 images of dimension $(n/2, n/2)$. This operation is repeated for $256, 512, 1024$ channels. The result is flattened and passed through two Dense layers to obtain the mean and the variance of the latent variables.
The decoder has the same structure as the encoder, with Transposed convolutions and Upsample layers.
Each convolutional filter has kernel size 4 and ReLU activation function, except for the last layer of the decoder, where we used a sigmoid activation to ensure that the output is in the range $[0, 1]$.

Resnet-like

The Resnet-like architecture was adopted in [145]. The main difference of this network with respect to the Vanilla VAE is that, before downsampling, the input is processed by a so-called *Scale Block*, which is just a sequence of Residual Blocks. In turn, a Residual Block is an alternated sequence of BatchNormalization and Convolutional layers (with unit stride), intertwined with residual connections. The number of Scale Blocks at each scale of

the image pyramid, the number of Residual Blocks inside each Scale Block, and the number of convolutions inside each Residual Block are user-configurable hyperparameters.

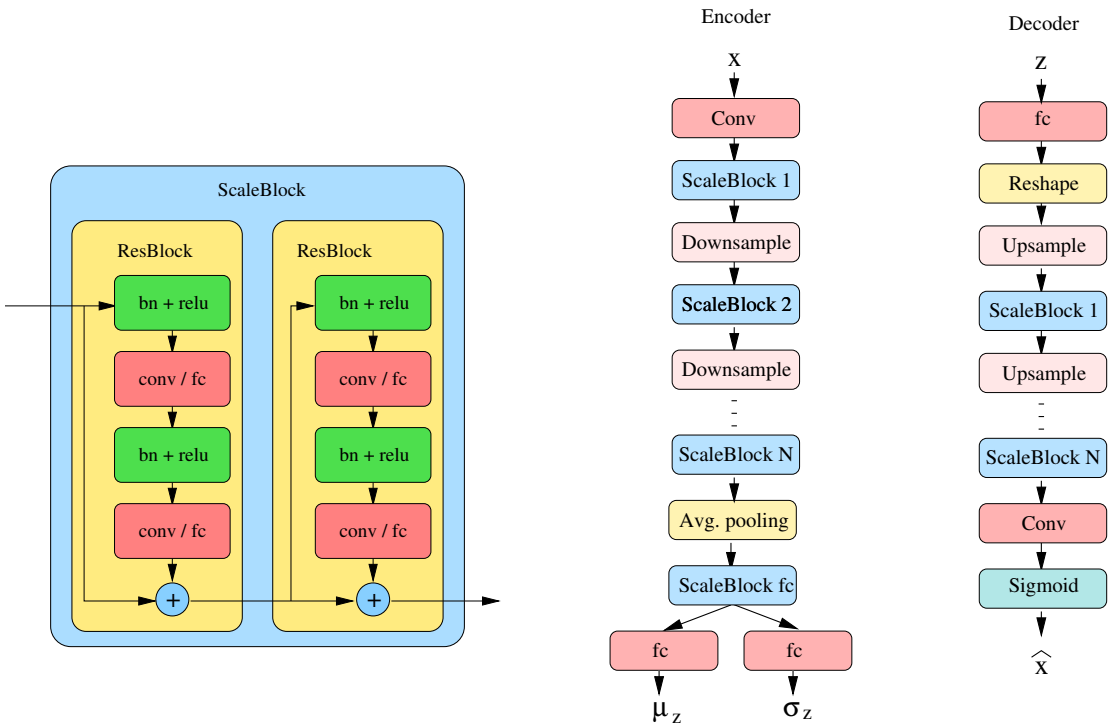


Figure 6.7: (a) Scale Block. The Scale Block is used to process features at a given scale; it is a sequence of Residual Blocks intertwined with residual connections. A Residual Block is an alternation of batch normalization layers, rectified linear units, and convolutions. (b) The input is progressively downsampled via convolutions with stride 2, intermixed by Scale Blocks; at a given scale, a global average pooling layer extracts features that are further processed via dense layers to compute mean and variance for latent variables. The decoder is essentially symmetric.

In the encoder, at the end of the last Scale Block, a global average level extracts spatial agnostic features. These are first passed through a so-called Dense Block (similar to a Residual Block but with dense layers instead of convolutions), and finally used to synthesize mean and variance for latent variables.

The decoder first maps the internal encoding $\boldsymbol{z}$ to a small map of dimension $4 \times 4 \times$ *base_dim* via a dense layer suitably reshaped. This is then up-sampled to the final expected dimension, inserting Scale Blocks at each scale.

Two-Stage VAE

To check to what extent the Two-Stage VAE improves the generation ability of a Variational Autoencoder, we tried to fit a second stage to every model we tested, following the architecture described in the following and graphically represented in Figure 6.7.
The encoder in the second stage model is composed of a couple of Dense layers of dimension 1536 and ReLU activation function, followed by a concatenation with the Input of the model and then by another Dense layer to obtain the latent representation u with the same dimension of z, following what's described in Section 6.3. The decoder has exactly the same structure as the encoder.
As already described, we used the *cosine similarity* as the reconstruction part of the ELBO objective function.
We observed that to improve the quality of the generation, the second stage should be trained for a large number of epochs.

Convolutional RAE

In our implementation of RAE, we followed exactly the same structure as in Convolutional Vanilla VAE, with the sole difference that, in RAEs, the latent space is composed of just one fully connected layer representing the variable z (see Figure 6.8).
In our tests, we only compared L_2 and GP regularization, with parameter λ heuristically computed to achieve the best performance.

NVAE

The model is organized in a bottom-up inference network and a top-down generative network (see Figure 6.9). Each one of the two networks is composed of a hierarchy of modules at different *scales*. Each scale is composed of *groups* of sequential (residual) blocks.

During generation, each module computes from the current input $\mathbf{h}_l$ a prior $p(\boldsymbol{z}_l|\mathbf{h}_l)$ ($\mathbf{h}_l$ depends from $\boldsymbol{z}_{<l}$): after sampling from this prior, the result is combined in some way with the current input $\mathbf{h}_l$, the two pieces of information are processed together and passed to the next module.

During inference, we extract the latent representation at stage l by synthesizing a mean and a standard deviation for $q(\boldsymbol{z}_l|\boldsymbol{x}, \mathbf{h}_l)$: since this information depends from $\mathbf{h}_l$, we expect to provide additional information, not already available by previous latent encodings. Moreover, the computation of $\mathbf{h}_l$, is done by the top-down network, that is hence a sub-network of the inference network. During training, both networks are trained together.

Each network has a hierarchical organization at different *scales*. Each scale is composed of *groups* of Blocks.

Both Encoder Blocks (EB) and Decoder Blocks (DB) have similar architectures and are essentially composed of an alternated sequence of BatchNormalization and Convolutional

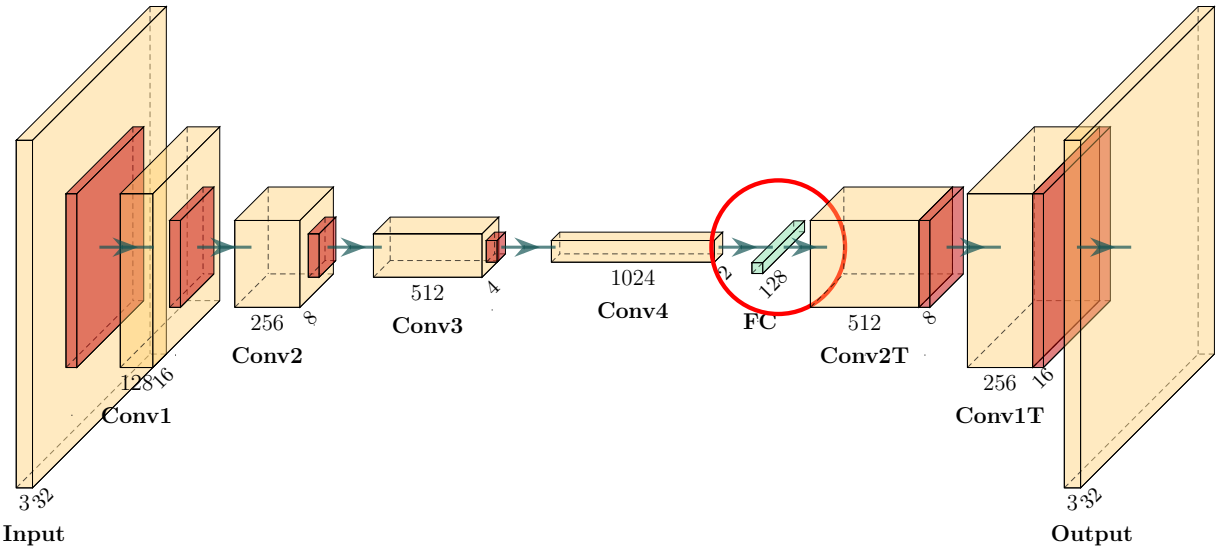


Figure 6.8: Graphical representation of the RAE architecture. The yellow, orange, and green boxes represent convolutional, downsampling and dense layers, respectively. The red circle underlines the sole architectural difference between our implementation of VanillaVAE and RAE, i.e. the fact that in the latter, the latent space is composed of a single Dense layer that directly encodes to z, while in VanillaVAE the encoding is performed by a couple of Dense layers that represents the mean and the variance of a Gaussian distribution.

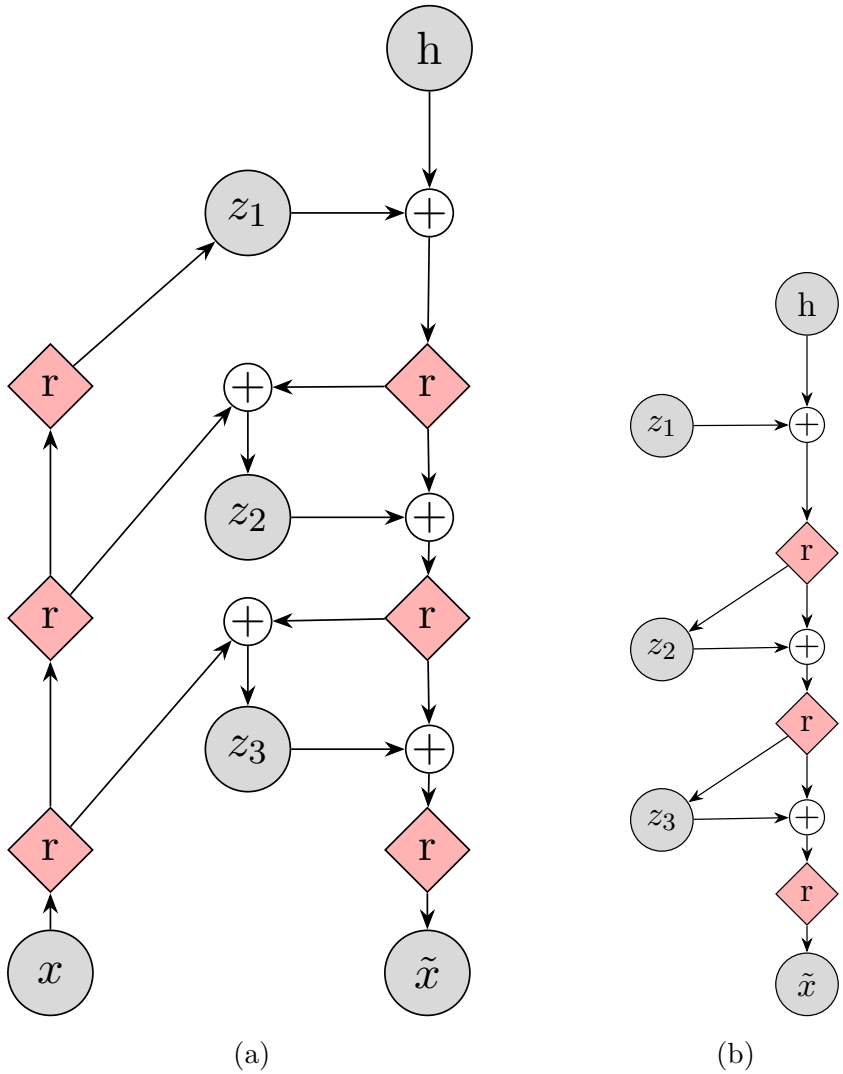


Figure 6.9: The whole NVAE architecture (a) and a focus on its decoder (b).

layers, separated by nonlinear activation layers, and intermixed with residual connections (so, very similar to the Scale Block discussed in the previous section). A few technical novelties are however introduced by the authors:

- the recent *Swish* activation function $f(u) = \frac{u}{1+e^{-u}}$ [189] is used instead of Relu, Elu, or other more traditional choices;
- a Squeeze-and-Excitation (SE) layer [190] is added at the end of each block;
- a moderate use of depthwise separable convolutions [191] is deployed in order to reduce the number of parameters of the network.

Table 6.1 gives a summary of hyperparameters used in training NVAE on the datasets addressed in this article, borrowed from [147]. D^2 indicates a latent variable with the spatial

dimensions of $D \times D$. As an example, the MNIST model consists of two scales: in the first one, we have five groups of $4 \times 4 \times 20$-dimensional latent variables: in the second one, we have 10 groups of $8 \times 8 \times 20$-dimensional variables.

Hyperparameter	**MNIST**	**Cifar10**	**CelebA**
Input size	28×28	32×32	64×64
Epochs	400	400	90
BatchSize	200	32	16
Normalizing Flows	0	2	2
Scales	2	1	3
Groups per Scale	5,10	30	5,10,20
Spatial dims of $\boldsymbol{z}$ per Scale	$4^2, 8^2$	16^2	$8^2, 16^2, 32^2$
Channel dims of z	20	20	20
Initial Channels in Enc.	32	128	64
Residual Cells per Group	1	2	2
GPUs	2	8	8
Total Train time (h)	21	55	92

Table 6.1: Summary of the hyperparameters used in the training of NVAE on the datasets used in the experiments.

The figures of merit in Table 6.1 help to understand the key novelty of NVAE, that is in the massive usage of space located latent variables. Consider for instance the case of Cifar10. The original input of dimension $32 \times 32 \times 3$ is first transformed to dimension $16 \times 16 \times 128$ and then, *without any further downscaling*, processed through a long sequence of residual cells (30×2). At each iteration, a huge number of latent variables ($16 \times 16 \times 20$) is extracted and used for the internal representation, which hence has a dimension widely larger than the input. Due to this fact, as is also observed by the authors in the appendix, it is not surprising that most of the variables will collapse during training.

Working with such a huge number of latent variables introduces a lot of issues; in particular, it becomes crucial to balance the KL component of variables belonging to different groups. To this aim, the authors introduce additional balancing coefficients γ_l to ensure that a similar amount of information is encoded in each group (see [147], appendix A):

$$D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) = \sum_{l=1}^{L} \gamma_l \mathbb{E}_{\boldsymbol{z}_{<l} \sim q(\boldsymbol{z}_{<l}|\boldsymbol{x})}[D_{KL}(q(\boldsymbol{z}_l|\boldsymbol{x}, \boldsymbol{z}_{<l})||p(\boldsymbol{z}_l|\boldsymbol{z}_{<l}))].$$

The balancing coefficient γ_l is kept proportional to the KL term for that group, in such a way to encourage the model to revive the latent variables in that group when KL is low, and to clip them if KL is too high. Additionally, γ_l is also proportional to the size of each group, to encourage the use of variables at lower scales.

NVAE architectures have a relatively small number of parameters, due to the extensive use of convolutions and depthwise separable convolutions; however, they require a massive amount of memory, and huge computational power: for the configuration used for Cifar10, composed of 30 groups at scale 16×16, we estimated the number of flops for the inference phase larger than 100G.

Due to this reasons, we experimented with a sensibly lighter architecture, just composed of 5 groups, with a few additional convolutions to augment the receptive fields of the spatially located latent variables. The good news is that the network, even in this severely crippled form, still seems to learn; however, results are really modest and below the performances of different networks with comparable complexity.

HFVAE

As we already remarked, the main novelty of NVAE is in the massive exploitation of a huge number of spatially located latent variables. In order to test the relevance of this architectural decision, we also tested a different variant of the hereditary architecture of Figure 6.9, where we drop the spatial dimension for latent variables, using instead a Featurewise Linear Modulation Layer [192] to modulate channels according to the internal representation. In addition, the first approximation $\mathbf{h}_1$ is directly produced from the latent variable set $\boldsymbol{z}_0$ through a dense transformation. The general idea is that at lower scales we decide the content of the resulting image, while stylistic details at different resolutions (usually captured in channels correlations [193]) are added at higher scales. We call this variant HFVAE (Hereditay Film VAE); a similar architecture has been investigated in [194].

6.7 Numerical results

In this section, we provide quantitative evaluations for some configurations of the models previously discussed. The precise configurations (layers, channels, blocks, etc.) are discussed below.

The datasets used for the comparison are CIFAR10 and CelebA: in a GreenAI perspective, we are reluctant to address more complex datasets, at higher resolutions, that would require additional computational resources and additional costs. On CelebA, we just evaluated a subset of particularly interesting models.

For each model, we provide the following figures:

- **Params**: the number of parameters;
- **FLOPs**: an estimation of number of FLOPS (see Section 6.6.1 for more details);
- **MSE**: the mean reconstruction error $\times 10^3$;

- **REC**: the FID value computed over *reconstructed* images;
- **GEN1**: the FID value computed over images generated by a first VAE;
- **GEN2**: the FID value computed taking advantage of a second VAE;
- **GMM**: the FID value computed by superimposing a GMM of ten Gaussians[1] on the latent space. In the case of hierarchical models, the GMM is computed on the innermost set of latent variables.

The following list provides a legenda for the names of models used in the following tables:

- **CNN-by-lz:** Vanilla VAE with CNN architecture, basedim of y channels and latent space of dimension $\boldsymbol{z}$.
- **L2-RAE-by-lz:** L_2-RAE with CNN architecture, basedim of y channels and latent space of dimension $\boldsymbol{z}$.
- **GP-RAE-by-lz:** GP-RAE with CNN architecture, basedim of y channels and latent space of dimension $\boldsymbol{z}$.
- **Resnet-sx-by-lz:** Resnet-like model, with x ScaleBlocks, a basedim of y channels, and a latent space of dimension z.
- **HFVAE-sx-by-lz:** HFVAE with x scales, ScaleBlocks, a basedim of y channels, and a latent space of dimension z at hereditary scales; the base latent dimension z_0 is 64.
- **NVAE-zx-by-lz:** NVAE with x latent variables channels, a basedim of y and z latent groups of the same scale.

6.7.1 Quality Evaluation

Here we draw a few conclusions about the design of Variational Autoencoders deriving from the previous investigation (Tables 6.2 and 6.3) and our past experience with VAEs.

- The decoder is more important than the encoder. For instance, in the ResNet architecture latent features are extracted via a GlobalAverage layer, obtaining robust features, less prone to overfitting.
- Working with a larger number of latent variables improves reconstruction, but this does not eventually imply better generation. This is e.g. evident when comparing the two Resnet-like architectures with latent spaces of dimension 128 and 100.

[1] Augmenting the number of Gaussians does not sensibly improve generation

Model	Params	FLOPs	MSE	REC	GEN1	GEN2	GMM
CNN-b128-l128	31,034,755	2,397M	2.8	27.6	96.2	96.8	89.0
L2-RAE-b128-l128	30,510,339	2,395M	**1.2**	**9.9**	108.1	88.4	78.2
GP-RAE-b128-l128	30,510,339	2,395M	**1.2**	10.6	118.0	97.6	**76.4**
Resnet-s4-b48-l128	16,179,363	1,431M	1.5	37.2	110.0	93.9	96.3
Resnet-s4-b48-l100	16,064,619	1,430M	1.6	37.5	102.9	88.4	91.4
Resnet-s4-b64-l64	27,766,275	2,539M	1.7	36.5	94.2	**78.8**	85.1
HFVAE-s4-z4-l48	27,139,755	**1,163M**	1.8	45.9	93.3	90.8	90.0
HFVAE-s4-z12-l64	48,113,051	2,085M	1.3	33.3	**89.0**	85.7	86.4
NVAE-z10-b100-l4	**8,305,521**	4,478M	3.2	62.6	96.1	87.4	91.4

Table 6.2: Summary of the results obtained with the networks in the first column on Cifar10.

Model	Params	FLOPs	MSE	REC	GEN1	GEN2	GMM
CNN-b128-l64	40,668,419	4,104M	3.2	48.4	66.9	56.2	55.2
L2-RAE-b128-l64	27,359,043	4,102M	3.3	39.8	230.2	61.7	45.1
GP-RAE-b128-l64	27,359,043	4,102M	3.2	41.2	230.6	65.3	47.0
Resnet-s4-b32-l64	**19,330,627**	**2,924M**	2.8	51.4	66.0	54.9	57.4
Resnet-s4-b48-l64	38,996,003	6,452M	**2.5**	46.8	61.7	50.8	54.5
Resnet-s3-b64-l64	21,370,179	5,949M	2.6	**39.2**	**59.3**	**44.9**	**45.8**

Table 6.3: Summary of the results obtained with the networks in the first column on CelebA.

- Fitting a GMM over the latent space [172] is a cheap technique (it just takes a few minutes) that invariably improves generation, both in terms of perceptual quality and FID score. This fact also confirms the mismatch between the prior and the aggregated posterior discussed in Section 6.2.3.

- The second stage technique [145] typically requires some tuning in order to properly work, but when it does it usually outperforms the GMM approach. Tuning may involve the loss function (we used cosine similarity in the following), the architecture of the second VAE, and the learning rate (more generally, the optimizer's parameters).

- Hierarchical architectures are complex systems, difficult to understand and to work with (monitoring/calibrating training is a really complex task). We cannot express an opinion about NVAE, since its complexity trespasses our modest computational facilities, but simpler architectures like those described in [180] or [181], in our experience,

do not sensibly improve generation over a well-constructed traditional VAE.

- The loss of variance for generated images [173] (see Section 6.2.4) is confirmed in all models, and it almost coincides with the mean squared error for reconstruction.

A qualitative comparison between the different models in generating images can also be done by looking at the images in the Appendix.

6.7.2 Energetic evaluation

Before comparing the energetic footprint of the different models, let us briefly discuss the notion of FLOPs as a measure of computational efficiency. FLOPs have been computed by a library for Tensorflow Keras under development at the University of Bologna, and inspired by similar works for PyTorch (see e.g. `https://github.com/sovrasov/flops-counter.pytorch`). FLOPs only provide an abstract, machine-independent notion of complexity; typically, only the most expensive layers are taken into account (those with superlinear complexity with respect to the size of inputs). The way this quantity will result in an actual execution time and energetic consumption does however largely depend on the underlying hardware, and the available parallelism. As an example, in Table 6.4 we compare the execution time for a forward step over the test set of Cifar10 (10K) for a couple of hardware configurations. The first one is a Laptop with an NVIDIA Quadro T2000 graphics card and a CPU Core i7-9850H; the second one is a workstation with an Asus GeForce DUAL-GTX1060-O6G graphic card and a CPU intel Core i7-7700K. Observe the strong dependency from the batch size, that is not surprising but worth recalling (see [195] for a critical analysis of the performance of Neural Networks architectures). Of course, as soon as we move the computation to a cloud, execution times are practically unpredictable.

Unfortunately, as we shall see, even for a *given* computational device, the relation between FLOPs and execution time is quite aleatory.

Following the traditional paradigm, we compare performances on the forward pass. This is already a questionable point; on one side, it is true that this reflects the final usage of the network when it is deployed in practical applications; on the other side, it is plausible to believe that training still takes a prevalent part of the lifetime of any neural network. Restricting the investigation to forward time means not taking into account some expensive techniques of the training of modern systems, such as regularization components. For example, it is possible to notice that in Table 6.5, L_2-RAE and GP-RAE have exactly the same number of FLOPs, since in terms of forward execution they are equal. However, we highlight that the training of GP-RAE is almost ten times slower than the training of L_2-RAE. This is a consequence of the fact that the regularization term of GP-RAE involves the computation of the decoder gradient with respect to the latent variables, which is an expensive operation not required in L_2-RAE. Consequently, even if the two models have more or less

Network	bs100	bs10	bs1
Resnet-s4-b48-l128	3.0 ± 0.2	6.0 ± 0.2	33.3 ± 0.4
	4.9 ± 0.2	8.8 ± 0.2	49.5 ± 0.5
Resnet-s4-b48-l100	2.86 ± 0.1	5.9 ± 0.2	32.9 ± 0.4
	4.8 ± 0.2	8.7 ± 0.2	49.1 ± 0.5
Resnet-s3-b64-l64	4.4 ± 0.2	$9. \pm 0.3$	47.8 ± 0.4
	7.2 ± 0.2	13.5 ± 0.2	78.6 ± 0.5

Table 6.4: Average Forward Time (in seconds) over the Cifar10 Test Set (10k images) for different networks, hardware, and batch size (bs). The two times entries in each cell refer to different machines: the first is a Laptop with an NVIDIA Quadro T2000 graphics card and a Core i7-9850H CPU; the second is a workstation with an Asus GeForce DUAL-GTX1060-O6G graphic card and an Intel Core i7-7700K CPU.

the same performance in terms of generation quality, L_2-RAE should be preferred, since its training is cheaper. Moreover, taking into account only the FLOPs of the model, the actual convergence speed of systems is neglected.

The results of the energetic evaluation on the forward pass are given in Table 6.5; inference times have been computed over a workstation with an Asus GeForceDUAL-GTX1060-O6G graphic card and an intel Core i7-7700K CPU. The same results have also been expressed in graphical form in Figure 6.10, relative to a batch size of dimension 1. In the plot, we omit L2-RAE and GP-RAE, since their architectures and figures are essentially analogous to the basic CNN; similar for some Resnet architectures.

As it is clear from these results, there is no precise correlation between FLOPS and execution time. As an example, from Table 6.5, we see that HFVAE requires a computation time 4-6 times higher than the others in the face of the lowest number of FLOPS. One of the possible reasons for this behavior is, in our opinion, the fact that the total computation time also includes memory access time in addition to FLOPS. As observed by several authors (see e.g. [188]), memory access time is a crucial factor in real implementations, as densely packed data might be read faster than a few numbers of scattered values. For instance, while depthwise convolutions greatly reduce the number of parameters and FLOPS, they require more fragmented memory access, harder to implement efficiently. In future works, we intend to deeper investigate other causes for the absence of correlation between FLOPS and time.

Model	Params	FLOPS	bs100	bs10	bs1
CNN-b128-l128	31,034,755	2,397M	5.8 ± 0.1	9.0 ± 0.1	54.1 ± 0.4
L2-RAE-b128-l128	30,510,339	2,395M	11.6 ± 0.2	13.9 ± 0.2	57.3 ± 0.5
GP-RAE-b128-l128	30,510,339	2,395M	12.5 ± 0.2	14.1 ± 0.2	56.3 ± 0.5
Resnet-s4-b48-l128	16,179,363	1,431M	4.9 ± 0.2	8.8 ± 0.2	49.5 ± 0.4
Resnet-s4-b48-l100	16,064,619	1,430M	4.8 ± 0.2	8.7 ± 0.2	49.1 ± 0.4
Resnet-s4-b64-l64	27,766,275	2,539M	7.2 ± 0.2	13.5 ± 0.2	78.6 ± 0.5
HFVAE-s4-z4-l48	27,139,755	1,163M	13.1 ± 0.2	29.2 ± 0.3	207.0 ± 1.1
HFVAE-s4-z12-l64	48,113,051	2,085M	21.3 ± 0.3	48.7 ± 0.4	325.2 ± 1.6

Table 6.5: Average Forward Time (in seconds) over the Cifar10 Test Set (10k images) for different architectures and different batch size (bs); times refer to a workstation equipped with an Asus GeForceDUAL-GTX1060-O6G GPU and an Intel Core i7-7700K CPU.

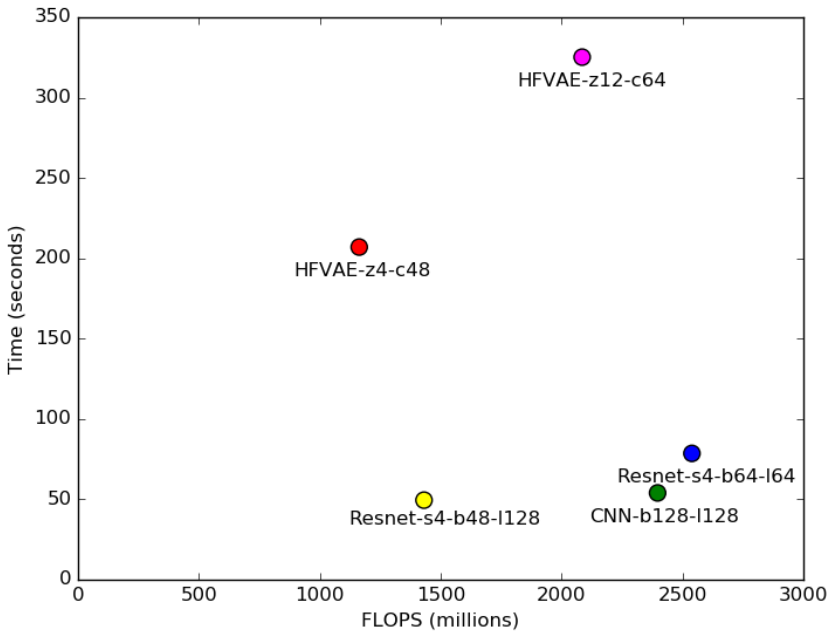


Figure 6.10: FLOPs versus Execution time. From the plot we can evince the little relation between the two figures but, possibly, at a magnitude level.

6.8 Conclusions

In this chapter, we presented a critical survey of recent variants of Variational Autoencoders, referring them to the several problems that still hinder this generative paradigm. In view of the emerging GreenAI paradigm [2], we also focused attention on the computational cost of

the different architectures. The main conclusions of our investigation are given in Section 6.7.1, and we shall not try to summarize them here; we just observe that, while we strongly support the GreenAI vision, we must eventually find better metrics than FLOPs to compare the energetic performance of neural networks, or more realistic way to compute them.

The constant improvement in generative sampling during the last few years is very promising for the future of this field, suggesting that state-of-the-art generative performance can be achieved or possibly even improved by carefully designed VAE architectures.

Chapter 7

Image embedding for denoising generative models

Denoising Diffusion Models (DDM) [127] are rapidly imposing as the new state-of-the-art technology in the field of deep generative modeling, challenging the role held so far by Generative Adversarial Networks [196]. The impressive text-to-image generation capability shown by models like DALL·E [128] and Imagen [197], recently extended to videos in [198], clearly proved the qualities of this technique, comprising excellent image synthesis quality, good sampling diversity, high sensibility and easiness of conditioning, stability of training and good scalability.

In very rough terms, a diffusion model trains a single network to denoise images with a parametric amount of noise and generates images by iteratively denoising pure random noise. This latter process is traditionally called *reverse diffusion* since it is meant to "invert" the *direct diffusion* process consisting in adding noise. In the important case of Implicit Diffusion models [138], reverse diffusion is deterministic, but obviously not injective: many noisy images can be denoised to a single common result. Let us call $emb(\boldsymbol{x})$ (embedding of $\boldsymbol{x}$) the set of points whose reverse diffusion generates $\boldsymbol{x}$. The problems we are interested in are investigating the shape of $emb(\boldsymbol{x})$ (e.g. is it a connected, convex space?), finding a "canonical" element in it (i.e. a sort of center of gravity), and, in case such a canonical element exists, finding an efficient way to compute it. This would allow us to embed an arbitrary image into the "latent" space of a diffusion model, providing functionality similar to GAN-recoders [199], or to encoders in the case of Variational AutoEncoders [4, 125].

Contributions This Chapter is based on the publication [11], where we aim to study the embedding space of diffusion models. In particular, we focus on Non-Markovian Diffusion Models known as DDIM, and we show that the associated embedding space has some remarkable properties, such as unicity with respect to training and model architecture, and we show that elements of $emb(\boldsymbol{x})$ can be efficiently computed by gradient descent. After

that, we introduce a neural network, named *embedding network*, that maps any image $\boldsymbol{x}_0$ to its *canonical embedding*, i.e. to a representative element in $emb(\boldsymbol{x}_0)$. Extensive experiments on the embedding network show its ability to generate elements of the embedding space, opening to chances for a deeper study.

My personal contribution to the work was to analyze and explain the theory, and to write down some of the experiments proposed. The source code of the experiments described in this Chapter is available at the GitHub repository https://github.com/asperti/Embedding-in-Diffusion-Models, along with links to weights for pre-trained models.

Structure of the Chapter The Chapter is structured as follows. Section 7.1 is devoted to formally introducing the notion of Denoising Diffusion Models, in addition to the deterministic variant of Denoising Diffusion Implicit Models we are particularly interested in. In the same Section, we also discuss an intuitive interpretation of denoising diffusion models in terms of a "gravitational analogy" (Section 7.1.3), which drove many of our investigations and plays an important role in understanding the structure of datapoint embeddings. A major consequence of this interpretation, which to the best of our knowledge has never been pointed out before, is the *invariance of the latent space with respect to different models*: a given seed, passed as input to different models, always produces the same image. In Section 7.2, we provide architectural details about our implementation of the Denoising Diffusion model. Our methodology to address the embedding problem is discussed in Section 7.3. Two main approaches have been considered, one based on a gradient descent technique, which allows us to synthesize large clouds of different seeds in the embedding space of specific data points (Section 7.3.2), and another one based on training a neural network to compute a single "canonical" seed for the given image: essentially, a sort of encoder. Conclusions and future works are discussed in Section 7.4.

7.1 Denoising Diffusion Models

In this Section, we provide a general overview of diffusion models from a mathematical perspective. All results in Section 7.1.1 and Section 7.1.2 are known in the literature; in Section 7.1.3 we propose an original interpretation of the reverse diffusion process in terms of a gravitational collapse of the latent space over the data manifold.

7.1.1 Diffusion and reverse diffusion

Consider the distribution $p_{gt}(\boldsymbol{x}_0)$ from which the data is generated. We recall that Denoising Diffusion Probabilistic Models (DDPM) [127], being a DLVM as described in Section 5.1.2, aim to approximate $p_{gt}(\boldsymbol{x})$ by a generative distribution $p_\Theta(\boldsymbol{x}_0)$, which is assumed to have

the form

$$p_\Theta(\boldsymbol{x}_0) = \int p_\Theta(\boldsymbol{x}_{0:T}) d\boldsymbol{x}_{1:T} \tag{7.1}$$

for a given time range horizon $T > 0$, where

$$p_\Theta(\boldsymbol{x}_{0:T}) = p_\Theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\Theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{7.2}$$

with $p_\Theta(\boldsymbol{x}_T) = \mathcal{N}(0, I)$ and $p_\Theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\mu_\Theta(\boldsymbol{x}_t, \alpha_t), \sigma_t^2 I)$. We also recall that the diffusion model $p_{gt}(\boldsymbol{x}_{0:T})$ is considered to be a Markov chain of the form

$$p_{gt}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\boldsymbol{x}_{t-1}, \left(1 - \frac{\alpha_t}{\alpha_{t-1}}\right) I\right), \tag{7.3}$$

with $\{\alpha_t\}_{t\in[0,T]}$ being a decreasing sequence in the interval $[0, 1]$. Note that, differently from the introduction done in Section 5.1.2, we renamed

$$\beta_t = \frac{\alpha_t}{\alpha_{t-1}}. \tag{7.4}$$

The parameters of the generative model $p_\Theta(\boldsymbol{x}_0)$ are then trained to fit $p_{gt}(\boldsymbol{x}_0)$ by minimizing the negative Evidence Lower Bound (ELBO) loss, defined as

$$\mathcal{L}(\Theta) = -\mathbb{E}_{\boldsymbol{x}\sim p_{gt}(\boldsymbol{x}_{0:T})}[\log p_\Theta(\boldsymbol{x}_{0:T}) - \log p_{gt}(\boldsymbol{x}_{1:T})]. \tag{7.5}$$

The ELBO loss can be rewritten in a computable form by noticing that, as a consequence of Bayes' Theorem, $p_{gt}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0), \sigma_q^2 I)$. Consequently,

$$\mathcal{L}(\Theta) = \sum_{t=1}^{T} \gamma_t \mathbb{E}_{\boldsymbol{x}_t\sim p_{gt}(\boldsymbol{x}_t|\boldsymbol{x}_0)}\Big[\|\mu_\Theta(\boldsymbol{x}_t, \alpha_t) - \tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0)\|_2^2\Big], \tag{7.6}$$

which can be interpreted as the weighted mean squared error between the reconstructed image from $p_\Theta(\boldsymbol{x}_t|\boldsymbol{x}_0)$ and the true image obtained by the reverse diffusion process $p_{gt}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ for each time t.

In [138], the authors considered a non-Markovian diffusion process

$$p_{gt,\sigma}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = p_{gt,\sigma}(\boldsymbol{x}_T|\boldsymbol{x}_0) \prod_{t=2}^{T} p_{gt,\sigma}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0), \tag{7.7}$$

where $p_{gt,\sigma}(\boldsymbol{x}_T|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_T}\boldsymbol{x}_0, (1 - \alpha_T)I)$, and

$$p_{gt,\sigma}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\Big(\mu_{\sigma_t}(\boldsymbol{x}_0, \alpha_{t-1}), \sigma_t^2 I\Big) \tag{7.8}$$

with

$$\mu_{\sigma_t}(\boldsymbol{x}_0, \alpha_{t-1}) = \sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}. \tag{7.9}$$

This construction implies that the forward process is no longer Markovian, since it depends both on the starting point $\boldsymbol{x}_0$ and on $\boldsymbol{x}_{t-1}$. Moreover, [138] proved that, with this choice of $p_{gt,\sigma}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$, the marginal distribution $p_{gt,\sigma}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)I)$, recovers the same marginals as in DDPM, which implies that $\boldsymbol{x}_t$ can be diffused from $\boldsymbol{x}_0$ and α_t by generating a realization of normally distributed noise $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, I)$ and defining

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t. \tag{7.10}$$

Note that when in Equation (7.8) $\sigma_t = 0$, the reverse diffusion $p_{gt,\sigma}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ becomes deterministic. With such a choice of σ_t, the resulting model is named Denoising Diffusion Implicit Models (DDIM) by the authors in [138]. Interestingly, in DDIM, the parameters of the generative model $p_\Theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ can be simply optimized by training a neural network $\boldsymbol{\epsilon}_\Theta^{(t)}(\boldsymbol{x}_t, \alpha_t)$ to map a given $\boldsymbol{x}_t$ to an estimate of the noise $\boldsymbol{\epsilon}_t$ added to $\boldsymbol{x}_0$ to construct $\boldsymbol{x}_t$ as in (7.10). Consequently, $p_\Theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ becomes a $\delta_{f_\Theta^{(t)}}$, where

$$f_\Theta^{(t)}(\boldsymbol{x}_t, \alpha_t) = \frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\Theta^{(t)}(\boldsymbol{x}_t, \alpha_t)}{\sqrt{\alpha_t}}. \tag{7.11}$$

Intuitively, the network in (7.11) is just a denoiser that takes as input the noisy image $\boldsymbol{x}_t$ and the variance of the noise α_t and returns an estimate of the denoised solution $\boldsymbol{x}_0$. In DDIM, one can generate new data by first considering random Gaussian noise $\boldsymbol{x}_T \sim p_\Theta(\boldsymbol{x}_T)$ with $\alpha_T = 1$. Then, $\boldsymbol{x}_T$ is processed by $f_\Theta^{(T)}(\boldsymbol{x}_T, \alpha_T)$ to generate an estimation of $\boldsymbol{x}_0$, which is then corrupted again by the reverse diffusion $p_{gt}(\boldsymbol{x}_{T-1}|\boldsymbol{x}_T, f_\Theta^{(T)}(\boldsymbol{x}_T, \alpha_T))$. This process is repeated until a new datum $\boldsymbol{x}_0$ is generated by $f_\Theta^{(1)}(\boldsymbol{x}_1, \alpha_1)$.

The sampling procedure of DDIM generates a trajectory $\{\boldsymbol{x}_T, \boldsymbol{x}_{T-1}, \dots, \boldsymbol{x}_0\}$ in the image space. In [200, 201] the authors found that the (stochastic) mapping from $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$ in DDPM follows a score-based stochastic differential equation (SDE), where the dynamic is governed by terms related to the gradient of the ground-truth probability distribution from which the true data is generated. The sampling procedure for DDIM can be obtained by discretizing the deterministic *probability flow* [200] associated with this dynamics. Consequently, training a DDIM model leads to an approximation of the score function of the ground-truth distribution.

7.1.2 The Diffusion Schedule

An important aspect in implementing diffusion models is the choice of the diffusion noise $\{\alpha_t\}_{t=1}^T$, defining the mean and the variance of $p_{gt}(\boldsymbol{x}_t|\boldsymbol{x}_0)$. In [127], the authors showed that

the diffusion process $p_{gt}(\boldsymbol{x}_t|\boldsymbol{x}_0)$ converges to a normal distribution if and only if $\alpha_T \approx 0$. Moreover, to improve the generation quality, α_t has to be chosen such that it slowly decays to 0. The specific choice for the sequence α_t defines the so-called *diffusion schedule*.

In [127], the authors proposed to use linear or quadratic schedules. This choice was criticized in [202, 203] since it exhibits a too steep decrease during the first time steps, causing difficulties during generation for the neural network model. To remedy this situation, alternative scheduling functions with a gentler decrease have been proposed in the literature, such as the *cosine* or *continuous cosine* schedule. The behavior of all these functions is compared in Figure 7.1.

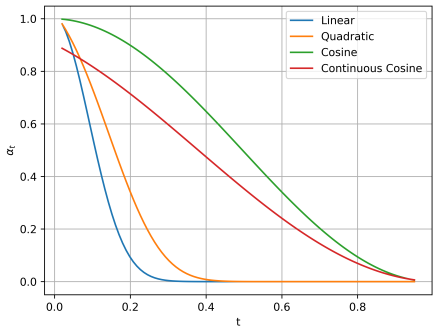


Figure 7.1: Comparison of different schedules. Generation is better if noise variance does not change too abruptly, so cosine and continuous cosine schedules usually work better than linear or quadratic ones.

The quantity of noise added by each schedule is also represented in Figure 7.2, where a single image is injected with increasing noise according to the given schedule. It is not hard to see that the cosine and the continuous cosine schedules exhibit a more uniform transaction between the original image and the pure noise.

7.1.3 The Gravitational Analogy

Similarly to other generative models, developing an intuition of the actual behavior of diffusion models (and of the mapping from a latent encoding to its visible outcome) can be challenging. In this Section, we propose a simple gravitational analogy that we found extremely useful to get an intuitive grasp of these models, and which suggested us some interesting conjectures about the actual shape of the embedding clouds for each object.

Simply stated, the idea is the following. You should think of the datapoints as corps with a gravitational attraction. Regions of the space where the data manifold has high probability are equivalent to regions with high density. The denoising model essentially learns the gravitational map induced over the full space: any single point of the space gets mapped to the point where it would naturally "land" if subject to the "attraction" of the

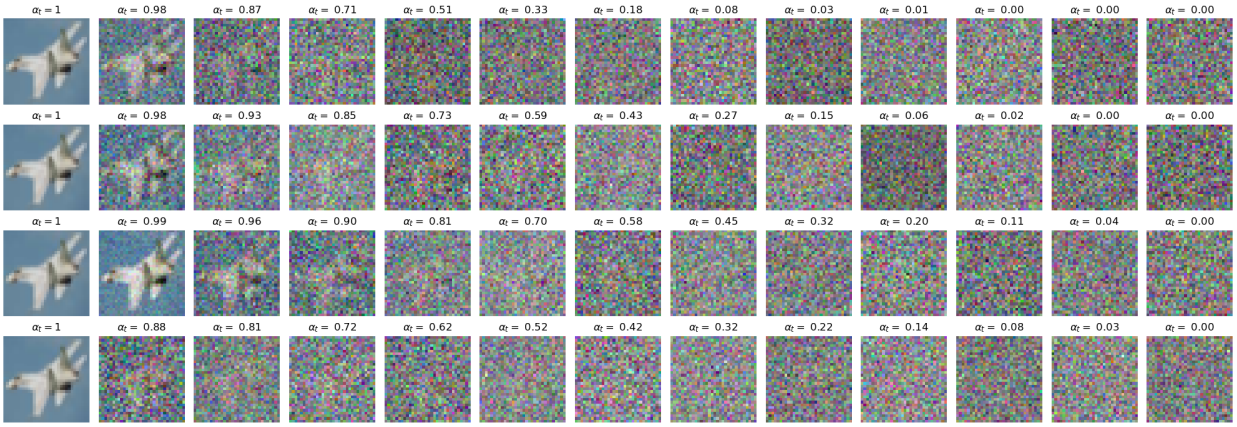


Figure 7.2: Increasing noise added by the different scheduling: in order, from top to bottom, linear, quadratic, cosine, and continuous cosine schedules. For each row, from left to right, the time t is increased linearly from 0 to T. The corresponding α_t for each schedule and for any t is reported above each image.

data manifold.

In more explicit terms, *any* point $\boldsymbol{z}$ of the space can be seen as a noisy version of *any* point $\boldsymbol{x}$ in the dataset. The "attraction" exerted by $\boldsymbol{x}$ on $\boldsymbol{z}$ (i.e. the loss) is directly proportional to their distance, usually an absolute or quadratic error. However, the probability of training the network to reconstruct $\boldsymbol{x}$ from $\boldsymbol{z}$ has a Gaussian distribution $\mathcal{N}(\boldsymbol{x}, \sigma_z^2 I)$, with σ_z^2 depending on the denoising step. Hence, the *weighted attraction* exerted by $\boldsymbol{x}$ on $\boldsymbol{z}$ at each step is

$$\mathcal{G}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{x},\sigma_z^2 I)}\left[\|\boldsymbol{x} - \boldsymbol{z}\|_1\right] \tag{7.12}$$

To get a grasp of the phenomenon, in Figure 7.3 we compare the gravitational low for a corp $\boldsymbol{x}$ with the *weighted attraction* reported in Equation (7.12), under the assumption that the variance σ has to be compared with the radius of the corp (with constant density, for simplicity).

According to the gravitational analogy, the embedding space $emb(\boldsymbol{x})$ of each datapoint $\boldsymbol{x}$ should essentially coincide with the set of points in the space corresponding to trajectories ending in $\boldsymbol{x}$. We can study this hypothesis on synthetic datasets. In Figure 7.4 we show the gravitational map for the well-known "circle" (a) and "two moons" datasets (b); examples of embeddings are given in figures (c) and (d).

From the pictures, it is clear in which way the model "fills the space", that is associating to each datapoint $\boldsymbol{x}$ all "trajectories" landing in $\boldsymbol{x}$. The trajectories are almost straight and oriented along directions orthogonal to the data manifold. We believe that this behavior can be formally understood by exploiting the dynamics of the trajectories introduced in [200], as mentioned in Section 7.1. We aim to deeply investigate those aspects in future work.

The most striking consequence of the "gravitational" interpretation is, however, the in-

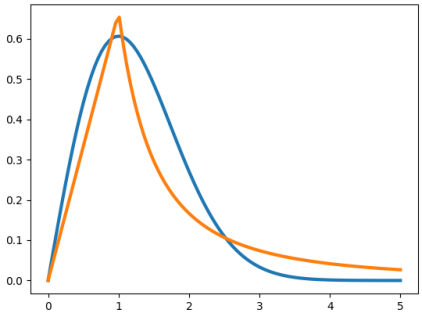


Figure 7.3: Gravitational analogy. The orange line is the usual gravitational low for a body with a radius of 1 and constant density. The blue line is the weighted attraction $\mathcal{G}(\boldsymbol{x})$ as a function of $\boldsymbol{x}$. The two lines have been rescaled to have an equal integral.

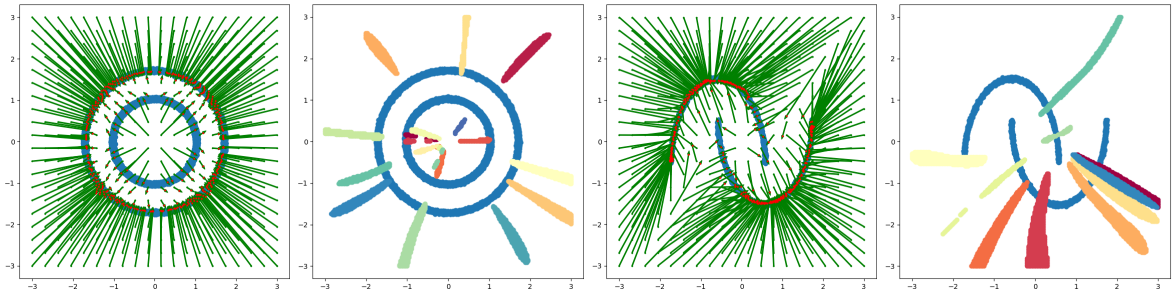
dependence of the latent encoding from the neural network or its training: the gravitational map only depends on the data manifold and it is unique, so distinct networks or different trainings of the same network, if successful, should eventually end up with the same results. This seems miraculous: if we pick a random seed in an almost immense space, and pass it as input to two diffusion (deterministic) models for the same dataset, they should generate essentially *identical* images.

We experimentally verified and confirmed the previous property on a large number of variants of generative diffusion models and different datasets (see Figure 7.5 for some results relative to CIFAR10, MNIST and Oxford Flowers). In particular, we tested different variants of the U-Net, with different numbers of downsampling blocks, different channel dimensions, and different layers in each block. We also optionally added different kinds of attention layers, in the traditional spatial form, or acting on channels like in squeeze-and-excitation layers [190] or in the recent NAFNet [56].

Provided generative models produce acceptable samples, the average quadratic distance between images generated by different generators on the same latent seed is always very small: typically, two to three orders of magnitude smaller than the average quadratic distance between random samples.

The fact that the same encoding works for different models seems to be peculiar to this kind of generative models. In [204], it was observed that we can essentially pass from a latent space to another of different generative models with a simple *linear map*: however, an identity or even a permutation of latent variables does not usually suffice[1].

[1]It remains to be checked if imposing a spatial structure to the latent space of GANs and VAEs is enough to induce uniqueness in that case too.



(a) circles (b) two moons (c) circle (d) two moons

Figure 7.4: Gravitational map and embeddings for the "circles" (a,b) and "two moons" (c,d) datasets. Datapoints are in blue. We consider a dense grid of seeds in the latent space, depicted in green. To visualize the maps (a) and (c) we draw an arrow pointing from each seed to the corresponding point generated by reverse diffusion (in red). To visualize embeddings (b) and (d) we consider a set of elements in the datasets, and for each element $\boldsymbol{x}$ we consider all points in the grid generating a sample $\hat{\boldsymbol{x}}$ sufficiently close to $\boldsymbol{x}$.

7.2 Denoising Architecture

The pseudocodes explaining training and sampling for diffusion models are respectively given in Algorithms 9 and 10 below.

Algorithm 9 Training

repeat
  $\boldsymbol{x}_0 \sim p_{gt}(\boldsymbol{x}_0)$
  $t \sim$ Uniform(1,..,T)
  $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
  Take gradient descent step on $||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\Theta(\sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \alpha_t)||^2$
until converged

Algorithm 10 Sampling

$\boldsymbol{x}_T \sim \mathcal{N}(0, I)$
for $t = T, ..., 1$ **do**
  $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}_\Theta(\boldsymbol{x}_t, \alpha_t)$
  $\tilde{\boldsymbol{x}}_0 = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}}\boldsymbol{\epsilon})$
  $\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\tilde{\boldsymbol{x}}_0 + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}$
end for

As a denoising network, it is quite standard to consider a conditional variant of the U-Net. This is a very popular network architecture originally proposed for semantic segmentation [55] and subsequently applied to a variety of image manipulation tasks. In general, the network is structured with a downsample sequence of layers followed by an upsample sequence, with skip connections added between the layers of the same size.

To improve the sensibility of the network to the noise variance, α_t is taken as input, which is then embedded using an ad-hoc sinusoidal transformation by splitting the value in a set of frequencies, in a way similar to positional encodings in Transformers [205]. The embedded noise variance is then vectorized and concatenated to the noisy images along the channel axes before being passed to the U-Net. This can be done for each convolution blocks

Figure 7.5: Uniqueness of the generative model. Different diffusion models generate essentially identical images when fed with the same seed. The two models in the picture are different versions of the U-Net: the first one has three downsampling blocks with channels $[32, 64, 128]$, and the second one has four dowsampling blocks with channels $[48, 96, 192, 384]$. The training sets are CIFAR10 (top), MNIST (middle), and Oxford Flowers 102 (bottom). Seeds have been randomly generated.

separately, or just at the starting layer; we adopted the latter solution due to its simplicity and the fact that it does not seem to entail any loss in performance.

Having worked with a variety of datasets, we used slightly different implementations of the previously described model. The U-Net is usually parameterized by specifying the number of downsampling blocks, and the number of channels for each block; the upsampling structure is symmetric. The spatial dimension does not need to be specified, since it is inferred from the input. Therefore, the whole structure of a U-Net is essentially encoded in a single list such as [32, 64, 96, 128] jointly expressing the number of downsampling blocks (4, in this case), and the respective number of channels (usually increasing as we decrease the spatial dimension).

For our experiments, we have mainly worked with two basic architectures, mostly adopting [32, 64, 96, 128] for simple datasets such as MNIST of Fashion MNIST, and using more complex structures such as [48, 96, 192, 384] for CIFAR10 or CelebA. We also used different U-Net variants to extensively test the independence of the latent encoding discussed in Section 7.1.3.

7.3 Embedding

We experimented with several different approaches for the embedding task. The most effective ones have been the direct synthesis through gradient descent, and the training of ad-hoc neural networks. Both techniques have interesting aspects worth discussing.

The gradient descent technique is intrinsically non-deterministic, producing a variegated set of "noisy" versions of a given image $\boldsymbol{x}$, all able to reconstruct $\boldsymbol{x}$ via reverse diffusion. The investigation of this set allows us to draw interesting conclusions on the shape of $emb(\boldsymbol{x})$.

Gradient descent is, however, pretty slow. A direct network can be trained to compute a single element inside $emb(\boldsymbol{x})$. Interestingly enough, this single element seems to be very close to the *average* of all noisy versions of $\boldsymbol{x}$ synthesized by the previous technique, suggesting evidence of its "canonical" nature.

The two techniques will be detailed in the following subsections.

7.3.1 Gradient Descent Synthesis

In Section 7.1.3, we computed the shape of embeddings for a few synthetic datasets by defining a dense grid of points in the latent space and looking for their final mapping through the reverse denoising process. Unfortunately, the number of points composing the grid grows exponentially in the number of features, and the technique does not scale to more complex datasets.

A viable alternative is the gradient descent approach, where we synthesize inputs starting from random noise, using the distance from a given target image as the objective function. In particular, given a sample $\boldsymbol{x}_0 \in \mathbb{R}^n$, we propose solving the minimization problem

$$\min_{\boldsymbol{x}_T \in \mathbb{R}^d} \frac{1}{2} \| f_\Theta(\boldsymbol{x}_T, \{\alpha_t\}_{t\in[0,T]}) - \boldsymbol{x}_0 \|_2^2 \tag{7.13}$$

where $f_\Theta(\boldsymbol{x}_T, \{\alpha_t\}_{t\in[0,T]})$ models the sampling process described above with schedule $\{\alpha_t\}_{t\in[0,T]}$. Due to the non-convex nature of (7.13), the obtained solution strongly depends on the starting guess that initializes the optimization algorithm. Thus, by repeating the procedure above with different starting guesses $\boldsymbol{x}_T^0 \sim \mathcal{N}(0, I)$, we were able to obtain multiple samples from $emb(\boldsymbol{x}_0)$.

Generation usually requires several thousand steps, but it can be done in parallel on a batch of inputs. This allows us to compute, within a reasonable time, a sufficiently large number of samples in $emb(\boldsymbol{x})$ for any given $\boldsymbol{x}$ (Figure 7.6). Having a full cloud of data, we can use standard techniques like PCA to investigate its shape, as well as to study how the image changes when moving along the components of the cloud (see Section 7.3.1). For PCA investigations we need an embedding cloud with a dimension larger than the dimension of the latent space. We typically worked with clouds of 2,000 points for MNIST and Fashion MNIST and 4,000 points for CIFAR10.

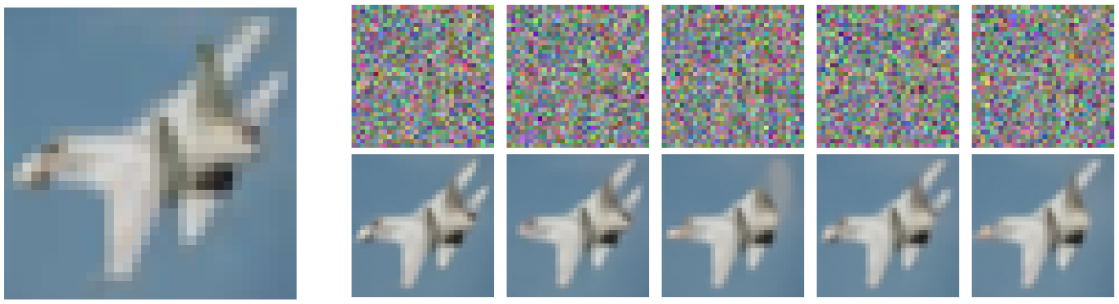


Figure 7.6: Examples of seeds in the latent space. The image on the left is the original. On the right, we see 5 different seeds and their corresponding generations through the reverse diffusion process.

A first interesting observation is that the average Euclidean distance among samples in $emb(\boldsymbol{x})$ is typically very high, around 0.9: they *are not* concentrated is a small portion of the latent space. However, they seem to occupy a convex region. In Figure 7.7 we show images obtained by reverse diffusion from 100 random *linear combinations* of seeds belonging to the embedding of the image on the left: all of them result in very similar reconstructions of the starting image.

Figure 7.7: Linear combination of seeds. Given the original image (1 and 3) we compute by gradient descent a large cloud of seeds (4K) in its embedding. Then, we compute 100 *internal points*, as a 1-sum random linear combination of the given seeds. Images 2 and 4 contain the results of these linear combinations. All generated images are similar between each other and are very close to the original image. Therefore, all internal points seem to belong to the embedding.

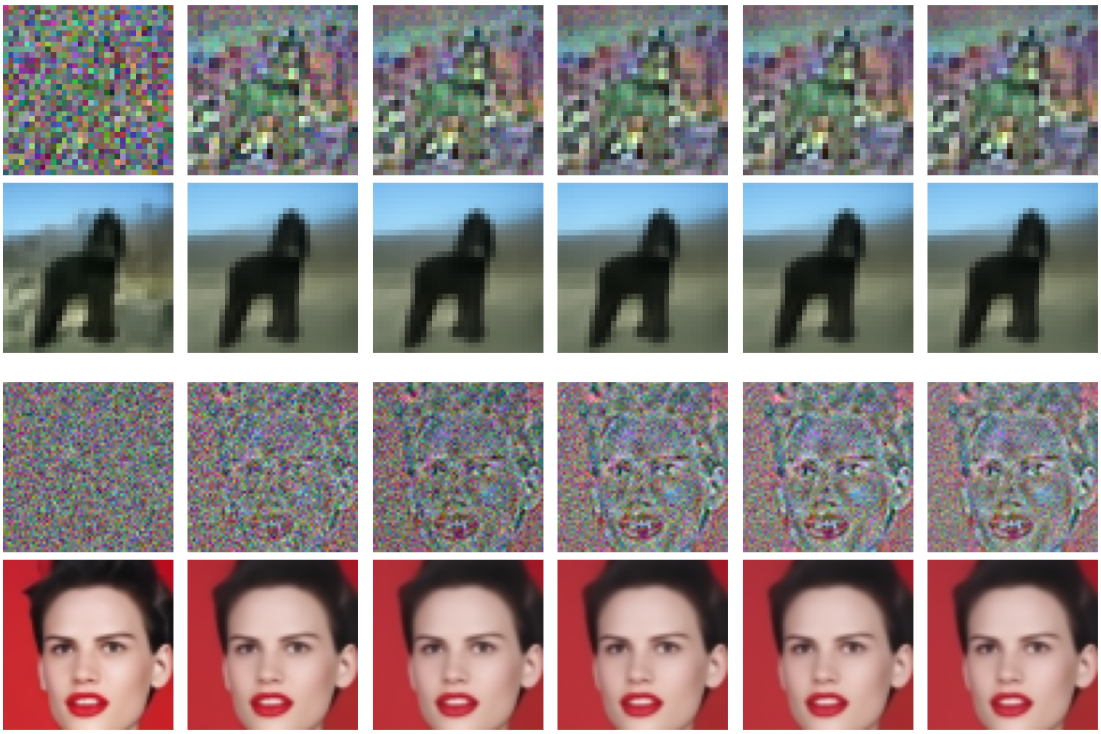


Figure 7.8: Progressive averaging in CIFAR10 and CelebA. The first row shows seeds computed as the mean of a progressive number of seeds in $emb(\boldsymbol{x})$, in a linear progression between 1 and 16
; the second row shows their respective output through the reverse denoising process. The output is very similar. Additionally, observe that the original image becomes identifiable in the seeds, even averaging a relatively small number of samples.

Due to the convexity of the space, its mean is also comprised in it. In Figure 7.8 we see the reconstructions obtained by considering as seed the average of a progressive number of seeds. The resulting images stabilize soon, although the result is slightly more blurry compared to using a single seed. The seeds on the borders of $emb(\boldsymbol{x})$ seem to provide slightly better reconstructions than internal points (which makes the quest for a "canonical", high-quality seed even more challenging).

PCA Decomposition

Principal Component Analysis allows us to fit an ellipsoid over the cloud of datapoints, providing a major tool for investigating the actual shape of embeddings. According to the "gravitational" intuition exposed in Section 7.1.3, $emb(\boldsymbol{x})$ should be elongated along directions orthogonal to the data manifold: moving along those directions should not sensibly influence generation, which should instead be highly affected by movements along minor components. Moreover, since the data manifold is likely oriented along a relatively small

number of directions (due to the low dimensionality of the manifold), we expect that most PCA components in each cloud will be orthogonal to the manifold, and have relatively high eigenvalues.

For instance, in the case of the clouds of seeds for CIFAR10, eigenvalues along all 3072 components typically span between 0.0001 and 4. We observe significant modifications only moving along the minor components of the clouds: in fact, they provide the shortest way to leave the embedding space of a given point. However, as soon as we leave the embedding space of $\boldsymbol{x}$ we should enter the embedding space of some "adjacent" point $\boldsymbol{x}'$. In other words, the minor components should define directions *inside* the data manifold, and possibly have a "semantical" (likely entangled) interpretation.

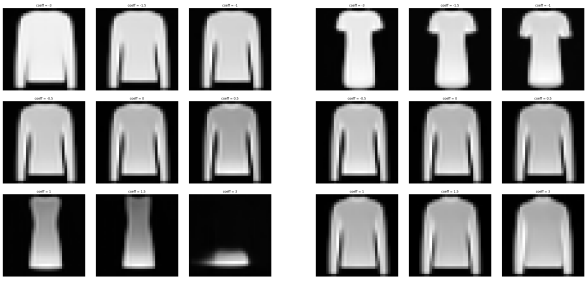


Figure 7.9: Fashion MNIST: movements along directions with minimal eigenvalues. The two groups of images refer to different components: starting from a mean seed generating the image in the middle, we move along a given component by the indicated positive or negative factor of the normalized eigenvector. Observe the progressive change in intensity and shape.

7.3.2 Embedding Networks

The second approach consists in training a neural network to directly compute a sort of "canonical" embedding for each image of the data manifold. The network takes as input an image $\boldsymbol{x}$ and produces a seed $\boldsymbol{z_x} \in emb(\boldsymbol{x})$; the loss function used to train the network is simply the distance between $\boldsymbol{x}$ and the result $\hat{\boldsymbol{x}}$ of the denoising process starting from $\boldsymbol{z_x}$.

We tested several different networks; metrics relative to the most significant architectures are reported in Table 7.1. A visual comparison of the behavior of the different networks is given in Figure 7.10, relative to CIFAR10. More examples on CelebA are given below. We started our investigation with a very simple network: a single convolution with a 5×5 kernel. The reason for this choice is that, according to the discussion we made in the introduction and the visualization of the mean element of the embedding clouds of Figure 7.8, we expected the latent encoding to be similar to a suitably rescaled version of the source image. The results on a simple dataset like MNIST confirmed this hypothesis, but on more complex ones like CIFAR10 it does not seem to be the case, as exemplified in Figure 7.10. We then progressively improved the model's architecture by augmenting their depth and channel dimensions, with the latter being typically the most effective way to improve their performance. In the end, the best results were obtained with a U-Net architecture that is practically identical to the denoising network. Many additional experiments have been performed, comprising autoencoders, residual networks, inception modules, and variants

with different padding modalities or regularizations. However, they did not prove to be particularly effective and were thus dropped from our discussion.

Network	Params	MSE				
		MNIST	Fashion MNIST	CIFAR10	Oxford Flowers	CelebA
layers: 1 conv. 5×5	78	0.00704	0.0152	0.0303	0.0372	0.0189
layers: 3 conv. 5×5 channels: 16-16-out	7,233	0.00271	0.00523	0.0090	0.0194	0.0101
layers: 3 conv. 5×5 channels: 64-64-out	105,729	0.00206	0.00454	0.0061	0.0153	0.00829
layers: 2conv.5×5 3conv.3×3 channels: 128-128-128-128-out	859,009	0.00121	0.00172	0.0038	0.00882	0.00396
U-Net	9,577,683	0.000361	0.000890	0.0012	0.00248	0.00147

Table 7.1: Comparing the Mean Square Error (MSE) through the embedding-reconstruction process using different embedding networks; the MSE standard deviation is below the last reported decimal. The number of parameters refers to the instance of the network for the CelebA dataset.

In Figure 7.11, we show some examples of embeddings and relative reconstructions in the case of the CelebA dataset.

The quality of the reconstruction is definitely high, with just a slight blurriness. There are two possible justifications for the tiny inaccuracy of this result: it could either be a fault of the generator, which is unable to create the requested images (as it is frequently the case with Generative Adversarial Networks [204]), or it could be a fault of the Embedding Network, which is unable to compute the correct seed.

To better investigate the issue, we performed two experiments. First, we restricted the reconstruction to images produced by the generator: in this case, if the Embedding network works well, it should be able to reconstruct almost perfect images. Secondly, we tried to improve the seeds computed by the Embedding Network through gradient descent, looking for better candidates.

We report the result of the first experiment in Figure 7.12.

While the reconstruction is qualitatively accurate, we can also confirm the effectiveness in a more analytical way. In Table 7.2 we compare the mean squared error of the reconstruction starting from original CelebA images versus generated data: the latter is sensibly smaller.

The fact that embedding works better for generated images is, however, not conclusive: it could either be explained by a deficiency of the generator, unable to generate all images

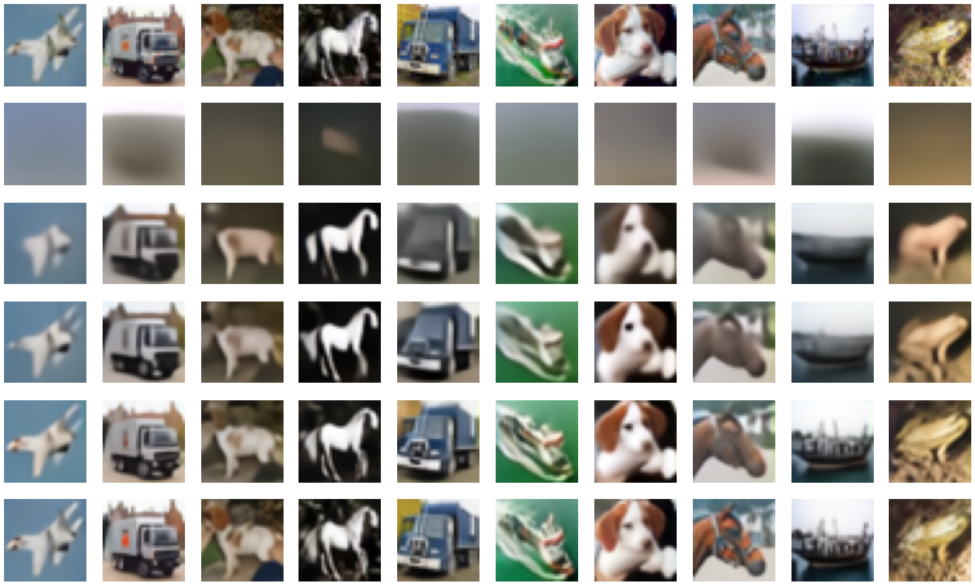


Figure 7.10: Visual comparison with different Embedding Networks. We consider a set of test images from CIFAR10 (first row) and compute the embedding with one of the Embedding Networks of Table 7.1. We then use the embeddings to generate the corresponding images (remaining rows).

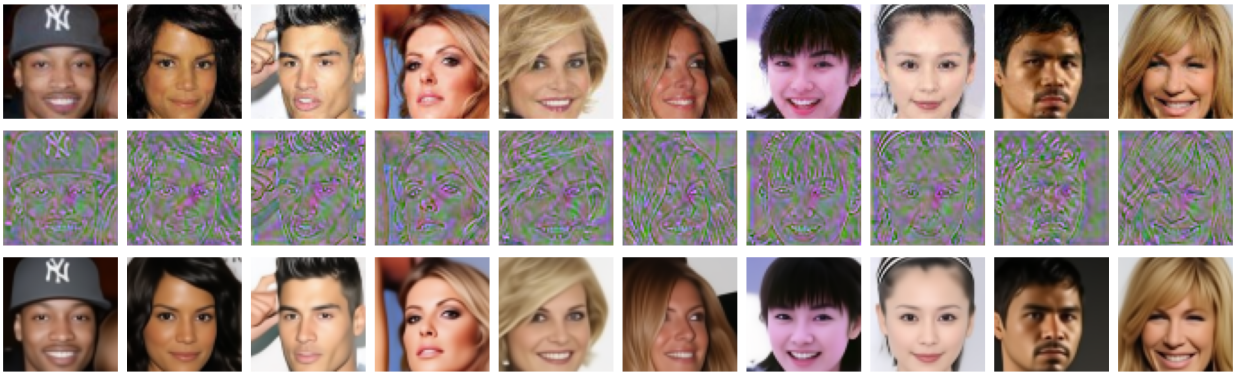


Figure 7.11: Embedding examples for the CelebA dataset. The first row contains the original examples, the second the synthesized latent seed, and the third the reconstructed image. Reconstruction is very good, with just a slight blurriness.

in the CelebA dataset, or just by the fact that generated images are "simpler" than real ones (observe the well-known patinated look, which is typical of most generative models) and hence more easily embeddable.

Figure 7.12: Embedding examples on generated images. In this case, we start from images created by the generator (first row) and re-embed them inside the latent space (second row) using the Embedding Network. In the third row, we show the reconstruction, which is almost perfect. This could be either explained by a deficiency of the generator, or just by the fact that generated images are "simpler", and hence can be more easily embedded than real ones.

Source Images	**MSE**
Dataset	0.00147
Generated	0.00074

Table 7.2: Reconstruction error. In the first case, images are taken from the CelebA dataset: in the second case, they have been generated through the reverse diffusion process. The mean squared error (MSE) was computed over 1000 examples. Both experiments achieve a small reconstruction error, although the second one is even smaller.

Even the results of the second experiment are not easily deciphered. From a visual point of view, refining the embedding through gradient descent is not producing remarkable results, as exemplified in Figure 7.13. However, numerically, we see an improvement from an MSE of 0.00147 to an MSE of 0.00058, which seems to suggest some margin of improvement for the embedding network.

In conclusion, both the generator and the embedder can likely still be improved. However, a really interesting research direction seems to be the possibility to modify the latent representation to improve the realism of the resulting image, even if possibly not in the direction of the original. Therefore, a basic embedder, even if not fully accurate, could still provide the starting point for very interesting manipulations.

7.3.3 Latent Space Interpolation

A typical application of the embedding network is for the investigation of semantical properties of the latent space, starting from real samples and their attributes. As a preliminary

Figure 7.13: Gradient descent fine-tuning. The seeds obtained through the embedding network (second row) are refined through gradient descent (fourth row). The respective resulting reconstructions are depicted in rows 3 and 5. The improvement is almost imperceptible.

step in this direction, in this Section we provide examples of latent-space interpolations: the crucial additional ability added by the embedder is in the choice of the starting and ending point, that can be the embeddings of *real* data samples: this allows us to produce smooth interpolations between any pair of images in the dataset.

In Figure 7.14 we show an example relative to the CelebA dataset. The linear interpolation in the visible space between a source and a target sample, depicted in the first row, does not produce satisfactory results: the superposition of the two images is clearly visible, introducing annoying artifacts. A better result can be achieved by first embedding both source and target into the latent space, and them moving along their (linear) interpolation (second row). The images generated from the interpolated latent points provide a smooth transition from the source to the target, as shown in the third row of Figure 7.14. In this case, the "artifacts" of the latent representations are automatically corrected by the generator, trained to produce realistic faces.

7.4 Conclusions

In this Chapter, we addressed the problem of embedding data into the latent space of Deterministic Diffusion models, providing functionality similar to the encoder in a Variational Autoencoder, or the so-called *recoder* for Generative Adversarial Networks. The main source of complexity when inverting a diffusion model is the non-injective nature of the generator:

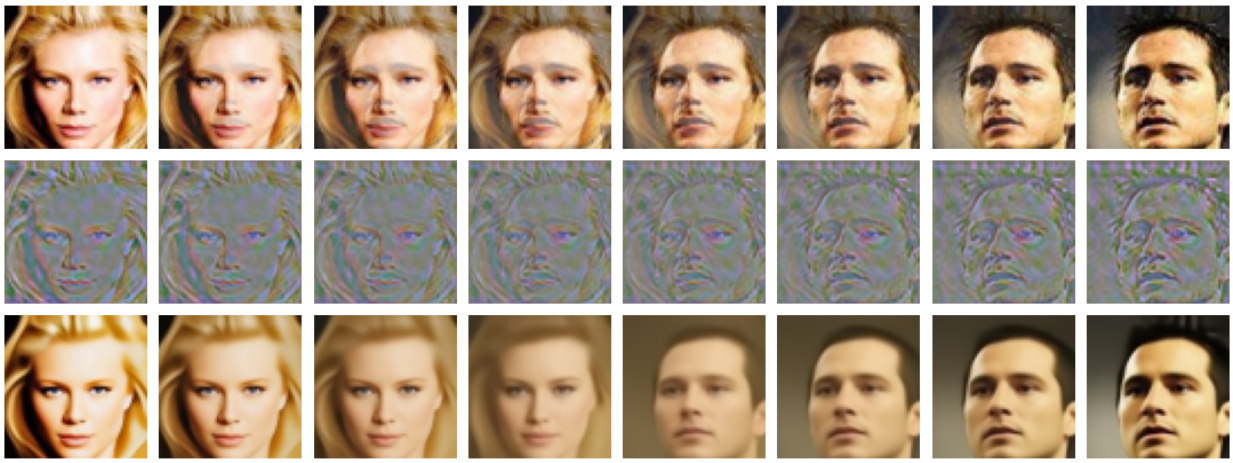


Figure 7.14: Interpolation between samples of the CelebA dataset. In the first raw, we have the linear interpolation between the source and the target. In the second raw, the linear interpolation between the latent embedding of the source, and the latent embedding of the target. In the third raw, the reconstructed images.

for each sample $\boldsymbol{x}$, there exists a cloud of elements $\boldsymbol{z}$ able to generate $\boldsymbol{x}$. We call this set the embedding of $\boldsymbol{x}$, denoted as $emb(\boldsymbol{x})$. We performed a deep investigation of the typical shape of $emb(\boldsymbol{x})$, which suggests that embeddings are usually orthogonal to the dataset. These studies point to a sort of gravitational interpretation of the reverse diffusion process, according to which the space is progressively collapsing over the data manifold. In this perspective, $emb(\boldsymbol{x})$ is just the set of all trajectories in the space ending in $\boldsymbol{x}$. We tested our interpretation on both low- and high-dimensional datasets, highlighting a quite amazing result: the latent space of a DDIM generator does not significantly depend on the specific generative model, but just on the data manifold. In other words, passing the same seed as input to different DDIMs will result in almost identical outputs. In order to compute embeddings, we considered both gradient descent approaches, as well as the definition and training of specific Embedding Networks. We showed that, among all the architectures we tested, a U-Net obtained the best results, achieving a high-quality reconstruction from both a quantitative and qualitative point of view.

Embedding networks have a lot of interesting applications, largely exemplified in the introduction. More generally, the simplicity and ease of use of Embedding Networks open a wide range of fascinating perspectives about the exploration of semantic trajectories in the latent space, the disentanglement of the different aspects of variations, and the possibility of data editing. We thus hope that our results, by expanding the current understanding of generative models, can guide future research efforts.

Conclusion and Future Works

The aim of this thesis was to present innovative frameworks to solve inverse problems related to life sciences, such as image deblurring and sparse computed tomography, by neural networks. This has been done by considering a hybrid approach, where iterative, model-based algorithms are employed to guarantee stability in the solution, both to unexpected noise in the data and to global, striking artifacts. The proposed methods are listed in the first part of the thesis. The second part is dedicated to an exploration of the properties of two Deep Generative Models, namely the Variational Autoencoders and the Diffusion Models. The second part is the beginning of a research project where we aim to introduce a new hybrid framework, joining together the stability and the consistency guaranteed by the model-based algorithms with the flexibility offered by generative models. A deeper discussion of this idea will be given in the Future Work section.

In Chapter 2, we extended the widely known mathematical properties of reconstruction algorithms in inverse problems, introduced in [3], to neural networks, where the lack of convergence to the true solution in the noiseless limit, makes it necessary to introduce the concept of *accuracy* of a reconstruction algorithm. The theoretical analysis shows the existence of a trade-off between accuracy and stability for any reconstruction method applied to ill-conditioned inverse problems. In the second part, we introduced a new framework, called StReNN, which considers a model-based iterative algorithm as a pre-processing step to a neural network, leading to a method that exhibits greater stability than the classical end-to-end neural networks, with negligible accuracy loss. Extensive tests considering various architectures and pre-processing methods on the image deblurring inverse problem showed that the proposed framework surpasses most state-of-the-art end-to-end neural networks when the input image is corrupted by unexpected noise.

In Chapter 3, we considered a hybrid scheme, similar to StReNN, in the solution of SparseCT inverse problems, where the non-injectivity of the forward operator leads to a reconstruction algorithm with infinite solutions. The resulting method, named RISING, was shown to surpass the classical LPP algorithm in both performance and stability. Moreover, the proposed scheme didn't require any ground-truth solution, making it viable for real medical applications where collecting pairs of associated full dose - low dose acquisition is

unfeasible.

In Chapter 4, we extended the RISING scheme to the non-convex optimization setup. In particular, we employed a Total p-Variation regularization scheme, obtaining promising results.

In Chapter 6, we reviewed modern architectures for Variational Autoencoders, and we compared their performance over multiple datasets. The comparison has been done by considering the ability to generate new images, similar to those of the test set, and measured by a widely used metric called Fréchet Inception Distance (FID). The sustainability of the compared methods is also taken into account, by computing the number of Floating Points Operations (FLOPs) required by each network to generate a single image, following the paradigm of GreenAI.

Finally, in Chapter 7, we considered a variant of the modern Diffusion Models, named Denoising Diffusion Implicit Models (DDIM), and we analyzed some features of the associated latent space. In particular, we observed that the latent encoding of any given image is invariant to changes in network architectures and training setup. We then proposed to consider a neural network, called embedding network, that maps any image to one of its latent encodings. We also compared the performance of the embedding network against a classical, gradient descent-based algorithm designed to find the latent encodings of an image, and showed that our proposed method is able to generate comparable encodings in just a fraction of a second, compared to the tenth of minutes required by the classical approach.

Future Works

The methods proposed in the thesis can be further extended. In the following, we discuss two major extensions: the first one, called GraphLa+, is an extension of my publication [9], while the second one makes use of the considerations made on the Generative Models (and in particular Diffusion Models) in [11], to solve inverse problems in a way that is closely related to the concept of Deep Generative Priors (DGP), discussed in [70]. We recall that an alternative extension has already been discussed in Section 3.3, where we propose to adapt the (TpV)^2 method to solve iterated reweighted non-convex optimization problems, with provable convergence.

GraphLa+ Consider the $L_2 - L_1$ variational problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{1}{2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta||_2^2 + \lambda||\mathbf{L}\boldsymbol{x}||_1, \tag{7.14}$$

where $\mathbf{L} \in \mathbb{R}^{n\times n}$ is any matrix such that $\ker(\boldsymbol{K}) \cap \ker(\mathbf{L}) = \{0\}$. In [9], we consider the case where $\mathbf{L}$ is the discretization of the graph laplacian operator on an initial guess $\tilde{\boldsymbol{x}}$ of the

ground-truth solution $\boldsymbol{x}^{gt}$. In particular, let $\tilde{\mathbf{X}}$ be the graph associated with $\tilde{\boldsymbol{x}}$, i.e. a graph with n nodes, each corresponding to a pixel of $\tilde{\boldsymbol{x}}$, such that there exists an edge connecting the i-th and the j-th node if and only if $w(i,j) > 0$, where:

$$w(i,j) = \begin{cases} e^{-\frac{1}{\sigma^2}|\tilde{\boldsymbol{x}}(i)-\tilde{\boldsymbol{x}}(j)|^2} & \text{if } i \neq j \text{ and } ||i-j||_\infty < R \\ 0 & \text{otherwise.} \end{cases} \tag{7.15}$$

Given that, the graph laplacian operator $\Delta^{\tilde{\boldsymbol{x}}} \in \mathbb{R}^{n \times n}$ is defined as:

$$\left(\Delta^{\tilde{\boldsymbol{x}}}\right)_i = \sum_{j \in \tilde{\mathbf{X}}} w(i,j)(\boldsymbol{x}(i) - \boldsymbol{x}(j)). \tag{7.16}$$

It is known [206, 207] that the quality of the reconstruction of 7.14 is dependent on how well $\tilde{\boldsymbol{x}}$ approximates $\boldsymbol{x}^{gt}$. For this reason, we propose to compute $\tilde{\boldsymbol{x}}$ by a neural network, whose ability to recover the structure of the true solution is undoubted. In [9] we show that this algorithm called *GraphLaNet*, is able to surpass the reconstruction quality of both the neural network and the classical graph laplacian methods. In future work, we aim to theoretically justify this approach, proving the convergence of this method to the true solution $\boldsymbol{x}^{gt}$ when the noise level δ approaches 0, and error bounds based on the discrepancy between $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}^{gt}$.

Deep Generative Prior with Diffusion Models In [70], the author describes a technique to solve inverse problems in the form of (1.2) with a generative model, called *Deep Generative Prior (DGP)*. In particular, let $p_\theta(\boldsymbol{x})$ be a DLVM and let $G_\theta(\boldsymbol{z})$ be the neural network mapping a sampled latent variable $\boldsymbol{z} \in \mathcal{Z}$ to a new sample $\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|\boldsymbol{z})$. We already observed that, especially when $\boldsymbol{K}$ does not satisfy (HC2), the direct solution of the unregularized least squares problem

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathbb{R}^n}{\arg\min} \frac{1}{2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta||_2^2, \tag{7.17}$$

fails in approximating the true solution $\boldsymbol{x}^{gt}$. A solution is *regularize* the problem, by constraining the solution $\boldsymbol{x}^*$ to be in the set $\mathcal{X}$ of *good* solutions, i.e. by solving:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathbb{R}^n}{\arg\min} \frac{1}{2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta||_2^2 \text{ such that } \boldsymbol{x} \in \mathcal{X}. \tag{7.18}$$

In Section 5.1 we remarked that a basic assumption in generative models is that $\mathcal{X} = supp(\mu_{gt})$, where μ_{gt} is the distribution of ground-truth data, with density $p_{gt}(\boldsymbol{x})$. After training, the DLVM $p_\theta(\boldsymbol{x})$ is approximately equal to $p_{gt}(\boldsymbol{x})$ which implies that, by definition, $\mathcal{X} \approx Rg(G_\theta)$. Equation (7.18) reads:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathbb{R}^n}{\arg\min} \frac{1}{2}||\boldsymbol{K}\boldsymbol{x} - \boldsymbol{y}^\delta||_2^2 \text{ such that } \exists \boldsymbol{z} \in \mathcal{Z}, \text{ with } G_\theta(\boldsymbol{z}) = \boldsymbol{x}, \tag{7.19}$$

from which

$$\begin{aligned} \boldsymbol{x}^* &= G(\boldsymbol{z}^*), \\ \boldsymbol{z}^* &= \arg\min_{\boldsymbol{z}\in\mathcal{Z}} \frac{1}{2}||\boldsymbol{K}G_\theta(\boldsymbol{z}) - \boldsymbol{y}^\delta||_2^2. \end{aligned} \tag{7.20}$$

Since the latent variables $\boldsymbol{z} \in \mathcal{Z}$ are distributed as $p(\boldsymbol{z}) = \mathcal{N}(0, I)$ by definition, then the optimization problem above can be reformulated as:

$$\begin{aligned} \boldsymbol{x}^* &= G(\boldsymbol{z}^*), \\ \boldsymbol{z}^* &= \arg\min_{\boldsymbol{z}\in\mathbb{R}^s} \frac{1}{2}||\boldsymbol{K}G_\theta(\boldsymbol{z}) - \boldsymbol{y}^\delta||_2^2 + \frac{\lambda}{2}||\boldsymbol{z}||_2^2. \end{aligned} \tag{7.21}$$

This particular formulation of the inverse problem is known as Deep Generative Prior (DGP) and the quality of its solution is strictly related to the ability of the generator $G_\theta(\boldsymbol{z})$ in approximating $p_{gt}(\boldsymbol{x})$.

A key limitation of the application of Diffusion Models in a DGP approach is that in their classical formulation, their generator $G_\theta(\boldsymbol{z})$ is a stochastic transformation of $\boldsymbol{z}$, since it is defined as an iterative algorithm (the reverse diffusion described in Chapter 7) where at each step a stochastic realization of noise is added to the processed input. For this reason, an optimization problem such as (7.21) is stochastic, and solving it is non-trivial in practice. The observations we did in [11] are important for two reasons: first, the considered variant of diffusion models (the DDIM), allows for a deterministic generator $G_\theta(\boldsymbol{z})$, that can be easily plugged in (7.21). Second, the observation that the latent encoding of any input image $\boldsymbol{x}$ is invariant to changes in network architectures, can be exploited to derive flexible models, where multiple variants of DDIM with different mathematical properties are considered in an alternated scheme, with the intent of optimizing the accuracy-stability trade-off.

Bibliography

[1] Liane Bernstein, Alexander Sludds, Ryan Hamerly, Vivienne Sze, Joel Emer, and Dirk Englund. Freely scalable and reconfigurable optical hardware for deep learning. *Scientific reports*, 11(1):3144, 2021.

[2] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Commun. ACM*, 63(12):54–63, 2020.

[3] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[4] Andrea Asperti, Davide Evangelista, and Elena Loli Piccolomini. A survey on variational autoencoders from a green AI perspective. *SN Computer Science*, 2(4):301, 2021.

[5] Andrea Asperti, Davide Evangelista, and Moreno Marzolla. Dissecting FLOPs along input dimensions for GreenAI cost estimations. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 86–100. Springer, 2021.

[6] Elena Morotti, Davide Evangelista, and Elena Loli Piccolomini. A green prospective for learned post-processing in sparse-view tomographic reconstruction. *Journal of Imaging*, 7(8):139, 2021.

[7] Davide Evangelista, Elena Morotti, and Elena Loli Piccolomini. RISING: A new framework for model-based few-view CT image reconstruction with deep learning. *Computerized Medical Imaging and Graphics*, 103:102156, 2023.

[8] Davide Evangelista, James Nagy, Elena Morotti, and Elena Loli Piccolomini. To be or not to be stable, that is the question: understanding neural networks for inverse problems, 2024.

[9] Davide Bianchi, Marco Donatelli, Davide Evangelista, Wenbin Li, and Elena Loli Piccolomini. Graph Laplacian and Neural Networks for Inverse Problems in Imaging: GraphLaNet. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 175–186. Springer, 2023.

[10] Davide Evangelista, Elena Morotti, Elena Loli Piccolomini, and James Nagy. Ambiguity in Solving Imaging Inverse Problems with Deep-Learning-Based Operators. *Journal of Imaging*, 9(7), 2023.

[11] Andrea Asperti, Davide Evangelista, Samuele Marro, and Fabio Merizzi. Image embedding for denoising generative models. *Artificial Intelligence Review*, pages 1–23, 2023.

[12] Elena Morotti, Davide Evangelista, and Elena Loli Piccolomini. Increasing noise robustness of deep learning-based image processing with model-based approaches. *Numerical Computations: Theory and Algorithms NUMTA 2023*, page 155, 2023.

[13] J. Hadamard. Sur les problèmes aux dérivés partielles et leur signification physique. *Princeton University Bulletin*, 13:49–52, 1902.

[14] Christian Clason. Regularization of inverse problems. *ArXiv*, abs/2001.00617, 2020.

[15] Per Christian Hansen. The discrete picard condition for discrete ill-posed problems. *BIT Numerical Mathematics*, 30(4):658–672, 1990.

[16] Per Christian Hansen. *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. SIAM, 1998.

[17] Per Christian Hansen. *Discrete inverse problems: insight and algorithms*. SIAM, 2010.

[18] Mario Bertero, Patrizia Boccacci, and Christine De Mol. *Introduction to inverse problems in imaging*. CRC press, 2021.

[19] Otmar Scherzer, Markus Grasmair, Harald Grossauer, Markus Haltmeier, and Frank Lenzen. *Variational methods in imaging*. Springer, 2009.

[20] Per Christian Hansen, James G Nagy, and Dianne P O'leary. *Deblurring images: matrices, spectra, and filtering*. SIAM, 2006.

[21] Avinash C Kak and Malcolm Slaney. *Principles of computerized tomographic imaging*. SIAM, 2001.

[22] Jennifer L Mueller and Samuli Siltanen. *Linear and nonlinear inverse problems with practical applications*. SIAM, 2012.

[23] Frank Natterer. *The mathematics of computerized tomography*. SIAM, 2001.

[24] Kyong Hwan Jin, Michael T McCann, Emmanuel Froustey, and Michael Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017.

[25] Elena Loli Piccolomini, V.L. Coli, E. Morotti, and L. Zanni. Reconstruction of 3D X-ray CT images from reduced sampling by a scaled gradient projection algorithm. *Comp. Opt. Appl.*, 71:171–191, 2018.

[26] E Loli Piccolomini and E Morotti. A fast total variation-based iterative algorithm for digital breast tomosynthesis image reconstruction. *Journal of Algorithms & Computational Technology*, 10(4):277–289, 2016.

[27] Elena Loli Piccolomini, Vanna Lisa Coli, Elena Morotti, and Luca Zanni. Reconstruction of 3d X-ray CT images from reduced sampling by a scaled gradient projection algorithm. *Computational Optimization and Applications*, 2018.

[28] Elena Loli Piccolomini and Elena Morotti. A model-based optimization framework for iterative digital breast tomosynthesis image reconstruction. *Journal of Imaging*, 7(2):36, 2021.

[29] TM Peters and RM Lewitt. Computed tomography with fan beam geometry. *Journal of Computer Assisted Tomography*, 1(4):429–436, 1977.

[30] Frédéric Noo, C Bernard, FX Litt, and P Marchot. A comparison between filtered backprojection algorithm and direct algebraic method in fan beam CT. *Signal processing*, 51(3):191–199, 1996.

[31] Jiangsheng You, Gengsheng L Zeng, and Zhengrong Liang. FBP algorithms for attenuated fan-beam projections. *Inverse Problems*, 21(3):1179, 2005.

[32] Mario Bertero, Patrizia Boccacci, and Valeria Ruggiero. Inverse imaging with Poisson data. *IOP Publish*, page 115, 2018.

[33] Andrei Nikolaevich Tikhonov, AV Goncharsky, VV Stepanov, and Anatoly G Yagola. *Numerical methods for the solution of ill-posed problems*, volume 328. Springer Science & Business Media, 1995.

[34] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[35] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[36] Simon Foucart, Holger Rauhut, Simon Foucart, and Holger Rauhut. *An invitation to compressive sensing*. Springer, 2013.

[37] Mallat Stephane. *A wavelet tour of signal processing*. Elsevier, 1999.

[38] Øyvind Ryan, P Ryan, and Peters. *Linear Algebra, Signal Processing, and Wavelets: A Unified Approach*. Springer, 2019.

[39] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[40] Vicent Caselles, Antonin Chambolle, and Matteo Novaga. Total variation in imaging. *Handbook of mathematical methods in imaging*, 2015.

[41] Ludwig Ritschl and et al. Improved total variation-based CT image reconstruction applied to clinical data. *Physics in Medicine & Biology*, 2011.

[42] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40:120–145, 2011.

[43] Silvia Bonettini, Riccardo Zanella, and Luca Zanni. A scaled gradient projection method for constrained image deblurring. *Inverse problems*, 25(1):015002, 2008.

[44] Giovanni S Alberti, Alessandro Felisi, Matteo Santacesaria, and S Ivan Trapasso. Compressed sensing for inverse problems and the sample complexity of the sparse radon transform. *arXiv preprint arXiv:2302.03577*, 2023.

[45] Henri Lanteri, Muriel Roche, and Claude Aime. Penalized maximum likelihood image restoration with positivity constraints: multiplicative algorithms. *Inverse problems*, 18(5):1397, 2002.

[46] S Bonettini, F Porta, and V Ruggiero. A variable metric inertial method for convex optimization. *SIAM J. Sci. Comput*, 31(4):A2558–A2584, 2016.

[47] Giacomo Frassoldati, Luca Zanni, and Gaetano Zanghirati. New adaptive stepsize selections in gradient methods. *Journal of industrial and management optimization*, 4(2):299–312, 2008.

[48] Federica Porta, Marco Prato, and Luca Zanni. A new steplength selection for scaled gradient methods with application to image deblurring. *Journal of Scientific Computing*, 65(3):895–919, 2015.

[49] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

[50] S. Bonettini and M. Prato. New convergence results for the scaled gradient projection method. *Inv. Probl.*, 31(9):1196–1211, 2015.

[51] Heinz H Bauschke, Patrick L Combettes, Heinz H Bauschke, and Patrick L Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2017.

[52] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[53] Hanming Zhang, Liang Li, Kai Qiao, Linyuan Wang, Bin Yan, Lei Li, and Guoen Hu. Image prediction for limited-angle tomography via deep learning with convolutional neural network. *arXiv preprint arXiv:1607.08707*, 2016.

[54] Hung Le and Ali Borji. What are the Receptive, Effective Receptive, and Projective Fields of Neurons in Convolutional Neural Networks? *arXiv e-prints*, page arXiv:1705.07049, 2017.

[55] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[56] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VII*, pages 17–33. Springer, 2022.

[57] Hu Chen, Yi Zhang, Weihua Zhang, Peixi Liao, Ke Li, Jiliu Zhou, and Ge Wang. Low-dose FBP via convolutional neural network. *Biomedical optics express*, 8(2):679–694, 2017.

[58] Jiaxi Wang, Li Zeng, Chengxiang Wang, and Yumeng Guo. ADMM-based deep reconstruction for limited-angle CT. *Physics in Medicine & Biology*, 64(11):115011, 2019.

[59] Yo Seob Han, Jaejun Yoo, and Jong Chul Ye. Deep residual learning for compressed sensing CT reconstruction via persistent homology analysis. *arXiv preprint arXiv:1611.06391*, 2016.

[60] Jorge Nocedal. Optimization methods for large-scale machine learning [j]. *Siam Review*, 60(2), 2016.

[61] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[62] H Li, Z Xu, G Taylor, C Studer, and T Goldstein. Visualizingthelosslandscapeofneuralnets. *arXiv preprint arXiv:1712.09913*, 2017.

[63] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

[64] Jinxi Xiang, Yonggui Dong, and Yunjie Yang. Fista-net: Learning a fast iterative shrinkage thresholding network for inverse problems in imaging. *IEEE Transactions on Medical Imaging*, 40(5):1329–1339, 2021.

[65] Carla Bertocchi, Emilie Chouzenoux, Marie-Caroline Corbineau, Jean-Christophe Pesquet, and Marco Prato. Deep unfolding of a proximal interior point method for image restoration. *Inverse Problems*, 36(3):034005, 2020.

[66] A Vedaldi, V Lempitsky, and D Ulyanov. Deep image prior. *International Journal of Computer Vision*, 128(7):1867–1888, 2020.

[67] Pasquale Cascarano, Andrea Sebastiani, Maria Colomba Comes, Giorgia Franchini, and Federica Porta. Combining weighted total variation and deep image prior for natural and medical image restoration via ADMM. In *2021 21st International Conference on Computational Science and Its Applications (ICCSA)*, pages 39–46. IEEE, 2021.

[68] Pasquale Cascarano, Elena Loli Piccolomini, Elena Morotti, and Andrea Sebastiani. Plug-and-play gradient-based denoisers applied to FBP image enhancement. *Applied Mathematics and Computation*, 422:126967, 2022.

[69] Ernest Ryu, Jialin Liu, Sicheng Wang, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Plug-and-play methods provably converge with properly trained denoisers. In *International Conference on Machine Learning*, pages 5546–5557. PMLR, 2019.

[70] M. A. G. Duff, N. D. F. Campbell, and M. J. Ehrhardt. Regularising inverse problems with generative machine learning models. *Journal of Mathematical Imaging and Vision*, 2023.

[71] Ajil Jalal, Marius Arvinte, Giannis Daras, Eric Price, Alexandros G Dimakis, and Jon Tamir. Robust compressed sensing MRI with deep generative priors. *Advances in Neural Information Processing Systems*, 34:14938–14954, 2021.

[72] Nina M. Gottschling, Vegard Antun, Anders C. Hansen, and Ben Adcock. The troublesome kernel – on hallucinations, no free lunches and the accuracy-stability trade-off in inverse problems, 2023.

[73] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 07 2017.

[74] Baiyu Chen, Shuai Leng, Lifeng Yu, David Holmes III, Joel Fletcher, and Cynthia McCollough. An open library of CT patient projection data. In *Medical Imaging 2016: Physics of Medical Imaging*, volume 9783, pages 330–335. SPIE, 2016.

[75] C McCollough. Tu-fg-207a-04: Overview of the Low Dose CT Grand Challenge. *Medical physics*, 43(6Part35):3759–3760, 2016.

[76] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[77] Jaweria Amjad, Jure Sokolić, and Miguel RD Rodrigues. On deep learning for inverse problems. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1895–1899. IEEE, 2018.

[78] Chang Min Hyun, Seong Hyeon Baek, Mingyu Lee, Sung Min Lee, and Jin Keun Seo. Deep learning-based solvability of underdetermined inverse problems in medical imaging. *Medical Image Analysis*, 69:101967, 2021.

[79] Vegard Antun, Francesco Renna, Clarice Poon, Ben Adcock, and Anders C Hansen. On instabilities of deep learning in image reconstruction and the potential costs of AI. *Proceedings of the National Academy of Sciences*, 117(48):30088–30095, 2020.

[80] Yixing Huang, Tobias Würfl, Katharina Breininger, Ling Liu, Günter Lauritsch, and Andreas Maier. Some investigations on robustness of deep learning in limited angle tomography. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I*, pages 145–153. Springer, 2018.

[81] Patricia M Johnson, Geunu Jeong, Kerstin Hammernik, Jo Schlemper, Chen Qin, Jinming Duan, Daniel Rueckert, Jingu Lee, Nicola Pezzotti, Elwin De Weerdt, et al. Evaluation of the robustness of learned MRI image reconstruction to systematic deviations between training and test data for the models from the fastmri challenge. In *Machine Learning for Medical Image Reconstruction: 4th International Workshop, MLMIR 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, October 1, 2021, Proceedings 4*, pages 25–34. Springer, 2021.

[82] Jan Nikolas Morshuis, Sergios Gatidis, Matthias Hein, and Christian F Baumgartner. Adversarial robustness of MRI image reconstruction under realistic perturbations. In *Machine Learning for Medical Image Reconstruction: 5th International Workshop, MLMIR 2022, Held in Conjunction with MICCAI 2022, Singapore, September 22, 2022, Proceedings*, pages 24–33. Springer, 2022.

[83] Matthew J Muckley, Bruno Riemenschneider, Alireza Radmanesh, Sunwoo Kim, Geunu Jeong, Jingyu Ko, Yohan Jun, Hyungseob Shin, Dosik Hwang, Mahmoud Mostapha, et al. Results of the 2020 fastmri challenge for machine learning MRI image reconstruction. *IEEE transactions on medical imaging*, 40(9):2306–2317, 2021.

[84] Chi Zhang, Jinghan Jia, Burhaneddin Yaman, Steen Moeller, Sijia Liu, Mingyi Hong, and Mehmet Akçakaya. Instabilities in conventional multi-coil MRI reconstruction with small adversarial perturbations. In *2021 55th Asilomar Conference on Signals, Systems, and Computers*, pages 895–899. IEEE, 2021.

[85] Matthew J Colbrook, Vegard Antun, and Anders C Hansen. Can stable and accurate neural networks be computed?–on the barriers of deep learning and Smale's 18th problem. *arXiv preprint arXiv:2101.08286*, 2021.

[86] Harshit Gupta, Kyong Hwan Jin, Ha Q Nguyen, Michael T McCann, and Michael Unser. CNN-based projected gradient descent for consistent CT image reconstruction. *IEEE transactions on medical imaging*, 37(6):1440–1453, 2018.

[87] Yixing Huang, Alexander Preuhs, Günter Lauritsch, Michael Manhart, Xiaolin Huang, and Andreas Maier. Data consistent artifact reduction for limited angle tomography with deep learning prior. In *International workshop on machine learning for medical image reconstruction*, pages 101–112. Springer, 2019.

[88] Zhengxia Zou, Tianyang Shi, Zhenwei Shi, and Jieping Ye. Adversarial training for solving inverse problems in image processing. *IEEE Transactions on Image Processing*, 30:2513–2525, 2021.

[89] Daniel Obmann, Linh Nguyen, Johannes Schwab, and Markus Haltmeier. Augmented NETT regularization of inverse problems. *Journal of Physics Communications*, 5(10):105002, 2021.

[90] Zalan Fabian, Reinhard Heckel, and Mahdi Soltanolkotabi. Data augmentation for deep learning based accelerated MRI reconstruction with limited data. In *International Conference on Machine Learning*, pages 3057–3067. PMLR, 2021.

[91] Chris M Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[92] Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019.

[93] Rafael C Gonzalez. *Digital image processing*. Pearson education india, 2009.

[94] C. Graff and E. Sidky. Compressive sensing in medical imaging. *Appl. Opt.*, 54(8):C23–C44, 2015.

[95] Ge Wang, Jong Chu Ye, Klaus Mueller, and Jeffrey A Fessler. Image reconstruction is a new frontier of machine learning. *IEEE transactions on medical imaging*, 37(6):1289–1296, 2018.

[96] Yoseob Han and Jong Chul Ye. Framing U-Net via deep convolutional framelets: Application to sparse-view CT. *IEEE transactions on medical imaging*, 37(6):1418–1429, 2018.

[97] R Cavicchioli, J Hu, E Loli Piccolomini, E Morotti, and L Zanni. A first-order primal-dual algorithm for convex problems with applications to imaging. GPU acceleration of a model-based iterative method for digital breast tomosynthesis. *Scientific Reports*, 10(1):120–145, 2020.

[98] Emil Y Sidky, Iris Lorente, Jovan G Brankov, and Xiaochuan Pan. Do CNNs solve the CT inverse problem? *IEEE Transactions on Biomedical Engineering*, 68(6):1799–1810, 2020.

[99] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.

[100] Qiaofeng Xu, Deshan Yang, Jun Tan, Alex Sawatzky, and Mark A Anastasio. Accelerated fast iterative shrinkage thresholding algorithms for sparsity-regularized cone-beam CT image reconstruction. *Medical physics*, 43(4):1849–1872, 2016.

[101] Emil Y Sidky, Rick Chartrand, John M Boone, and Xiaochuan Pan. Constrained TpV minimization for enhanced exploitation of gradient sparsity: Application to CT image reconstruction. *IEEE journal of translational engineering in health and medicine*, 2:1–18, 2014.

[102] Zenith Purisha, Juho Rimpeläinen, Tatiana Bubba, and Samuli Siltanen. Controlled wavelet domain sparsity for x-ray tomography. *Measurement Science and Technology*, 29(1):014002, 2017.

[103] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259 – 268, 1992.

[104] Wim van Aarle, Willem Jan Palenstijn, Jan De Beenhouwer, Thomas Altantzis, Sara Bals, K. Joost Batenburg, and Jan Sijbers. The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography. *Ultramicroscopy*, 157:35–47, 2015.

[105] Wim van Aarle, Willem Jan Palenstijn, Jeroen Cant, Eline Janssens, Folkert Bleichrodt, Andrei Dabravolski, Jan De Beenhouwer, K. Joost Batenburg, and Jan Sijbers. Fast and flexible x-ray tomography using the astra toolbox. *Opt. Express*, 24(22):25129–25147, Oct 2016.

[106] Curtis R Vogel and Mary E Oman. Iterative methods for total variation denoising. *SIAM Journal on Scientific Computing*, 1996.

[107] Tony Chan et al. Recent developments in total variation image restoration. *Mathematical Models of Computer Vision*, 2005.

[108] Zhen Tian and et al. Low-dose CT reconstruction via edge-preserving total variation regularization. *Physics in Medicine & Biology*, 2011.

[109] Mila Nikolova and et al. Fast nonconvex nonsmooth minimization methods for image restoration and reconstruction. *IEEE Trans on Image Processing*, 2010.

[110] Buxin Chen and et al. Non-convex primal-dual algorithm for image reconstruction in spectral CT. *Computerized Medical Imaging and Graphics*, 2021.

[111] Emil Y. Sidky and et al. Constrained TpV minimization for enhanced exploitation of gradient sparsity: Application to CT image reconstruction. *IEEE Journal of Translational Engineering in Health and Medicine*, 2014.

[112] Ailong Cai and et al. Efficient TpV minimization for circular, cone-beam computed tomography reconstruction via non-convex optimization. *Computerized Medical Imaging and Graphics*, 45:1–10, 2015.

[113] Hanming Zhang, Linyuan Wang, Bin Yan, Lei Li, Ailong Cai, and Guoen Hu. Constrained total generalized p-variation minimization for few-view x-ray computed tomography image reconstruction. *PLoS One*, 11(2):e0149899, 2016.

[114] Rick Chartrand. Exact reconstruction of sparse signals via nonconvex minimization. *IEEE Signal Processing Letters*, 2007.

[115] Ulugbek S Kamilov and et al. A plug-and-play priors approach for solving nonlinear imaging inverse problems. *IEEE Signal Processing Letters*, 2017.

[116] Siqi Ye, Zhipeng Li, Michael T McCann, Yong Long, and Saiprasad Ravishankar. Unified supervised-unsupervised (super) learning for x-ray CT image reconstruction. *IEEE Transactions on Medical Imaging*, 40(11):2986–3001, 2021.

[117] Yunmei Chen and other. Learnable descent algorithm for nonsmooth nonconvex image reconstruction. *SIAM Journal on Imaging Sciences*, 2021.

[118] Emil Y Sidky, Jakob H Jørgensen, and Xiaochuan Pan. Convex optimization problem prototyping for image reconstruction in computed tomography with the Chambolle–Pock algorithm. *Physics in Medicine & Biology*, 2012.

[119] Emmanuel J Candès, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted L1 minimization. *Journal of Fourier analysis and applications*, 2008.

[120] D Lazzaro, E Loli Piccolomini, and F Zama. A nonconvex penalization algorithm with automatic choice of the regularization parameter in sparse imaging. *Inverse Problems*, 35(8):084002, 2019.

[121] Achraf Oussidi and Azeddine Elhassouny. Deep generative models: Survey. In *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–8, 2018.

[122] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[123] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[124] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org, 2015.

[125] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Found. Trends Mach. Learn.*, 12(4):307–392, 2019.

[126] Ruoqi Wei, Cesar Garcia, Ahmed ElSayed, Viyaleta Peterson, and Ausif Mahmood. Variations in variational autoencoders - a comparative evaluation. *IEEE Access*, PP:1–1, 08 2020.

[127] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[128] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[129] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[130] Abdul Jabbar, Xi Li, and Bourahla Omar. A survey on generative adversarial networks: Variants, applications, and training, 2020.

[131] Carl Doersch. Tutorial on variational autoencoders. *CoRR*, abs/1606.05908, 2016.

[132] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[133] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

[134] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

[135] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (CelebA) dataset. *Retrieved August*, 15(2018):11, 2018.

[136] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6629–6640, 2017.

[137] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society, 2016.

[138] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[139] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[140] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019,*

Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics, 2019.

[141] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[142] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org, 2014.

[143] R. Wei and A. Mahmood. Recent advances in variational autoencoders with representation learning for biomedical informatics: A survey. *IEEE Access*, 9:4939–4956, 2021.

[144] Ashley Spindler, James E. Geach, and Michael J. Smith. Astrovader: Astronomical variational deep embedder for unsupervised morphological classification of galaxies and synthetic image generation, 2020.

[145] Bin Dai and David P. Wipf. Diagnosing and enhancing VAE models. In *Seventh International Conference on Learning Representations (ICLR 2019), May 6-9, New Orleans*, 2019.

[146] Abhishek Kumar, Ben Poole, and Kevin Murphy. Regularized autoencoders via relaxed injective probability flow. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 4292–4301. PMLR, 2020.

[147] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. *CoRR*, abs/2007.03898, 2020.

[148] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2016.

[149] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-VAE. *arXiv preprint arXiv:1804.03599*, 2018.

[150] Alexander A. Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a broken ELBO. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 159–168, 2018.

[151] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. *CoRR*, abs/1511.06349, 2015.

[152] Diederik P. Kingma, Tim Salimans, Rafal Józefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational autoencoders with inverse autoregressive flow. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4736–4744, 2016.

[153] Serena Yeung, Anitha Kannan, Yann Dauphin, and Li Fei-Fei. Tackling over-pruning in variational autoencoders. *CoRR*, abs/1706.03643, 2017.

[154] Andrea Asperti and Matteo Trentin. Balancing reconstruction error and Kullback-Leibler divergence in variational autoencoders. *IEEE Access*, 8:199440–199448, 2020.

[155] Huajie Shao, Zhisheng Xiao, Shuochao Yao, Aston Zhang, Shengzhong Liu, and Tarek Abdelzaher. ControlVAE: Tuning, analytical properties, and performance analysis, 2020.

[156] Andrea Asperti. Sparsity in variational autoencoders. In *Proceedings of the First International Conference on Advances in Signal Processing and Artificial Intelligence, ASPAI, Barcelona, Spain, 20-22 March 2019*, 2019.

[157] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2015.

[158] Brian Trippe and Richard Turner. Overpruning in variational bayesian neural networks. In *Advances in Approximate Bayesian Inference workshop at NIPS 2017*, 2018.

[159] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[160] Andrea Asperti. Variational autoencoders and the variable collapse phenomenon. *Sensors & Transducers*, 234(6):1–8, 2019.

[161] Andrea Asperti. About generative aspects of variational autoencoders. In *Machine Learning, Optimization, and Data Science - 5th International Conference, LOD 2019, Siena, Italy, September 10-13, 2019, Proceedings*, pages 71–82, 2019.

[162] Matthew D. Hoffman and Matthew J. Johnson. ELBO surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, 2016.

[163] Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. Distribution matching in variational inference, 2018.

[164] Ilya O. Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. *CoRR*, abs/1711.01558, 2017.

[165] Jakub M. Tomczak and Max Welling. VAE with a vampprior. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 1214–1223, 2018.

[166] Matthias Bauer and Andriy Mnih. Resampled priors for variational autoencoders. *CoRR*, abs/1810.11428, 2018.

[167] Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. In Amir Globerson and Ricardo Silva, editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 856–865. AUAI Press, 2018.

[168] N. I. Fisher, T. Lewis, and B. J. J. Embleton. *Statistical Analysis of Spherical Data*. Cambridge University Press, 1987.

[169] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315, 2017.

[170] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 478–487. JMLR.org, 2016.

[171] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *CoRR*, abs/1611.02648, 2016.

[172] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael J. Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[173] Andrea Asperti. Variance loss in variational autoencoders. In *Machine Learning, Optimization, and Data Science - 6th International Conference, LOD 2020, Siena, Italy, September 10-13, 2020, July 19-23, 2020, Proceedings*, volume To appear of *Lecture Notes in Computer Science*. Springer, 2020.

[174] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 658–666, 2016.

[175] Xianxu Hou, LinLin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. *CoRR*, abs/1610.00291, 2016.

[176] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1558–1566. JMLR.org, 2016.

[177] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. F-VAEGAN-D2: A feature generating framework for any-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10275–10284. Computer Vision Foundation / IEEE, 2019.

[178] Rui Gao, Xingsong Hou, Jie Qin, Jiaxin Chen, Li Liu, Fan Zhu, Zhao Zhang, and Ling Shao. Zero-VAE-GAN: Generating unseen features for generalized and transductive zero-shot learning. *IEEE Trans. Image Process.*, 29:3665–3680, 2020.

[179] Wenxiao Chen, Wenda Liu, Zhenting Cai, Haowen Xu, and Dan Pei. VAEPP: variational autoencoder with a pull-back prior. In Haiqin Yang, Kitsuchart Pasupa, Andrew Chi-Sing Leung, James T. Kwok, Jonathan H. Chan, and Irwin King, editors, *Neural Information Processing - 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 23-27, 2020, Proceedings, Part III*, volume 12534 of *Lecture Notes in Computer Science*, pages 366–379. Springer, 2020.

[180] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1462–1471. JMLR.org, 2015.

[181] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[182] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2980–2988, 2015.

[183] Emile Mathieu, Tom Rainforth, N. Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4402–4412. PMLR, 2019.

[184] Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, N. Siddharth, Brooks Paige, Dana H. Brooks, Jennifer G. Dy, and Jan-Willem van de Meent. Structured disentangled representations. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 2525–2534. PMLR, 2019.

[185] Priyank Jaini, Ivan Kobyzev, Yaoliang Yu, and Marcus Brubaker. Tails of Lipschitz Triangular Flows. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4673–4681. PMLR, 2020.

[186] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning, 2019.

[187] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications, 2017.

[188] Yunho Jeon and Junmo Kim. Constructing fast network through deconstruction of convolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman,

Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5955–5965, 2018.

[189] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

[190] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(8):2011–2023, 2020.

[191] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[192] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3942–3951. AAAI Press, 2018.

[193] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2414–2423. IEEE Computer Society, 2016.

[194] Danilo Branca. Generazione di attributi facciali mediante feature-wise linear modulation. Master's thesis, University of Bologna, 2020.

[195] Alfredo Canziani, Eugenio Culurciello, and Adam Paszke. Evaluation of neural network architectures for embedded systems. In *IEEE International Symposium on Circuits and Systems, ISCAS 2017, Baltimore, MD, USA, May 28-31, 2017*, pages 1–4. IEEE, 2017.

[196] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8780–8794, 2021.

[197] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad

Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *CoRR*, abs/2205.11487, 2022.

[198] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022.

[199] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. GAN inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[200] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[201] Valentin Khrulkov and Ivan Oseledets. Understanding DDPM latent codes through optimal transport. *arXiv preprint arXiv:2202.07477*, 2022.

[202] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

[203] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[204] Andrea Asperti and Valerio Tonelli. Comparing the latent space of generative models. *Neural Computing & Applications*, 35:3155—-3172, 2023.

[205] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[206] Alessandro Buccini and Marco Donatelli. Graph Laplacian in $\ell^2 - \ell^q$ regularization for image reconstruction. In *2021 21st International Conference on Computational Science and Its Applications (ICCSA)*, pages 29–38. IEEE, 2021.

[207] Davide Bianchi, Alessandro Buccini, Marco Donatelli, and Emma Randazzo. Graph Laplacian for image deblurring. *ETNA*, 55:169–186, 2022.