Invertible Neural Networks and Normalizing Flows for Image Reconstruction

Dissertation

zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften
(Dr. rer. nat)

vorgelegt von

Alexander Konstantin Denker

Bremen, 12.03.2024

Promotionskomitee:
1. Gutachter: Prof. Dr. Dr. h.c. Peter Maass
2. Gutachterin: Dr. Lisa Maria Kreusser

Abstract

Data-based methods, in particular deep learning methods, have been successfully applied to solve various inverse problems. In medical imaging, for example in computed tomography (CT) or magnetic resonance imaging (MRI), deep learning methods have been developed that can lead to a remarkable increase in image quality. During my doctoral studies, I analyzed and developed various deep learning methods for image reconstruction with a focus on the use of invertible neural networks. Invertible neural networks form the basis for the implementation of generative normalizing flows. Furthermore, I participated in data challenges to apply and validate these and other novel deep learning methods to real-world problems.

This work consists of six publications. In the first works, the application of invertible networks to approximate both prior and posterior distributions in the context of medical imaging problems is investigated. In the Helsinki Tomography Challenge, we evaluated different approaches for CT image reconstruction on a real data set and achieved second place. Also, we investigated the application of score-based diffusion models, as an extension of the generative models in the other publications, to CT and MR image reconstruction. In particular, we developed methods to deal with discrepancies between the training and test distributions. In addition, we have made necessary adaptations for the application of these methods to positron emission tomography image reconstruction.

The necessary background is presented in the first part of the thesis, where the application and construction of invertible neural networks are discussed. Furthermore, the connection between normalizing flows and score-based diffusion models is drawn.

Zusammenfassung

Datenbasierten Methoden, insbesondere Deep Learning Methoden, wurden in den letzten Jahren erfolgreich zur Lösung von verschiedenen Inversen Problemen angewendet. In der medizinischen Bildgebung, zum Beispiel in der Computertmographie (CT) oder Magnetresonanztomographie (MR), wurden Methoden entwickelt, die zu einer bemerkenswerten Steigerung der Bildqualität führen können. In meiner Promotion habe ich verschiedene Deep Learning Methoden zur Bildrekonstruktion analysiert und entwickelt. Mein Fokus lag in der Analyse von invertierbaren neuronalen Netzen. Invertierbare neuronale Netze bilden die Grundlage für die Implementation von generativen Normalizing Flows. Des weiteren habe ich an verschiedenen Challenges teilgenommen, um diese neuartigen Deep Learning Methoden auf reale Problemstellungen anzuwenden und zu validieren.

Diese Arbeit besteht aus sechs Veröffentlichungen. In den ersten beiden Arbeiten wird die Anwendung von invertierbaren Netzwerken zur Approximation von a-posteriori Verteilungen im Kontext von verschiedenen medizinischen Bildgebungsproblemen untersucht. Zusätzlich haben wir an der Implementierung von invertierbaren Netzwerken zur Approximation der a-priori Verteilung gearbeitet. In der Helsinki Tomographie Challenge haben wir verschiedenen Ansätze für die CT-Rekonstruktion auf einem realen Datensatz ausgewertet und den zweiten Platz erreichen können. In dem letzten Abschnitt haben wir die Anwendung von Score-basierten Diffusionsmodellen, als Erweiterung der generativen Modelle in den anderen Veröffentlichungen, auf CT-Rekonstruktion und MR-Rekonstruktion untersucht. Insbesondere wurden Methoden für den Fall einer Diskrepanz der Training- und Testverteilung entwickelt. Außerdem haben wir notwendige Anpassungen für die Anwendung dieser Methoden auf Positronen-Emissions-Tomographie vorgenommen.

Zusätzlich wird im ersten Teil der Arbeit der notwendige Hintergrund präsentiert. Hier wird insbesondere auf die Anwendung und Konstruktion von invertierbaren neuronalen Netzwerken eingegangen. Weiterhin wird die Verbindung von Normalizing Flows und Score-basierten Diffusionsmodellen untersucht.

Acknowledgements

I would like to thank Peter Maass very much for his support and guidance throughout my doctorate. I had the privilege of being part of the Research Training Group π^3 and the AG Technomathematik. I am very grateful to the members of the working group, in particular Clemens Arndt, Sören Dittmer, Judith Nickel, Maximilian Schmidt, and Johannes Leuschner, for many exciting and interesting discussions. I want to thank Tom Freudenberg for all the help with FEniCS and Tobias Kluth for discussions about EIT. Furthermore, I wanted to thank the two administrators, Dörte Mindermann and Judith Barthel, for always helping me with all organizational questions.

I started my PhD during the COVID-19 pandemic. I would like to thank Jens Behrmann, Johannes Leuschner, and Maximilian Schmidt in particular for giving me a smooth and positive start to my time as a PhD student.

In 2022 I had the opportunity for a research visit to the University College London. I want to thank Simon Arridge for welcoming me so kindly to his group. I had a great time in London and wanted to thank Imraj Singh, Riccardo Barbano, and Željko Kereta for the great collaboration. I very much look forward to continuing this collaboration in the future.

Finally, I need to thank my friends and family for all the support during the last few years. Throughout our Bachelor's, Master's, and doctoral studies, Marek Wiesner, David Erzmann, and I mutually encouraged and supported each other. Thanks to Mattea Schorr for accompanying me all the way.

Contents

List of Figures

List of Tables

Introduction

Motivation

The task in inverse problems is to reconstruct unknown parameters based on indirect, noisy measurements. Inverse problems arise in a wide variety of scientific and technical applications. In particular, medical imaging relies on a stable solution of these inverse problems, for example in computed tomography or magnetic resonance imaging. The integration of data-based methods into the reconstruction process has led to significant advances in recent years. In addition to a single point estimate for the solution, we are interested in quantifying the uncertainty of this estimate. Deep generative models are a promising tool for this task. Here, my research focuses on two methods based on the theory of optimal transport: normalizing flows and score-based generative models. First, we worked on the construction of normalizing flows with invertible neural networks. Enforcing invertiblity, while still retaining flexibility, is a hard problem. During my doctoral studies, I explored various methods to construct flexible invertible neural networks.

Generative models, like normalizing flows, can be used to approximate the prior or posterior distribution based on available data. Recently, score-based diffusion models, have received a large amount of attention, due to their outstanding performance in many imaging domains. In contrast to normalizing flows, the neural network architectures used to implement diffusion models are not required to be invertible. This flexibility allows for a more simple design of diffusion models and modern network architectures can be integrated more easily. Still, specifically in the application to inverse problems, various challenges remain. In this thesis, we tackle two of these: distribution shifts and necessary adaptations for application to positron emission tomography.

Contribution

This cumulative thesis is based on the following papers (organised chronologically):

Conditional Normalizing Flows for Low-Dose Computed Tomography Image Reconstruction
Alexander Denker, Maximilian Schmidt, Johannes Leuschner, Peter Maass, Jens Behrmann
Second workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (ICML), (2020).
I had the idea for the project, did most of the implementation and did the majority of the writing of the paper.

Conditional Invertible Neural Networks for Medical Imaging
Alexander Denker, Maximilian Schmidt, Johannes Leuschner, Peter Maass
Journal of Imaging, 7(11):243, (2021). DOI: 10.3390/jimaging7110243
The idea was formed in discussions with Maximilian Schmidt and Johannes Leuschner. I am responsible for the magnetic resonance experiments and most of the computed tomography experiments.

PatchNR: Learning From Very Few Images by Patch Normalizing Flow Regularization
Fabian Altekrüger, Alexander Denker, Paul Hagemann, Johannes Hertrich, Peter Maass, Gabriele Steidl
Inverse Problems, 39(6):064006, (2023). DOI: 10.1088/1361-6420/acce5e
The idea was formed during my visit to the group of Professor Steidl. I am responsible for parts of the analysis and implementation regarding the computed tomography experiments and baseline methods. I contributed to the writing of the paper.

Model-based Deep Learning Approaches to the Helsinki Tomography Challenge 2022
Clemens Arndt, Alexander Denker, Sören Dittmer, Johannes Leuschner, Judith Nickel, Maximilian Schmidt[1]
Applied Mathematics for Modern Challenges, (2023).
DOI: 10.3934/ammc.2023007
It was my idea to take part in the challenge. I am responsible for the implementation and estimation of the forward operator, most of the creation of the synthetic dataset, and the implementation and training of the modified learned primal-dual method, which scored second place in the challenge.

Steerable Conditional Diffusion for Out-of-Distribution Adaptation in Imaging Inverse Problems
Riccardo Barbano, Alexander Denker, Hyungjin Chung, Tae Hoon Roh, Simon Arrdige, Peter Maass, Bangti Jin, Jong Chul Ye
submitted to *AAAI*, under review.
The idea was formed with Riccardo Barbano during my research stay at the group of Professor Arridge. Riccardo Barbano and I did the initial implementation and are equally responsible for the computed tomography experiments. Riccardo Barbano, Hyungjin Chung and I have contributed equally to the work.

Score-Based Generative Models for PET Image Reconstruction
Imraj RD Singh, Alexander Denker, Riccardo Barbano, Željko Kereta, Bangti Jin, Kris Thielemans, Peter Maass, Simon Arridge
submitted to *MELBA*, under review.
The idea was formed in discussions during my research stay at the group of Professor Arridge at UCL. Riccardo Barbano and worked on the initial implementation of the score model. Imrah Singh and I are responsible for running and designing the experiments and the PET-specific modifications. Imraj Singh, Riccardo Barbano, and I have contributed equally to the work.

[1] Alphabetical author order

Furthermore, I co-authored the following papers (organized chronologically):

Quantitative Comparison of Deep Learning-based Image Reconstruction Methods for Low-dose and Sparse-angle CT Applications
Johannes Leuschner and Maximilian Schmidt and Poulami S. Ganguly, Vladyslav Andriiashen, Vladyslav, Sophia B. Coban, Alexander Denker, Dominik Bauer, Amir Hadjifaradji, Kees J. Batenburg, Peter Maass, Maureen van Eijnatten
Journal of Imaging, 7(3):44, (2021). DOI: 10.3390/jimaging7030044
I am responsible for the experiments regarding the conditional invertible neural network and the learned iterative methods. Further, I helped write the paper.

The Deep Capsule Prior–Advantages Through Complexity?
Maximilian Schmidt, Alexander Denker, Johannes Leuschner
PAMM, 21(1), (2021). DOI: 10.1002/pamm.202100166
I am responsible for the deep image prior baseline experiments and some of the deep capsule prior experiments. Further, I helped write the paper.

In Focus-hybrid Deep Learning Approaches to the HDC2021 Challenge
Clemens Arndt, Alexander Denker, Judith Nickel, Johannes Leuschner, Maximilian Schmidt, Gael Rigaud[1]
Inverse Problems and Imaging, (2022). DOI: 10.3934/ipi.2022061
It was my idea to tackle the challenge using a synthetic training dataset. I implemented the synthetic training dataset. Johannes Leuschner and myself worked on the estimation of the forward operator. I implemented the modified learned gradient descent method and the training routine.

An Educated Warm Start For Deep Image Prior-Based Micro CT Reconstruction
Riccardo Barbano, Johannes Leuschner, Maximilian Schmidt, Alexander Denker, Andreas Hauptmann, Peter Maass, Bangti Jin
IEEE Transactions on Computational Imaging, (2022).
DOI: 10.1109/TCI.2022.3233188
I was responsible for deep image prior baseline experiments and helped with the writing of the paper.

Invertible Residual Networks in the Context of Regularization Theory for Linear Inverse Problems
Clemens Arndt, Alexander Denker, Sören Dittmer, Nick Heilenkötter, Meira Iske, Tobias Kluth, Peter Maass, Judith Nickel[1]
Inverse Problems, (2023). DOI: 10.1088/1361-6420/ad0660
I am responsible for parts of the analysis concerning the specialized architectures and the initial numerical experiments.

Thesis Overview

Chapter 1 provides the necessary background for inverse problems and deep learning. Specifically, Section 1.4 introduces various concepts of applying deep learning to inverse problems. Chapter 2 focuses on invertible neural networks, with an emphasis on generative modeling and normalizing flows. Additionally, in Section 2.4, the link between normalizing flows and the recently introduced score-based diffusion models is explored.

My own work will be cited in red, for example [23], whereas other sources are cited in blue, for example [1].

Part I

Background

Chapter 1

Inverse Problems and Deep Learning

The goal in inverse problems is to recover a parameter of interested from indirect and noisy measurements. Inverse problems arise naturally in many scientific applications. For example, many medical imaging tasks, like computed tomography (CT), magnetic resonance imaging (MRT) or positron emission tomography (PET), can be formulated as inverse problems. In these different imaging modalities, we want to reconstruct an image of the interior of the human body from indirect measurements. For example in CT, the parameter of interest is the density and the measurements correspond to the attenuation of x-rays, which are shot through the body from various angles [146]. These three medical imaging modalities are examples of *linear* inverse problems, which will be the focus of this thesis. In particular, we will mostly focus on inverse problem in imaging, where the parameter of interest is an *image*. In a linear inverse problem, we want to recover the image $\mathbf{x} \in \mathcal{X}$ using noisy data $\mathbf{y}_\delta \in \mathcal{Y}$ given by

$$\mathbf{y}_\delta = \mathbf{A}\mathbf{x} + \eta, \tag{1.1}$$

where $\mathbf{A} : \mathcal{X} \to \mathcal{Y}$ is a *linear* forward operator and $\eta \in \mathcal{Y}$ denotes measurement noise. In Eqn. (1.1), we highlight the case of additive noise. In many important applications, the inverse problem (1.1) will be ill-posed, in the sense that no unique solution exists, the reconstruction is highly susceptible to noise, or both, see for example [65, 140, 177]. To handle ill-posedness, the inverse problem needs to be regularized. With the advent of deep learning models, data-driven regularization methods have become increasingly popular, see for example the review [16], and several novel data-driven methods will be discussed in this chapter.

In this chapter, we start with a short presentation of inverse problems, where we follow the work of Rieder [165]. We will cover both the functional analytic and the statistical approach to solving inverse problems. This is followed by an introduction to deep learning and deep neural networks, with a focus on imaging applications. In the last part of the chapter, we deal with different applications of deep learning models to inverse problems.

1.1 Inverse Problems

In this section, we discuss the theory of linear inverse problems and cover regularization techniques for a stable recovery of reconstructions. The prototypical inverse problem is defined as follows.

Definition 1.1 (Inverse problem)**.** *The measurements* $\mathbf{y} \in \mathcal{Y}$ *follow an observation model*

$$\mathbf{y} = \mathbf{A}\mathbf{x},$$

where $\mathbf{A} : \mathcal{X} \to \mathcal{Y}$ *maps from the set of all possible images* $\mathcal{X}$ *to the set of all possible measurements* $\mathcal{Y}$*. The inverse problem is defined by*

$$\textit{Given } \mathbf{y} \in \mathcal{Y}, \textit{ find } \mathbf{x} \in \mathcal{X} \textit{ such that } \mathbf{y} = \mathbf{A}\mathbf{x}.$$

In this thesis, we only consider bounded linear operators $\mathbf{A} : \mathcal{X} \to \mathcal{Y}$ between Hilbert spaces. However, there exist also a rich theory of inverse problems in Banach spaces [178].

1.1.1 Ill-posedness and Regularization Theory

The definition of well-posed problems goes back to Hadamard [81]. A problem is well-posed, if and only if three conditions hold: 1) the problem has a solution, 2) the solution is unique and 3) the solution changes continuously with the data. Adapted to inverse problems, we arrive at the following definition of well-posedness.

Definition 1.2 (Well-posedness (Hadamard))**.** *Let* $\mathbf{A} : \mathcal{X} \to \mathcal{Y}$*. The problem*

$$\textit{find } \mathbf{x} \in \mathcal{X} \textit{ s.t. } \mathbf{A}\mathbf{x} = \mathbf{y} \textit{ for } \mathbf{y} \in \mathcal{Y}$$

is called ***well-posed according to Hadamard****, if and only if*

1) $\forall \mathbf{y} \in \mathcal{Y}\ \exists \mathbf{x} \in \mathcal{X} : \mathbf{A}\mathbf{x} = \mathbf{y}$ *(solvability)*

2) $\forall \mathbf{y} \in \mathcal{Y}\ \exists! \mathbf{x} \in \mathcal{X} : \mathbf{A}\mathbf{x} = \mathbf{y}$ *(uniqueness)*

3) $\mathbf{A}^{-1} : \mathcal{Y} \to \mathcal{X}$ *is continuous with respect to the norm in* $\mathcal{X}, \mathcal{Y}$ *(stability).*

If one of these conditions is violated, the problem is ***ill-posed according to Hadamard****.*

The three conditions can be expressed in terms of the forward operator $\mathbf{A}$. Solvability requires a surjective and uniqueness an injective operator $\mathbf{A}$. To fulfill the stability condition it is necessary for the inverse $\mathbf{A}^{-1}$ to exist and to be continuous.

We can deal with non-surjective or non-injective operators by broadening the definition of a solution. If the operator $\mathbf{A}$ is not surjective, there exist $\mathbf{y} \notin \text{Range}(\mathbf{A})$ for which we cannot find a $\mathbf{x}$ that exactly maps to $\mathbf{y}$. To handle this scenario, we use a Best-approximation, defined as the image $\mathbf{x}$ such that $\mathbf{A}\mathbf{x}$ is closest to $\mathbf{y}$ with respect to the norm used in $\mathcal{Y}$.

Definition 1.3 (Best-approximation)**.** $\mathbf{x} \in \mathcal{X}$ *is a Best-approximation if and only if*

$$\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_{\mathcal{Y}} \leq \|\mathbf{A}\tilde{\mathbf{x}} - \mathbf{y}\|_{\mathcal{Y}} \quad \forall \tilde{\mathbf{x}} \in \mathcal{X}.$$

If both $\mathcal{X}$ *and* $\mathcal{Y}$ *are Hilbert spaces, we refer to* $\mathbf{x}$ *as the least-squares solution.*

If the forward operator $\mathbf{A}$ is not injective, there might be several elements mapping to the same $\mathbf{y}$ and the Best-approximation is not unique. We need an additional criterion to select a unique solution. Here, the Moore-Penrose solution, also referred to as Minimum-Norm solution, is commonly used.

Definition 1.4 (Moore-Penrose solution)**.** $\mathbf{x} \in \mathcal{X}$ *is the Moore-Penrose solution (Minimum-Norm solution) if and only if* $\mathbf{x}$ *is a Best-approximation and* $\|\mathbf{x}\|_{\mathcal{X}} \leq \|\tilde{\mathbf{x}}\|_{\mathcal{X}}$ *for all Best-approximations* $\tilde{\mathbf{x}}$*. We denote the Moore-Penrose inverse as* $\mathbf{A}^{\dagger} : \mathit{Range}(\mathbf{A}) \oplus \mathit{Range}(\mathbf{A})^{\perp} \subset \mathcal{Y} \to \mathcal{X}$*.*

The Moore-Penrose solution is defined as the Best-approximation with minimum norm. This definition of a solution handles the case of non-injective or non-surjective forward operators. However, the domain of the Moore-Penrose inverse is a subset of $\mathcal{Y}$ if the Range($\mathbf{A}$) is not closed. Thus, the Moore-Penrose inverse is only continuous if the range of $\mathbf{A}$ is closed. This range criterion is used by Nashed to characterize ill-posed inverse problems [65, 165]. For compact operators, we can characterize ill-posedness using the singular value decomposition. For ill-posed inverse problems, the forward operator has singular values σ_i which converge to 0. Further, the

decay rate of singular values can be used as a measure of ill-posedness. The singular values of $\mathbf{A}^\dagger$ are given as σ_i^{-1}, i.e., a decay of the singular values of $\mathbf{A}$ can lead to unbounded singular values for $\mathbf{A}^\dagger$, see the Picard condition (for example in [165, Theorem 2.3.7]).

For real-world data, the measurements are corrupted by noise and thus we only have access to $\mathbf{y}_\delta = \mathbf{y}+\eta$ with $\|\eta\|_\mathcal{Y} \leq \delta$ and δ is the noise level. Applying the Moore-Penrose inverse to the noisy data $\mathbf{y}_\delta$ leads to amplification of the noise by σ_i^{-1} in the direction of the respective singular vector. The noise amplification makes $\mathbf{A}^\dagger\mathbf{y}_\delta$ generally unusable in practice.

To deal with the noise amplification, the reconstruction process has to be stabilized, referred to as *regularization* [17, 140, 201]. Let $\{\mathcal{R}_\alpha\}_{\alpha>0}$ be a family of bounded linear operators that approximate the Moore-Penrose inverse for $\alpha \to 0$. The parameter α is referred to as the regularization parameter and can depend on the noisy measurements or the noise level, i.e., $\alpha = \alpha(\mathbf{y}^\delta, \delta)$. The reconstruction $\mathcal{R}_{\alpha(\mathbf{y}^\delta,\delta)}(\mathbf{y}^\delta)$ is then used in place of the Moore-Penrose solution. The resulting reconstruction error can be decomposed into two terms

$$\|\mathbf{A}^\dagger\mathbf{y} - \mathcal{R}_\alpha\mathbf{y}_\delta\|_\mathcal{X} \leq \|\mathbf{A}^\dagger\mathbf{y} - \mathcal{R}_\alpha\mathbf{y}\|_\mathcal{X} + \|\mathcal{R}_\alpha(\mathbf{y} - \mathbf{y}_\delta)\|_\mathcal{X}, \tag{1.2}$$

where the first part is referred to as the approximation error and the second part is governed by the noise level. In general, the first part converges to zero as $\alpha \to 0$ and the second part is small for $\alpha \to \infty$. This means, that we have to choose a suitable regularization parameter α to balance both types of errors. There exists a plethora of parameter choice strategies either dependent on the noise level, the noisy measurements, or both.

A particularly powerful and flexible approach to regularization is *variational regularization* [28, 177]. In variational regularization, the reconstruction is obtained by solving an optimization problem

$$\mathcal{R}_\alpha(\mathbf{y}_\delta) = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{X}} d(\mathbf{A}\mathbf{x}, \mathbf{y}_\delta) + \alpha\mathcal{S}(\mathbf{x}), \tag{1.3}$$

with a data discrepancy functional $d : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ and $\mathcal{S} : \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a regularization functional with a regularization parameter α. The data discrepancy functional measures the closeness of the reconstruction to the measured data, whereas the regularization functional promotes desired features of the solution. Variational regularization has generalized many other well-established regularization methods. For example, the famous Tikhonov regularization [201] can be formulated as a variational problem

$$\mathcal{R}_\alpha(\mathbf{y}_\delta) = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{X}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}_\delta\|_\mathcal{X}^2 + \alpha\frac{1}{2}\|\mathbf{x}\|_\mathcal{X}^2. \tag{1.4}$$

Variational regularization offers a flexible approach to modeling the reconstruction process, as both the data discrepancy term and the regularization can be modeled independently. In particular for convex regularizers $\mathcal{S} : \mathcal{X} \to \mathbb{R}_{\geq 0}$

$$\mathcal{R}_\alpha(\mathbf{y}_\delta) = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{X}} \|\mathbf{A}\mathbf{x} - \mathbf{y}_\delta\|_{\mathcal{X}}^2 + \alpha\mathcal{S}(\mathbf{x}), \tag{1.5}$$

variational regularization offers a rich theory, i.e., uniquess of a solution or convergence rates. Further, there exist extensions to Banach spaces [178] and to nonlinear forward operators [64].

1.1.2 Statistical Inverse Problems

In the statistical view on inverse problems, both the image $\mathbf{x}$ and the measurements $\mathbf{y}$ are modeled as random variables. Instead of recovering a single reconstruction, the goal is to estimate the *posterior* distribution $p^{\text{post}}(\mathbf{x}|\mathbf{y})$, i.e., the conditional distribution of images $\mathbf{x}$ given measurements $\mathbf{y}$ [68, 103, 197]. Access to the posterior allows for uncertainty quantification. Besides a single point estimate, we can provide the variance or confidence intervals of the solution. This is especially useful in medical imaging problems, where it can help clinical decision-making, for example informing the decision if a part of an image is a reconstruction artifact or corresponds to a lesion [5]. The statistical formulation further comes with desirable theoretical properties. For example, the statistical inverse problem can be well-posed with respect to distances in probability space, even if it is ill-posed in the functional analytic framework [117]. Moreover, in many examples, the posterior will concentrate around the true solution, if the noise level goes to zero [147].

The foundation for modeling statistical inverse problems is given by Bayes' theorem. For this thesis, we will stick to finite-dimensional inverse problems and assume that all random variables admit a (Lebesgue) density. We therefore denote both the density and the probability distribution with that density with $p(\cdot)$. However, there exists a rich theory for infinite-dimensional inverse problems, for example in the reviews [49] or [193]. In the finite-dimensional setting, we can give Bayes' theorem in the following form

$$p^{\text{post}}(\mathbf{x}|\mathbf{y}) = \frac{p^{\text{lkhd}}(\mathbf{y}|\mathbf{x})\pi(\mathbf{x})}{p(\mathbf{y})}. \tag{1.6}$$

The relationship between measurements $\mathbf{y}$ and images $\mathbf{x}$ is governed by the *likelihood* $p^{\text{lkhd}}(\mathbf{y}|\mathbf{x})$, which is the statistical analogue of the forward model

in Eqn. (1.1). The prior $\pi(\mathbf{x})$ describes prior information about the solution before we get any measurements. The term $p(\mathbf{y})$ is denoted as the evidence and has applications in Bayesian model selection [127], see [21] for an example of selecting the angles in CT. For all sampling and optimization tasks in this thesis, the evidence does not need to be evaluated. Therefore we write

$$p^{\text{post}}(\mathbf{x}|\mathbf{y}) \propto p^{\text{lkhd}}(\mathbf{y}|\mathbf{x})\pi(\mathbf{x}). \tag{1.7}$$

Similar to variational regularization, Bayes' theorem offers a flexible approach to modeling by choosing likelihood and prior independently. One of the most common likelihood models is given by the additive noise model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \eta, \tag{1.8}$$

with a deterministic linear forward operator $\mathbf{A}$ and the noise $\eta \sim p_\eta(\eta)$ follows some pre-specified distribution [16]. For this modeling approach, the likelihood reduces to

$$p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}) = p_\eta(\mathbf{y} - \mathbf{A}\mathbf{x}). \tag{1.9}$$

With a Gaussian noise model $p_\eta \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, the negative log-likelihood

$$-\log p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}) = \frac{1}{2\sigma^2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \text{constant}, \tag{1.10}$$

can be interpreted as a data-discrepancy term, similar to the squared error. Statistical estimation techniques, which only rely on the likelihood, suffer from the ill-posedness of the inverse problem. In the setting of i.i.d. Gaussian noise, the maximum likelihood estimator $\mathbf{x}_{\text{ML}}$ is given by

$$\mathbf{x}_{\text{ML}} := \operatorname*{argmax}_{\mathbf{x}\in\mathbb{R}^n} \log p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}) = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2, \tag{1.11}$$

and admits the same solution and stability issues as unregularized least squares. Thus, estimators based on the full posterior are generally preferred. In high-dimensional imaging problems, it is not possible to visualize the full posterior and we have to rely on point or interval estimates [103]. Two of the most common estimators for Bayesian inference are the maximum a posteriori (MAP) estimator

$$\mathbf{x}_{\text{MAP}} := \operatorname*{argmax}_{\mathbf{x}\in\mathbb{R}^n} \log p^{\text{post}}(\mathbf{x}|\mathbf{y}), \tag{1.12}$$

and the conditional mean (CM)

$$\mathbf{x}_{\text{CM}} := \mathbb{E}_{\mathbf{x}\sim p^{\text{post}}(\mathbf{x}|\mathbf{y})}[\mathbf{x}] = \int_{\mathbb{R}^n} \mathbf{x}\; p^{\text{post}}(\mathbf{x}|\mathbf{y}) d\mathbf{x}. \tag{1.13}$$

Taking a closer look at the MAP estimator, we see a strong link to variational

regularization. Bayes' theorem allows us to rewrite the MAP estimator as

$$\mathbf{x}_{\text{MAP}} = \underset{\mathbf{x}\in\mathbb{R}^n}{\text{argmin}} - \log p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}) - \log \pi(\mathbf{x}). \tag{1.14}$$

In the setting of additive Gaussian noise and a Gibbs prior

$$\pi(\mathbf{x}) \propto \exp\bigl(-\lambda\|\mathbf{L}\mathbf{x}\|_2^2\bigr), \tag{1.15}$$

where $\mathbf{L}^T\mathbf{L}$ is a positive definite matrix, the MAP reduces to

$$\mathbf{x}_{\text{MAP}} = \underset{\mathbf{x}\in\mathbb{R}^n}{\text{argmin}} \frac{1}{2\sigma^2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2^2 + \lambda\|\mathbf{L}\mathbf{x}\|_2^2, \tag{1.16}$$

which is equivalent to generalized Tikhonov regularization [177], see also Eqn. (1.4). This highlights the fact that the prior can take the role of a regularizer. Besides the CM, other moments of the posterior are also of interest. For example, the point-wise standard deviation is often used for uncertainty quantification in imaging problems. Estimating moments requires computations of expectations of the form

$$\mathbb{E}_{\mathbf{x}\sim p^{\text{post}}(\mathbf{x}|\mathbf{y})}[g(\mathbf{x})] = \int_{\mathbb{R}^n} g(\mathbf{x}) p^{\text{post}}(\mathbf{x}|\mathbf{y}) d\mathbf{x}, \tag{1.17}$$

for different functions $g : \mathbb{R}^n \to \mathbb{R}$, e.g., $g(\mathbf{x}) = (\mathbf{x}_{\text{CM},i} - \mathbf{x}_i)^2$ for the pointwise conditional variance for the ith component of $\mathbf{x}$. Computing this expectation requires the evaluation of a high-dimensional integral, which is intractable using standard numerical integration techniques. Instead, the expectation is estimated using Monte Carlo methods

$$\mathbb{E}_{\mathbf{x}\sim p^{\text{post}}(\mathbf{x}|\mathbf{y})}[g(\mathbf{x})] \approx \frac{1}{N}\sum_{i=1}^{N} g(\mathbf{x}^{(i)}), \quad \mathbf{x}^{(i)} \sim p^{\text{post}}(\mathbf{x}|\mathbf{y}). \tag{1.18}$$

Monte Carlo estimators show a slow convergence rate of $\mathcal{O}(N^{-1/2})$, using the law of large numbers [111], but crucially are independent of the dimension of the image n and thus do not suffer from the curse-of-dimensionality. Howver, computing the Monte Carlo estimator requires samples of the posterior.

Sampling from the Posterior There are two main families of methods for sampling from the posterior distribution: Markov Chain Monte Carlo (MCMC) [71] and Variational Inference (VI) [31].

MCMC defines an iterative sampling procedure with the posterior as the stationary distribution. For high-dimensional imaging problems, Langevin MCMC is one of the most used variants [155, 118] and has extensions to incorporate non-differentiable priors like total variation [154]. However, MCMC methods often require a large number of iterations to converge.

In contrast, VI frames the sampling problem as an optimization problem. Let $\{p_\theta | \theta \in \Theta\}$ be a family of probabilistic models. We often require that these models are *tractable*, which means that they allow for efficient sampling, likelihood computation, or both. Then, the probabilistic model $p_\theta(\mathbf{x})$ that best approximates the posterior is identified. A common choice, to measure the difference between $p_\theta(\mathbf{x})$ and the posterior $p^{\text{post}}(\mathbf{x}|\mathbf{y})$, is the Kullback-Leibler (KL) [116] divergence

$$\begin{aligned} D_{\text{KL}}(p_\theta(\mathbf{x}) || p^{\text{post}}(\mathbf{x}|\mathbf{y})) &= \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x})}{p^{\text{post}}(\mathbf{x}|\mathbf{y})}\right] \\ &= \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[-\log p^{\text{post}}(\mathbf{x}|\mathbf{y}) + \log p_\theta(\mathbf{x})]. \end{aligned} \tag{1.19}$$

Using Bayes' theorem for the posterior and dropping all terms independent of θ, the goal in VI is to recover

$$\hat{\theta} = \underset{\theta\in\Theta}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[-\log p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}) - \log \pi(\mathbf{x}) + \log p_\theta(\mathbf{x})], \tag{1.20}$$

and use samples from the variational model $p_{\hat{\theta}}(\mathbf{x})$ as a surrogate for samples from the true posterior. The optimization has to be performed for every new measurement $\mathbf{y}$, which makes VI costly in applications where evaluating the forward operator and likelihood is expensive [181]. In *amortized* VI, we instead learn a conditional variational model $p_\theta(\cdot|\mathbf{y})$, which minimizes the expected KL divergence over measurements $\mathbf{y}$ [108, 131, 150]. The class of probabilistic models $\{p_\theta | \theta \in \Theta\}$ is critical for the success of VI and amortized VI. In Chapter 2, we will discuss how normalizing flows, based on invertible neural networks, can be used for VI. Both MCMC sampling and VI require access to an analytical formulation of both the likelihood and the prior.

The likelihood is given by the knowledge about the physics of the system and the noise model. However, the choice of a suitable prior is not as straightforward. The prior aims at encoding existing knowledge about the solution and can significantly influence the posterior and all estimates determined from it. A promising method is to choose the prior empirically given data, see for example [63]. In the next sections, we will discuss methods of learning a prior from data.

1.2 Deep Learning

Deep learning is a subfield of machine learning, which studies the questions of how machines or programs can learn from data [75, 158]. Many modern machine learning algorithms are implemented using neural networks. Deep

learning refers to the use of neural networks with a large number of trainable parameters and a large number of so-called layers. From a mathematical perspective, deep learning offers an interesting combination of concepts from statistical learning, data analysis, and optimization theory.

In this section, we give a short introduction to statistical learning theory, the theoretical framework underpinning deep learning. Next, we will introduce feedforward neural networks and training algorithms. Lastly, we will cover convolutional neural networks, a network architecture specifically designed to work with images.

1.2.1 Statistical Learning Theory

Statistical learning deals with the principles behind *learning from data* and forms the basis of machine learning theory, see e.g. [84, 206]. In statistical learning theory, learning from data is framed as a function estimation problem. In the case of supervised learning, we have access to a dataset $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ of inputs $\mathbf{x}^{(i)} \in \mathcal{X}$ and labels $\mathbf{y}^{(i)} \in \mathcal{Y}$ and want to find a function $\hat{f} : \mathcal{X} \to \mathcal{Y}$ such that $\hat{f}(\mathbf{x}^{(i)}) \approx \mathbf{y}^{(i)}$ for all $i = 1, \ldots, N$. Assume, in a general setting that we have access to the full joint probability distribution $p(\mathbf{x}, \mathbf{y})$ of inputs $\mathbf{x}$ and labels $\mathbf{y}$. The goal of statistical learning is to find a function $f \in \mathcal{H}$ such that the *risk*

$$L_{p(\mathbf{x},\mathbf{y})}(f) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p(\mathbf{x},\mathbf{y})}[\ell(f, (\mathbf{x}, \mathbf{y})] \tag{1.21}$$

is minimized, where $\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}_{\geq 0}$ is a loss function and $\mathcal{H}$ is the hypothesis space. The hypothesis space is the search space and elements $f \in \mathcal{H}$ are called *models*. The loss function ℓ measures the discrepancy between the model output and the label and is defined for the task at hand. For regression tasks, for example, arising in image reconstruction, the loss function is usually chosen as the mean squared error. For image segmentation or classification, a cross-entropy loss function is employed.

In general, the risk in Eqn. (1.21) can not be computed as the joint distribution is unknown. Instead, the principle of empirical risk minimization (ERM) is employed

$$\hat{f}_{\text{ERM}} = \underset{f\in\mathcal{H}}{\operatorname{argmin}} \left\{ L_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^{N} \ell\left(f, (\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right) \right\}, \tag{1.22}$$

where $L_{\text{emp}}(f)$ is the empirical estimate of the risk.

There are two common failure modes associated with empirical risk

minimization and the hypothesis class $\mathcal{H}$. The first is *underfitting*, i.e., no model in $\mathcal{H}$ can represent the relationship of inputs and labels. The second failure mode is called *overfitting*. For overfitting, the model $\hat{f}_{\text{ERM}}$ achieves a low empirical risk, but the risk $L_{p(\mathbf{x},\mathbf{y})}(\hat{f}_{\text{ERM}})$ is not minimized, i.e., $L_{p(\mathbf{x},\mathbf{y})}(\hat{f}_{\text{ERM}}) \gg L_{\text{emp}}(\hat{f}_{\text{ERM}})$. Cross-validation techniques can be employed to diagnose overfitting [141, 192]. However, cross-validation requires the retraining of the model on different subsets of the data and is thus computationally expensive for large deep learning models. Instead, in deep learning the dataset is split into two parts; a training dataset and a validation dataset [75]. The training dataset is used for empirical risk minimization and the loss on the validation dataset is tracked to estimate generalization capabilities and diagnose overfitting. Generalization capabilities depend crucially on the choice of the hypothesis class [139]. The hypothesis class can be restricted to include prior knowledge, referred to as *inductive bias*. In image classification tasks, the label $\mathbf{y}$ is often independent of the rotation of the image $\mathbf{x}$. This information can be encoded in the hypothesis class, for example through the use of group equivariant convolutions, see [46]. Another way to tackle overfitting is to introduce regularisation terms to the ERM objective [169].

Besides supervised learning, there exist many other learning tasks [75]. Important for this thesis is *unsupervised* learning, where only inputs $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ without the associated labels are provided. Typical tasks here include generative modeling, which will be discussed in Section 1.3, dimensionality reduction, or clustering. Various learning tasks arising in solving inverse problems will be presented in Section 1.4.

1.2.2 Feedforward Neural Networks

The prototypical example of a neural network is the feedforward neural network, without any recurrent connections. Feedforward neural networks are composed of several building blocks, referred to as *layers* [75]. Each layer consists of an affine linear transformation and a non-linear activation function $\sigma : \mathbb{R} \to \mathbb{R}$. The linear transformation has learnable parameters, called *weights* and *biases*. For a fully-connected network, a single layer is defined as follows. Let $\mathbf{h}^l \in \mathbb{R}^{d_l}$ be the input to the l-th layer, the output $\mathbf{h}^{l+1} \in \mathbb{R}^{d_{l+1}}$ is given by

$$\mathbf{h}^{l+1} = \Phi^l(\mathbf{h}^l) =: \boldsymbol{\sigma}(\mathbf{W}^l\mathbf{h}^l + \mathbf{b}^l), \tag{1.23}$$

where $\boldsymbol{\sigma} : \mathbb{R}^{d_{l+1}} \to \mathbb{R}^{d_{l+1}}$ is defined by

$$\boldsymbol{\sigma}(\mathbf{h}) = (\sigma(\mathbf{h}_1), \ldots, \sigma(\mathbf{h}_{d_{l+1}})), \quad \mathbf{h} \in \mathbb{R}^{d_{l+1}}, \tag{1.24}$$

and denotes the pointwise application of the activation function σ. The weights $\mathbf{W}^l$ are a $d_{l+1} \times d_l$ matrix and the bias $\mathbf{b}^l \in \mathbb{R}^{d_{l+1}}$ is a vector. Thus, a single fully connected layer has $d_{l+1}d_l + d_l$ learnable parameters. The activation function σ is usually fixed. One of the most widely used activation functions is the rectified linear unit (ReLU) [145] defined by

$$\sigma(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}. \tag{1.25}$$

A fully-connected neural network with L layers is defined by iteratively stacking the layers in Eqn. (1.23) as

$$\begin{aligned} \Phi^1(\mathbf{x}) &= \boldsymbol{\sigma}^1(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1), \\ \Phi^{l+1}(\mathbf{x}) &= \boldsymbol{\sigma}^{l+1}(\mathbf{W}^{l+1}\Phi^l(\mathbf{x}) + \mathbf{b}^{l+1}), \quad \text{for } l = 1, \ldots, L-1, \end{aligned} \tag{1.26}$$

and $f(\mathbf{x}, \theta) := \Phi^L(\mathbf{x})$ is the final output of the neural network with parameters $\theta = (\mathbf{W}^1, \mathbf{b}^1, \ldots, \mathbf{W}^L, \mathbf{b}^L)$. The first layer Φ^1 is called the *input layer*, the last layer Φ^L the *output layer* and the intermediate layers are called *hidden layers*.

1.2.3 Training a Neural Network

Training a neural network involves minimizing the empirical risk, typically with the inclusion of regularization terms. However, the training problem differs from classical optimization problems in some important aspects. Since the empirical risk serves only as a surrogate for the actual risk, it is not necessary to minimize the empirical risk exactly. Often, early stopping rules are applied based on the empirical risk of a held-out validation set. Consequently, training may be stopped even if the gradients of the empirical risk are still large (see for example Section 8 in [75]).

The empirical risk in Eqn. (1.22) decomposes into a sum over all training examples. Due to the linearity, computing the gradient of the empirical risk requires evaluating the network for all examples in the dataset. This is prohibitive for large datasets with thousands or even millions of data points. For training deep learning models stochastic gradient descent (SGD) methods are preferred [66, 75]. For SGD, the gradient of the empirical risk

is approximated by

$$\frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \ell(f_\theta,(\mathbf{x}^{(i)},\mathbf{y}^{(i)})) \approx \frac{1}{M}\sum_{i\in\mathcal{B}}\nabla_\theta \ell(f_\theta,(\mathbf{x}^{(i)},\mathbf{y}^{(i)})), \tag{1.27}$$

where $M = |\mathcal{B}|$ is called the batch size and $\mathcal{B} \subset \{1,\dots,N\}$ the batch, randomly drawn from the full training dataset. The term *stochastic* in this context refers to the fact that the gradient of the empirical risk is approximated using a random subset of the full dataset. Besides saving computational cost, SGD has some interesting theoretical properties. Under some assumptions, SGD can have the same convergence rate as methods making use of the gradient of the full dataset [171]. It has been observed in experiments, that SGD often leads to a better generalization error and faster convergence with respect to the total computational cost of the training algorithm [213]. Further, the estimation using a subset of the full dataset is also motivated by the fact that we can expect a correlation between examples in large-scale datasets, e.g., in a classification task we might have two very similar images of the same object. The update rule for SGD is given by

$$\theta^{k+1} = \theta^k - \gamma^k \frac{1}{M}\sum_{i\in\mathcal{B}^k}\nabla_\theta \ell(f_\theta,(\mathbf{x}^{(i)},\mathbf{y}^{(i)})), \tag{1.28}$$

where the batch $\mathcal{B}^k$ is randomly chosen for each update and $\gamma^k > 0$ is the learning rate. The gradient of the loss function for each example is computed using the *backpropagation* algorithm [167], which is an efficient way for computing the gradients in a neural network. A common extension of the update rule in Eqn. (1.28) is to include momentum terms or use adaptive learning rates, for example, resulting in the widely used Adam algorithm [106].

1.2.4 Convolutional Neural Networks

A digital or discrete image of size $d \times d$ is defined as a tensor $\mathbf{x} = (x_{i,j,k})_{i,j,k}$ with $i,j = 1,\dots,d$ as the spatial dimensions and $k = 1,\dots,c$ representing intensity channels [177]. Later, we often work with vectorized images, i.e. $\mathbf{x} \in \mathbb{R}^n$ with $n = d^2k$. For RGB images, the number of channels $c = 3$ represents red, green, and blue intensity values and for grayscale images $c = 1$ represents the gray value, usually between 0 and 1. Many medical images can be represented as grayscale images, for example in CT the grayscale value encodes the density at the specific pixel. Image processing with neural networks requires specialized network architectures due to the inherent

properties of images [75, 158]. First, images are generally high dimensional. Medical images used in deep learning often have a spatial dimension of at least 256×256px. A fully-connected layer, with the same input and output dimensions, would already have over 4 Billion parameters for images of this size. This makes employing classical fully-connected neural networks for imaging tasks impossible. Secondly, images have a high spatial correlation of neighboring pixels, which is not taken into account for fully-connected layers. Further, in many machine learning tasks images are stable under certain geometric transformations, i.e., small shifts or rotations do not change the content. Convolutional layers are designed to model these assumptions. A multi-channel convolution for an input image $\mathbf{h}^l \in \mathbb{R}^{d \times d \times c_1}$ with spatial dimensions $d \times d$ and c_1 channels and an output $\mathbf{h}^{l+1} \in \mathbb{R}^{d \times d \times c_2}$ with c_2 channels is defined as the correlation with a filter $\mathbf{w} \in \mathbb{R}^{(2k+1) \times (2k+1) \times c_1 \times c_2}$

$$\mathbf{h}^{l+1}_{i,j,\tilde{c}} = \sum_{c=0}^{c_1} \sum_{s=-k}^{k} \sum_{r=-k}^{k} \mathbf{w}_{s,r,c,\tilde{c}} \mathbf{h}^l_{i+s,j+r,c}, \tag{1.29}$$

for $i = 1, \dots, d$, $j = 1, \dots, d$ and $\tilde{c} = 1, \dots, c_2$. As $\mathbf{h}^l$ is only defined on the $d \times d$ pixel grid, the values outside of the domain have to be defined. This is referred to as *padding* and a common choice is zero padding, i.e., setting all elements outside of the $d \times d$ pixel grid to zero, e.g., $\mathbf{h}^l_{-1,-1,c} \coloneqq 0$. The number of channels is typically increased in the hidden layers for many deep learning architectures [75, 182]. The definition in Eqn. (1.29) is restricted to square filters ($k \times k$) and images on a square $d \times d$ pixel grid but can be extended to rectangle pixel grids and filters. For a 1×1 convolution, i.e., $k = 0$, the convolutional layer can be written as a linear transformation of the channels, repeated as every pixel location. Let $\mathbf{h}^{l+1}_{i,j,\cdot} \in \mathbb{R}^{c_2}$ be the vector of channels at location (i, j) and $\mathbf{h}^l_{i,j,\cdot} \in \mathbb{R}^{c_1}$, then we have

$$\mathbf{h}^{l+1}_{i,j,\cdot} = \mathbf{W}^l \mathbf{h}^l_{i,j,\cdot}, \tag{1.30}$$

where $\mathbf{W}^l \in \mathbb{R}^{c_2 \times c_1}$ is the filter matrix, i.e., $\mathbf{W}^l_{\tilde{c},c} = \mathbf{w}_{c,\tilde{c}}$.

A full convolutional layer consists of a multi-channel convolution with activation function and bias

$$\mathbf{h}^{l+1}_{i,j,\tilde{c}} = \sigma(\mathbf{h}^{l+1}_{i,j,\tilde{c}} + b_{\tilde{c}}), \tag{1.31}$$

where $\mathbf{b} \in \mathbb{R}^{c_2}$ is the bias for this layer, which is shared for all image pixels, and the activation function is applied element-wise. There exist a variety of extensions to this basic multi-channel convolution, for example, using strided or grouped convolutions [144].

Besides multi-channel convolutions, most CNNs also include downsampling layers, to reduce the spatial dimension. This helps to reduce the computational cost and makes the CNNs approximately invariant to small translations [75]. One of the most widely used architectures for image-to-image tasks, including image reconstruction and image segmentation, is the U-Net proposed by Ronneberger et al. [168]. The U-Net is an extension of fully convolutional networks [125]. It consists of two parts: a contracting part and an expansive part. Modern variants of the U-Net typically also include attention mechanisms in both the contracting and expansive part [55].

1.3 Generative Modeling

Deep generative models have many applications in a lot of different research areas. Two important applications are based on either estimating the likelihood of new observations, or generating new samples from the underlying distribution. Evaluating the likelihood allows for out-of-distribution or novelty detection [219], which is important in medical or industrial applications. The generation of new samples is by now widely known for the use of *deep fakes* [43] in particular in combination with the recent advent of large-scale text-to-image diffusion model [215]. Applications in inverse problems include the approximation of posterior or prior distributions and will be extensively discussed in the second chapter of this thesis.

1.3.1 Setting

The goal of generative modeling is to model and estimate an unknown distribution from a given dataset [75, 173, 186]. Let $\{\mathbf{x}^{(i)}\}_{i=1}^{N} \sim p_{\text{data}}(\mathbf{x})$ denote the dataset, where we assume that all elements are i.i.d. samples from the underlying data distribution $p_{\text{data}}(\mathbf{x})$. Similar to Section 1.1.2, we denote both the probability distribution and the density with $p(\cdot)$ and focus on finite-dimensional distributions.

In generative modeling, we choose a specific family of probabilistic models $\{p_\theta |\ \theta \in \Theta\}$ to model the data distribution $p_{\text{data}}(\mathbf{x})$. The probabilistic models p_θ are parametrized by parameters $\theta \in \Theta$, where Θ is the set of admissible parameter values. The goal is to identify parameters $\hat{\theta}$ such that

$$p_{\hat{\theta}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x}). \tag{1.32}$$

The probabilistic model $p_\theta(\mathbf{x})$ can then be used as a surrogate for the unknown data distribution to facilitate sampling and density estimation. In the context of deep learning, the probabilistic model $p_\theta(\mathbf{x})$ is typically denoted as a *generative model* and, in particular, if $p_\theta(\mathbf{x})$ is constructed using deep neural networks, as a *deep generative model* [173].

1.3.2 Density estimation as an ill-posed problem

Estimating the probability density function is an ill-posed inverse problem. We present an example by Vapnik [206]. Assume the case of $n = 1$, i.e., one-dimensional density estimation. Further, assume we have a dataset of i.i.d. samples $\{x^{(1)}, \ldots, x^{(N)}\}$ from the target distribution with density p. The density function is related to the distribution function via the integral equation

$$\int_{-\infty}^{x} p(x')dx' = F(x), \tag{1.33}$$

where we only have access to the empirical distribution function

$$F(x) \approx F_N(x) = \frac{1}{N}\sum_{i=1}^{N} H(x - x^{(i)}), \tag{1.34}$$

with $H : \mathbb{R} \to \{0, 1\}$ as the Heaviside function. The corresponding inverse problem, i.e., estimating the derivative from the antiderivative, is ill-posed.

This highlights the fact, that generative modeling requires regularization. In generative modeling, this regularization usually comes in the form of choosing a specific family of probabilistic models $\{p_\theta |\ \theta \in \Theta\}$.

1.3.3 Fitting the Model

Modern generative modeling is framed as an approximation problem [75, 173]. To find parameters $\hat{\theta}$, such that $p_{\hat{\theta}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$, we minimize the discrepancy of the probabilistic model to the target distribution. The goodness of fit is measured with respect to the KL divergence

$$D_{\text{KL}}(p_{\text{data}}(\mathbf{x})||p_\theta(\mathbf{x})) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log\left(\frac{p_{\text{data}}(\mathbf{x})}{p_\theta(\mathbf{x})}\right)\right], \tag{1.35}$$

where $D_{\mathrm{KL}}(p_{\mathrm{data}}(\mathbf{x})||p_\theta(\mathbf{x})) = 0$ if and only if the two distributions are the same. The optimal parameters are obtained by minimizing the KL divergence

$$\begin{aligned}
&\operatorname*{argmin}_{\theta\in\Theta} D_{\mathrm{KL}}(p_{\mathrm{data}}(\mathbf{x})||p_\theta(\mathbf{x})) \\
&= \operatorname*{argmin}_{\theta\in\Theta} \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}(\mathbf{x})}\left[\log p_{\mathrm{data}}(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}(\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \\
&= \operatorname*{argmin}_{\theta\in\Theta} \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}(\mathbf{x})}[-\log p_\theta(\mathbf{x})],
\end{aligned} \tag{1.36}$$

which reduces to the minimization of the negative log-likelihood. As the target distribution is unknown, we use the empirical estimate

$$\hat{\theta} = \operatorname*{argmin}_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} -\log p_\theta(\mathbf{x}^{(i)}), \tag{1.37}$$

as a proxy for Eqn. (1.36) using the available dataset $\{\mathbf{x}^{(i)}\}_{i=1}^N \sim p_{\mathrm{data}}(\mathbf{x})$. Besides the KL divergence, general f-divergences of the form

$$D_f(p_{\mathrm{data}}(\mathbf{x})||p_\theta(\mathbf{x})) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}(\mathbf{x})}\left[f\left(\frac{p_{\mathrm{data}}(\mathbf{x})}{p_\theta(\mathbf{x})}\right)\right], \tag{1.38}$$

with a convex function $f : [0, +\infty) \to (-\infty, +\infty]$, have been explored for generative modeling [13, 151].

1.3.4 Challenges

Not every neural network defines a generative model $p_\theta(\mathbf{x})$. As $p_\theta(\mathbf{x})$ models the density function of a distribution, it has to fulfill two important restrictions: non-negativity and normalization. While non-negativity is often trivial to enforce for neural network architectures, the normalization constraint, i.e.

$$\int_{\mathbb{R}^n} p_\theta(\mathbf{x}) d\mathbf{x} = 1 \quad \text{for all } \theta \in \Theta, \tag{1.39}$$

is much harder to fulfill.

One class of models, which directly parametrize the probabilistic model, are *energy-based models* (EBM) [1, 119, 161]. In an EBM, the probabilistic model $p_\theta(\mathbf{x})$ is parametrized as

$$p_\theta(\mathbf{x}) = \exp\{-f_\theta(\mathbf{x})\}/Z_\theta \quad \text{with } Z_\theta = \int_{\mathbb{R}^n} \exp\{-f_\theta(\mathbf{x})\} d\mathbf{x}, \tag{1.40}$$

where $f_\theta : \mathbb{R}^n \to \mathbb{R}$ is a neural network, without any architectural restrictions, and Z_θ is the normalization constant. In deep learning frameworks, Eqn. (1.37) is typically minimized by gradient descent, which requires the

evaluation of the gradient

$$-\nabla_\theta \log p_\theta(\mathbf{x}) = \nabla_\theta f_\theta(\mathbf{x}) + \nabla_\theta \log Z_\theta. \tag{1.41}$$

The first part can be obtained using the backpropagation algorithm as discussed in Section 1.2.3. The second term, $\nabla_\theta \log Z_\theta$, can be estimated as

$$\nabla_\theta \log Z_\theta = \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[\nabla_\theta f_\theta(\mathbf{x})], \tag{1.42}$$

and requires access to samples from the EBM [90]. The samples are usually obtained using Langevin MCMC methods. For each training step, new samples are required, making the training computationally expensive. However, there exists a lot of work to speed up the training, for example, using a persistent MCMC chain throughout the training [200].

Instead of a direct parametrization of the probabilistic model, most modern generative modeling frameworks construct $p_\theta(\mathbf{x})$ as a transformation of some tractable base distribution. Here, different methods, i.e., constraining the architecture or using a surrogate loss, are employed.

1.3.5 Implicit Generative Modeling

Most current deep learning frameworks use an implicit approach to modeling the density [173]. This means, that the density is not parametrized directly by a neural network. Instead, $p_\theta(\mathbf{x})$ is implicitly parameterized as a transformation of some base distribution $p_\mathbf{z}(\mathbf{z})$, usually chosen as a Gaussian. For this approach, the goal is to learn a *generator* $g_\theta : \mathbb{R}^k \to \mathbb{R}^n$, that pushes the base distribution $p_\mathbf{z}(\mathbf{z})$ to the desired image space. The dimensionality of the so-called *latent space* $\mathbb{R}^k$ is a crucial choice in the modeling process and determines which training techniques can be applied.

Equal Dimensionality

First, there are approaches using an equal dimensionality of the latent space and the image space, i.e., $k = n$. In this case, any measurable generator g_θ defines a valid distribution via the pushforward measure $p_\theta = (g_\theta)_\# p_\mathbf{z}$, which enables training based on the maximum likelihood principle. These push-forward models have strong connections to the optimal transport theory [209] and there exists a rich theory about using these push-forward models to learn to sample from a target distribution, see the review [132]. Moreover, if the generator is invertible, these models are denoted as normalizing flows, which will be covered in detail in Section 2.1. Further, this

theory has been extended to injective generators [114]. Recently, score-based diffusion models [91, 184, 189] where introduced as an alternative modeling approach. Score-based diffusion models define a forward diffusion process mapping the data distribution $p_{\text{data}}(\mathbf{x})$ to the latent distribution $p_{\mathbf{z}}(\mathbf{z})$. It has been shown that there exists a reverse diffusion process mapping the latent distribution back to the data distribution [10]. This reverse diffusion process requires access to the *score* $\nabla_x \log p_{\text{data}}$. The generator g_θ is implicitly defined by this reverse diffusion process. Training the score-based diffusion model requires estimating this score function. A detailed introduction to score-based diffusion models and connections to normalizing flows are given in Section 2.4.

Lower Dimensional Latent Space

Another approach is to choose a lower dimensional latent space, i.e., $k < n$. This is motivated by the manifold hypothesis [27], which states that real-world high-dimensional data, such as images, are concentrated around a low-dimensional manifold. The choice of using a lower dimensional latent space comes with challenges in training such a model. For a lower dimensional latent space, the range of the generator $\text{Range}(g_\theta)$ is a k-dim. manifold and any image $\mathbf{x}$ has probability zero almost everywhere under the generator, which makes maximum likelihood training intractable. Variational autoencoders (VAE) [108] define a density on the full image space by prescribing an observation model $p_g(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; g_\theta(\mathbf{z}), \eta \mathbf{I}_n)$ such that

$$p_\theta(\mathbf{x}) = \int_{\mathbb{R}^k} p_g(\mathbf{x}|\mathbf{z}) p_{\mathbf{z}}(\mathbf{z}) d\mathbf{z} \tag{1.43}$$

defines a probability density. However, evaluating the likelihood requires the computation of a high-dimensional integral, which is intractable for training. VAE define an additional trainable encoder $e_\psi(\mathbf{z}|\mathbf{x})$, usually implemented as $e_\psi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu_\psi(\mathbf{x}), \Sigma_\psi(\mathbf{x}))$ with two neural networks μ_ψ, Σ_ψ estimating mean and variance. With this additional encoder, the likelihood can be rewritten as

$$\begin{aligned} \log p_\theta(\mathbf{x}) &= \mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x})] \\ &= \mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{e_\psi(\mathbf{z}|\mathbf{x})}\right)\right] + \underbrace{D_{\text{KL}}(e_\psi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))}_{\geq 0} \\ &\geq \mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z})] - \mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\log e_\psi(\mathbf{z}|\mathbf{x})], \end{aligned} \tag{1.44}$$

following the derivation in [173]. This lower bound is referred to as the evidence lower bound (ELBO). Using the Gaussian observation mode, the ELBO can be rewritten as

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z})] - \mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\log e_\psi(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\log p_g(\mathbf{x}|\mathbf{z}) - \log p_{\mathbf{z}}(\mathbf{z})] - \mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\log e_\psi(\mathbf{z}|\mathbf{x})] \\
&= -\frac{1}{2\eta^2}\mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\|g_\theta(\mathbf{z}) - \mathbf{x}\|_2^2] - \mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\log e_\psi(\mathbf{z}|\mathbf{x})] + \text{ const.},
\end{aligned} \tag{1.45}$$

where the first term includes a reconstruction loss $\mathbb{E}_{\mathbf{z}\sim e_\psi(\mathbf{z}|\mathbf{x})}[\|g_\theta(\mathbf{z}) - \mathbf{x}\|_2^2]$, similar to traditional autoencoders. Maximizing the ELBO thus maximizes a lower bound to the log-likelihood. Both the parameters of the encoder ψ and the parameters of the probabilistic θ are optimized at the same time. Sampling from a VAE is a two-step process, first a sample from the latent distribution is drawn, which is then used as an input to $p_g(\mathbf{x}|\mathbf{z})$, i.e.,

$$\mathbf{x} \sim p_\theta(\mathbf{x}) \Leftrightarrow \mathbf{x} = g_\theta(\mathbf{z}) + \epsilon, \qquad \text{with } \mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z}), \epsilon \sim \mathcal{N}(0, \eta^2\mathbf{I}_n). \tag{1.46}$$

Even though $g(\mathbf{z})$ might be a clean image, the final sample of the VAE requires evaluation of the Gaussian observation model, which amounts to adding noise with a fixed variance to $g_\theta(\mathbf{z})$. In particular for the Gaussian observational model, the images produced by a VAE thus often have worse visual quality than other generative models [216].

Generative adversarial networks (GANs) [76] circumvent the problem of dealing with the likelihood by only training a generator for sampling. The difference between samples from the GAN and the dataset is measured directly in image space. However, in contrast to the VAE, this difference is not measured using an $L2$ loss function. Rather, training a GAN can be interpreted as a zero-sum-game, where an additional neural network $d_\phi : \mathbb{R}^n \to [0, 1]$ is trained to distinguish between samples from the training data, i.e., $d_\phi(\mathbf{x}) \approx 1$ for $\mathbf{x}$ in the training data, and samples from the GAN, i.e., $d_\phi(g_\theta(\mathbf{z})) \approx 0$. This can be encoded in the loss function

$$\mathcal{L}_{\text{GAN}}(\theta, \phi) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log d_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - d_\phi(g_\theta(\mathbf{z})))], \tag{1.47}$$

which is maximized with respect to the discriminator d_ϕ and minimized with respect to the GAN g_θ. Due to directly measuring the distance of samples in the image space, the visual quality of samples is often superior when compared to likelihood-based models [174]. However, the training of a GAN can be unstable and suffer from mode collapse. There exist different variations, for example, the Wasserstein GAN [13, 79], which propose different loss functions to enable more stable training.

1.4 Application to Inverse Problems

Deep learning offers immense opportunities to modify the model-based approach to inverse problems with novel data-driven techniques. The usage of available data can help to include prior knowledge in the reconstruction process, for example by adapting a reconstruction method to perform better on a specific image manifold [3] or learning to correct errors of the forward operator [129]. Data-driven techniques have already been explored prior to the advent of deep learning. Early examples include dictionary learning [203], basis-constrained reconstruction [207] or the estimation of regularization parameters based on available data [198]. However, due to new developments in hardware and the collection of large datasets, these data-based methods can now be implemented on a larger scale.

In recent years a plethora of deep learning methods have been proposed to tackle various inverse problems. Two main characteristics can be identified in which these methods differ: 1) what kind of data is used for training and 2) how much knowledge about the physical system is included. Ongie et al. [149] provides a comprehensive taxonomy categorizing different approaches by these two categories. For example supervised methods vs. unsupervised, postprocessing vs. learned-iterative, or generative vs. functional analytic methods. There exist many good reviews covering different methods, see for example [16, 133, 149, 176].

The theoretical foundation of many of these data-driven models is still lacking, compared to classical regularization. There has been a push to combine data-driven techniques with classical methods to provide some theoretical guarantees, see for example the recent review [143].

In this introduction, we cover different concepts, which are important in the scope of this thesis. These include learned reconstructors, learned regularizers, posterior estimation, and untrained models.

1.4.1 Learned Reconstruction

Training a parametrized reconstruction method in the supervised framework requires access to a paired dataset $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ of measurements $\mathbf{y}$ and corresponding ground truth images $\mathbf{x}$. The goal is to find parameters $\hat{\theta} \in \Theta$ of a parametrized reconstruction operator $\mathcal{R}_\theta : \mathcal{Y} \to \mathcal{X}$, such that

$$\mathcal{R}_{\hat{\theta}}(\mathbf{y}) \approx \mathbf{x}. \tag{1.48}$$

The trained operator can then be used in place of more traditional reconstruction methods. Using the paired dataset, the parameters $\hat{\theta}$ can be recovered by ERM

$$\hat{\theta} = \operatorname*{argmin}_{\theta \in \Theta} \sum_{i=1}^{N} \ell\left(\mathbf{x}^{(i)}, \mathcal{R}_\theta(\mathbf{y}^{(i)})\right), \tag{1.49}$$

with a suitable loss function $\ell : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$. After this initial training phase, the reconstructions are obtained as $\hat{\mathbf{x}} = \mathcal{R}_{\hat{\theta}}(\mathbf{y})$. The learning approach crucially depends on the parametrization of the reconstructor $\mathcal{R}_\theta$. In some works the full mapping from the measurements to the image is implemented as a neural network, for example, AUTOMAP [217] or iRadon [86], both for CT reconstruction. However, these approaches often require a larger dataset to achieve similar quality to other learned reconstruction approaches, see the empirical comparison in [19]. Further, these approaches require specialized architectures to parametrize the mapping as the measurement space $\mathcal{Y}$ and the image space $\mathcal{X}$ often have a different topology. For example in CT, the measurements are line integrals and $\mathcal{X}$ is defined as the space of discrete images. Typically more successful are learned reconstructors, which incorporate knowledge about the inverse problem directly into the architecture. These methods can be classified into postprocessing (two-step) methods [149, 179] and learned iterative methods [3, 78, 159].

Postprocessing methods parametrize the reconstructor as $\mathcal{R}_\theta = \tilde{\mathcal{R}}_\theta \circ \mathbf{A}^\dagger$, where $\mathbf{A}^\dagger : \mathcal{Y} \to \mathcal{X}$ is some initial (classical) reconstruction method and $\tilde{\mathcal{R}}_\theta : \mathcal{X} \to \mathcal{X}$ is a neural network only acting upon the image space $\mathcal{X}$. In this two-step process, the network $\tilde{\mathcal{R}}_\theta$ learns to remove noise and artifacts from the provided initial reconstruction. From another point of view, the postprocessing approach can be understood as a type of data preprocessing. Instead of training a model on a supervised dataset $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$, the measurements are replaced with initial reconstructions $\tilde{\mathbf{x}}^{(i)} = \mathbf{A}^\dagger \mathbf{y}^{(i)}$. This defines a new supervised dataset $\{(\mathbf{x}^{(i)}, \tilde{\mathbf{x}}^{(i)})\}_{i=1}^N$, which can be directly used to train the network $\tilde{\mathcal{R}}_\theta$. As the postprocessing network $\tilde{\mathcal{R}}_\theta$ only acts on images, the architecture can be efficiently implemented as a CNN.

For learned iterative methods, an iterative reconstruction algorithm is unrolled for a fixed number of steps, and some components are exchanged with trainable neural networks. As an example, assume we have the variational regularization objective from Eq. (1.3) with a convex regularizer. The reconstruction can be obtained by gradient descent

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \lambda\left(\mathbf{A}^*(\mathbf{A}\mathbf{x}^k - \mathbf{y}) - \alpha \nabla_x \mathcal{S}(\mathbf{x}^k)\right). \tag{1.50}$$

Instead of using this additive update for the current iterate, the update rule can be learned using a neural network $f_{\theta_k} : \mathcal{X} \times \mathcal{X} \times \mathcal{X} \to \mathcal{X}$, i.e.,

$$\mathbf{x}^{k+1} = f_{\theta_k}(\mathbf{x}^k, \mathbf{A}^*(\mathbf{A}\mathbf{x}^k - \mathbf{y}), \nabla_x \mathcal{S}(\mathbf{x}^k)), \ k = 0, \dots, K-1. \tag{1.51}$$

This unrolled iterative process defines a parametrized reconstruction operator $\mathcal{R}_\theta(\mathbf{y}) \coloneqq \mathbf{x}^K$, which directly incorporates the forward, adjoint, and gradients of the operator into the network architecture. Note that the learned network f_{θ_k} acts only on images, similar to the postprocessing network, and similar CNN architectures can be used. The technique of *unrolling* an iterative scheme was popularized by LISTA [78], a learned unrolled version of ISTA [69] for sparse coding. This research has led to a variety of learned iterative networks inspired by different classical iterative methods, for example, learned gradient descent [3], learned primal-dual [4] or the deep AADM-Net [195]. However, even though these networks are inspired by existing algorithms, all convergence properties of the original iterative algorithm are lost during the unrolling and learning process. Recently, a lot of research focused on incorporating guarantees into these learned iterative algorithms by introducing some constraints in the architecture, see for example, [85], but the empirical performance is still worse compared to the unconstrained counterparts.

The computational effort of learned reconstructors is quite different from the classical variational regularization framework. For the learned reconstructors, the main computational effort is spent for the initial training phase (see Eqn. (1.49)), which can take multiple days to weeks (see for example the training times in [121]), whereas the evaluation of the final model is fast, usually one a few milliseconds [149]. This is especially the case for reconstructors constructed using CNNs, which are fast to evaluate due to hardware accelerations on GPUs.

1.4.2 Learned Regularizers

The choice of the regularizer in the variational regularization framework is an important aspect and can drastically affect the performance. The prior can be hand-crafted to promote desired features in the reconstructed image, such as sparsity of edges [172] or smoothness. However, modeling the higher-order statistics or assumption of natural images by hand can become increasingly difficult. Therefore, quite early, frameworks for learning the regularizer from data have been developed, see [218] as an example for natural images. This

means, that the variational regularization objective changes to

$$\mathcal{R}_\theta(\mathbf{y}_\delta) = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{X}} \|\mathbf{A}\mathbf{x} - \mathbf{y}_\delta\|_{\mathcal{X}}^2 + \mathcal{S}_\theta(\mathbf{x}), \tag{1.52}$$

where $\mathcal{S}_\theta : \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a parametrized regularizer with parameters θ that should be learned from data. There exist different methods of obtaining suitable parameters θ.

Bilevel Optimization The parameters θ can be learned in a supervised framework, see for example [16, 50, 80]. Assume, we have a paired dataset $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$. In the bilevel optimization approach, we choose $\hat{\theta}$ as the minimizer of

$$\begin{aligned} \hat{\theta} &\in \operatorname*{argmin}_\theta \sum_{i=1}^N \|\mathcal{R}_\theta(\mathbf{y}^{(i)}) - \mathbf{x}^{(i)}\|_2^2, \\ \text{where } \mathcal{R}_\theta(\mathbf{y}) &= \operatorname*{argmin}_{\mathbf{x}\in\mathcal{X}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \mathcal{S}_\theta(\mathbf{x}). \end{aligned} \tag{1.53}$$

Bilevel optimization can be computationally expensive, as for most iterative methods the lower-level optimization problem has to be solved for every iteration [36]. Other approaches based on constrained optimization typically do not scale to large datasets [80].

Plug-and-Play Unsupervised regularizers are usually more flexible, in the sense that the same learned regularizer can be applied for several different inverse problems, as long as the underlying image distribution does not change [61]. An example here is the Plug-and-Play (PnP) framework. For convex (not necessarily differentiable) regularizers $\mathcal{S} : \mathcal{X} \to \mathbb{R}_{\geq 0}$, we can solve the variational problem in Eqn. (1.3) using proximal algorithms [152]. Proximal gradient descent is given by

$$\mathbf{x}^{k+1} = \operatorname{prox}_{\lambda^k \alpha \mathcal{S}}(\mathbf{x}^k - \lambda^k \mathbf{A}^*(\mathbf{A}\mathbf{x}^k - \mathbf{y})), \tag{1.54}$$

where $\lambda^k > 0$ is the step size and the proximal mapping is defined by

$$\operatorname{prox}_{\lambda^k \alpha \mathcal{S}}(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{u}} \frac{1}{2}\|\mathbf{x} - \mathbf{u}\|_2^2 + \lambda^k \alpha \mathcal{S}(\mathbf{u}). \tag{1.55}$$

Evaluating the proximal mapping can be seen as a denoising task with regularizer $\mathcal{S}$. In PnP algorithms, the proximal mapping is replaced with some off-the-shelf denoiser $D : \mathcal{X} \to \mathcal{X}$ [208]. Here, both classical denoisers, such as BM3D [47], or deep denoisers [134], based on neural networks, have been employed. In order to train a deep denoiser $D_\theta : \mathcal{X} \to \mathcal{X}$, parameterized

by θ, we need a dataset of clean images $\{(\mathbf{x}^{(i)})\}_{i=1}^N$ and train the denoiser by

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,I)}[\|D_\theta(\mathbf{x}^{(i)} + \sigma\mathbf{z}) - \mathbf{x}^{(i)}\|_2^2], \tag{1.56}$$

where σ is a noise level chosen prior. After training, the trained denoiser can be integrated into the PnP algorithm

$$\mathbf{x}^{k+1} = D_\theta(\mathbf{x}^k - \lambda^k \mathbf{A}^*(\mathbf{A}\mathbf{x}^k - \mathbf{y})). \tag{1.57}$$

The PnP framework has been extended to other proximal algorithms, for example to proximal ADMM [70].

Generative Regularizers In the statistical framework, see Section 1.1.2, the variational regularization framework can be interpreted as MAP estimation. In this formulation, the regularizer $\mathcal{S}(\mathbf{x}) = -\log \pi(\mathbf{x})$ is given by the negative log-likelihood of the prior. Learning a regularizer thus is equivalent to learning the prior. Here, any of the probabilistic models from Section 1.3 can be employed. For likelihood-based models, such as the normalizing flow, we can directly use the negative log-likelihood as a regularizer. Generative regularization, in particular with normalizing flows, will be extensively discussed in Section 2.3.2. For likelihood-free models, such as the GAN, the distance to the data manifold can be used as a regularizer, see for example the recent overview article [61].

Adversarial Regularizers The regularizer $\mathcal{S}$ in variational regularization, see Eqn. (1.3), takes on small values for desirable solutions and large values for undesirable solutions[1]. Notably, two learning frameworks make use of this intuition and train a regularizer using a dataset of desirable and undesirable solutions: quotient minimization [29] and the adversarial regularizer [128]. The adversarial regularizer has an extension for learning convex regularizers with proven convergence properties [142].

1.4.3 Learning the Posterior

Access to the posterior enables uncertainty estimation for statistical inverse problems. Further, it allows for exploring different estimators, for example, the MAP or conditional mean, as discussed in Section 1.1.2. In the Bayesian framework, we can decompose the posterior into likelihood and prior. In the

[1]Note that the regularizer actually also distinguishes between solutions with the same function value, see the example in Section 3 of [28].

previous section, we assumed that the likelihood is known, i.e., we have access to the forward operator and the noise model. Under this assumption, only the prior is unknown and has to be learned from data. However, if the forward operator is unknown or if we are unsure about the specific noise model, this approach cannot be used. In this case, we have the option to estimate the full posterior given a supervised dataset $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ of images and corresponding measurements. In contrast to learning the prior, this approach is adapted to a specific forward operator and measurement setup. All modern deep generative models, discussed in Section 1.3, have extensions for conditional density estimation. There are conditional GANs [137], conditional VAEs [109], conditional diffusion models [24] and conditional normalizing flows [214]. In particular conditional normalizing flows and their applications to inverse problems will be discussed in Chapter 2.

1.4.4 Untrained Models

Training a neural network, whether for reconstruction, as a prior or to estimate the posterior, requires a large amount of training data. However, in many domains, especially in medical imaging, obtaining a large dataset can be challenging or costly. In these sparse data settings another class of deep learning models can be applied, categorized for example by Dimakis as *untrained generative models* [56]. Notably, the Deep Image Prior (DIP) [204] offers a promising alternative. DIP has been successfully applied to various image-to-image tasks such as denoising, deblurring, and in-painting [204]. It has also shown promising results in medical imaging, including PET reconstruction [183] and CT reconstruction [19].

In the DIP framework a convolutional neural network $f(\theta, \mathbf{z})$ is initialized with random weights θ. The reconstruction $\hat{\mathbf{x}} = f(\hat{\theta}, \mathbf{z})$ is obtained by optimizing the weights of the network to fit the measurements, i.e.,

$$\hat{\theta} \in \operatorname*{argmin}_{\theta \in \Theta} \|\mathbf{A} f(\theta, \mathbf{z}) - \mathbf{y}^{\delta}\|_2^2 \tag{1.58}$$

for a random, but fixed, input $\mathbf{z}$. The intuition behind the DIP is that the convolutional architecture acts as a regularizer and is a good representation of natural images [56, 88, 205]. The DIP objective in Eqn. (1.58) defines a challenging optimization problem as it is both non-convex and over-parameterized, as the number of network weights is usually larger than the number of image pixels. Van Veel et al. [205] demonstrated that a simple over-parameterized two-layer neural network with ReLU activations can recover a reconstruction $f(\hat{\theta}, \mathbf{z})$ fitting the measurements with zero

loss, highlighting the need for regularization, else we would fit the noise in $\mathbf{y}^\delta$. Usually this regularization is implemented as an early stopping of the optimization [56, 204]. The DIP generally has two drawbacks. First, it is hard to define an early stopping rule based surerly on the noise measurements $\mathbf{y}^\delta$. Secondly, the DIP has to be retraining for each measurement $\mathbf{y}^\delta$, resulting in a high computational effort.

Instead of early stopping, a regularizer can be integrated into the DIP objective. Baguer et al. [19] propose an additional regularizer $\mathcal{S}_\mathcal{X} : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ in image space

$$\min_\theta \|\mathbf{A}f(\theta, \mathbf{z}) - \mathbf{y}^\delta\|_2^2 + \alpha \mathcal{S}_\mathcal{X}(f(\theta, \mathbf{z})), \tag{1.59}$$

which is applied to the output of the network. Van Veen et al. [205] directly include a regularizer $\mathcal{S}_\Theta : \Theta \to \mathbb{R}_{\geq 0}$ for the weights of the network

$$\min_\theta \|\mathbf{A}f(\theta, \mathbf{z}) - \mathbf{y}^\delta\|_2^2 + \alpha \mathcal{S}_\Theta(\theta), \tag{1.60}$$

where the specific form of the regularizer is learned from data in an initial training stage. To speed up the training process, Barbano et al. [22] suggest pre-training the DIP on a carefully crafted, synthetic dataset. The weights of the DIP are then initialized with this pre-training and subsequently fine-tuned using the DIP objective in Eqn. (1.58).

1.4.5 Challenges

Two common challenges of applying deep learning image reconstruction techniques in real-world applications are the lack of ground truth images and distribution shifts, i.e., the data at test time is different from the training data.

Lack of Ground Truth Most of the methods discussed in the previous section either rely on a paired dataset $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$ or on ground truth images $\{\mathbf{x}^{(i)}\}_{i=1}^N$. However, getting access to a large dataset suitable for training deep learning models is often a time-consuming task. In particular in medical imaging, we do not even have access to real ground truth images. Rather, high-quality reconstructions from low-noise measurements are used as a proxy [99, 133]. Deep learning methods for inverse problems thus have a *chicken & egg-scenario* [59] as we need to know how to solve the inverse problem to create data needed for training the network. There are a variety of extensions based on transfer learning and self-supervised learning to tackle this problem. In transfer learning, the neural network is first pre-trained on

similar, or even synthetically generated, data and only fine-tuned based on a small dataset of available measurements [20]. Both the HDC 2021 [51] and the HTC 2022 [72] were won by deep learning methods making use of simulated data for training. In the self-supervised approach, one can make use of a large dataset of measurements $\{\mathbf{y}^{(i)}\}_{i=1}^{N}$ and adapt the loss function. One such choice is to use an unsupervised loss function, i.e., we train a deep reconstructor $\mathcal{R}_\theta : \mathcal{Y} \to \mathcal{X}$ via

$$\min_\theta \sum_{i=1}^{N} \left[\frac{1}{2} \|\mathbf{A}\mathcal{R}_\theta(\mathbf{y}^{(i)}) - \mathbf{y}^{(i)}\|_2^2 + \alpha \mathcal{S}(\mathcal{R}_\theta(\mathbf{y}^{(i)})) \right], \tag{1.61}$$

where the first term enforces data consistency and the second term introduces additional regularization via a functional $\mathcal{S} : \mathcal{X} \to \mathbb{R}_{\geq 0}$. The unsupervised loss function in Eqn. (1.61) mimics the variational objective in Eqn. (1.3). As the data consistency is only enforced in the measurement space $\mathcal{Y}$, no information in the kernel of $\mathbf{A}$ can be learned [37]. Geometrical properties of the forward operator can be exploited and integrated into the network architecture to improve the self-supervised approach. Chen et al. [38] propose to learn equivariant neural networks, which forces the neural network to be equivariant with respect to rotations of the input image. In another research direction, the supervised mean squared error loss function can be approximated by an unsupervised proxy via Stein's unbiased risk estimate (SURE) [191]. The original SURE framework was proposed only for image denoising. However, there exist extensions, i.e., GSURE, for more general forward operators $\mathbf{A}$ [136]. This framework has recently been extended for the training of deep generative models from noisy measurements only [104].

Distribution Shift Supervised end-to-end trained reconstruction methods have demonstrated excellent performance in various image reconstruction tasks. For instance, both the fastMRI [112] and the LoDoPabCT [121] challenge have been won using supervised neural network approaches. However, it has been observed that neural networks struggle to keep the performance when faced with different types of perturbations or distribution shifts [11]. This issue is not exclusive to image reconstruction, but has been observed in many deep learning approaches, for example in image classification networks [162]. Distribution shifts can generally be classified into two categories: domain shifts, where the prior $\pi(\mathbf{x})$ changes, and model shifts, where the likelihood $p^{\text{lkhd}}(\mathbf{y}|\mathbf{x})$ changes. A domain shift occurs, for example, when a network trained on CT scans of knees is applied to measurements of brains, or when a network trained on CT scans from one

hospital is applied to measurements from patients of another hospital [48]. An example of a model shift would be a different scanner setup for CT, where are different set of angles is measured. Han et al. [83] proposes to tackle the problem of domain shift by fine-tuning the trained network on a small dataset from the new domain in MRI reconstruction. However, such a dataset is not always available. Darestani et al. [48] developed a test-time-adaptation approach based on a self-supervised loss function combined with an early stopping rule, using only the new out-of-distribution measurement $\mathbf{y}$. Let $\mathcal{R}_\theta : \mathcal{Y} \to \mathcal{X}$ denote the pre-trained reconstruction network, test-time-adaptation is then performed by optimizing the following loss function

$$\min_\theta \frac{1}{2}\|\mathbf{A}\mathcal{R}_\theta(\mathbf{y}) - \mathbf{y}\|_2^2. \tag{1.62}$$

This loss function encourages the reconstruction to be consistent with the measured data. Similar approaches can be used to address the issue of model shift, i.e., if the forward operator $\mathbf{A}$ changes during evaluation [73]. This is of particular importance for medical imaging tasks, as each scanner setting (number of angles, choice of angles) in CT corresponds to a different discrete forward operator. Finally, it has been shown that distribution shifts can drastically reduce the performance of deep generative models. In the context of score-based diffusion models, we developed a test-time-training approach to increase the performance on out-of-distribution data [23].

Chapter 2

Invertible Neural Networks

We can impose an inductive bias on the learning problem by carefully designing the network architecture. A good inductive bias can guide and constrain the network to produce a more realistic output. In recent years, different frameworks constraining the network to learn a function with specific properties, e.g., input convex neural networks [9], equivariant convolutional networks [46], maximal monotone networks [156] and invertible neural networks, which are discussed in this chapter, have been proposed. Restricting the class of possible neural networks also helps in developing mathematical theory. It is also a crucial part of reliable neural networks as the model will have the same basic properties independent from the training algorithm or dataset. In particular, enforcing invertibility for neural networks allows us to

- learn generative models,
- construct memory-efficient neural networks,
- and learn provable regularizers for linear inverse problems.

We start with introducing normalizing flows in Section 2.1, where we cover both finite and continuous normalizing flows. In Section 2.2, we will present different methods to construct invertible neural networks, based on a variety of different building blocks. Then, in Section 2.3, we will discuss how these invertible neural networks can be applied in inverse problems. Lastly, in Section 2.4, the link between continuous normalizing flows and score-based diffusion models will be explored.

2.1 Normalizing Flows

Density estimation is a challenging task in statistics. The data of interest is often both high dimensional and highly structured, requiring powerful models to represent this complexity. Normalizing flows are a class of probabilistic models that define rich transformation while still being trainable. They work by defining a learnable *invertible* transformation, mapping a given base density to a target density. Invertibility is a necessary component, enabling exact computation of the log-likelihood, and thus allowing for maximum likelihood training.

The concept of transforming data into white noise, known as *whitening transformations* [100], is a standard prepropressing step in statistics. A technique called *Gaussianization* has already used this idea for density estimation in the early 2000s [42]. The modern framework of *normalizing flows* was introduced by Tabak and Turner [196], who observed that a complex transformation can be built by composing simple maps. If all simple maps are invertible, the composition is also invertible. With the rise of deep learning, neural networks were proposed to parametrize these simple maps [166]. Initially, the forward and inverse passes were parametrized separately using two neural networks, and an auto-encoder loss was employed to promote invertibility[1]. The NICE framework [57] introduced the use of invertible neural networks to parametrize both the forward and inverse transformation with the same network. This research sparked further work, leading to the development of various strategies for implementing flexible invertible neural networks, see for example the review articles [113, 151].

For this introduction, we follow the presentation of Papamakarios et al. [151] and focus on continuous probability distributions that are absolutely continuous with respect to the Lebesgue measure, i.e., admit a probability density. We therefore denote both the density and the probability distribution with that density with $p(\cdot)$. For a random variable $\mathbf{z}$, we write $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ to show that $\mathbf{z}$ is distributed according to $p_{\mathbf{z}}(\mathbf{z})$ and admits this density. Further, we work with finite-dimensional densities and assume that the images $\mathbf{x} \in \mathcal{X} = \mathbb{R}^n$ are n-dimensional and the measurements $\mathbf{y} \in \mathcal{Y} = \mathbb{R}^m$ are of dimension $m \in \mathbb{N}$.

[1] Recently, this idea has received renewed interest, see [60].

2.1.1 Finite Normalizing Flows

A normalizing flow is given by a transformation $\mathcal{T}_\theta : \mathbb{R}^n \to \mathbb{R}^n$ that transforms a vector $\mathbf{z} \in \mathbb{R}^n$

$$\mathbf{x} = \mathcal{T}_\theta(\mathbf{z}), \tag{2.1}$$

where $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ is sampled from a base distribution. In contrast to the latent variable models, described in Section 1.3, both the variable $\mathbf{z}$ and $\mathbf{x}$ have the same dimension. In the context of normalizing flows, we refer to $\mathbf{z}$ as base variables and not as latent variables. A distinct characteristic of normalizing flow is that the transformation has to be a diffeomorphism, i.e., invertible, and both $\mathcal{T}_\theta$ and $\mathcal{T}_\theta^{-1}$ need to be differentiable. Under these conditions, $\mathcal{T}_\theta$ is measurable and defines a distribution $p_\theta = (T_\theta)_\# p_{\mathbf{z}}$ via the pushforward operator. The density can be calculated via change-of-variable

$$p_\theta(\mathbf{x}) = p_{\mathbf{z}}(\mathcal{T}_\theta^{-1}(\mathbf{x}))|\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x})|, \tag{2.2}$$

where $J_{\mathcal{T}_\theta^{-1}}$ denotes the Jacobian matrix of $\mathcal{T}_\theta^{-1}$. Using $\mathbf{z} = \mathcal{T}_\theta^{-1}(\mathbf{x})$ the density can also be expressed using the Jacobian of $\mathcal{T}_\theta$, i.e.,

$$p_\theta(\mathbf{x}) = p_{\mathbf{z}}(\mathbf{z})|\det J_{\mathcal{T}_\theta}(\mathbf{z})|^{-1}, \quad \mathbf{z} = \mathcal{T}_\theta^{-1}(\mathbf{x}). \tag{2.3}$$

It is important to distinguish between the computational complexities associated with sampling and calculating the likelihood. When sampling from a flow-based model, as outlined in Eqn. (2.1), the forward mapping $\mathcal{T}_\theta$ is necessary. Sampling is a two-step process: first, we sample $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ from the base distribution and secondly, this sample gets transformed via the forward mapping $\mathcal{T}_\theta$.

To construct normalizing flows, we can use the fact that diffeomorphisms are composable. If both $\mathcal{T}_1 : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathcal{T}_2 : \mathbb{R}^n \to \mathbb{R}^n$ are diffeomorphisms then $\mathcal{T} \coloneqq \mathcal{T}_2 \circ \mathcal{T}_1 : \mathbb{R}^n \to \mathbb{R}^n$ is also a diffeomorphism. The inverse and Jacobian determinant are given by

$$\mathcal{T}^{-1} = \mathcal{T}_1^{-1} \circ \mathcal{T}_2^{-1}, \quad \det J_{\mathcal{T}}(\mathbf{z}) = \det J_{\mathcal{T}_2}(\mathcal{T}_1(\mathbf{z})) \cdot \det J_{\mathcal{T}_1}(\mathbf{z}), \tag{2.4}$$

i.e., if one knows how to invert both $\mathcal{T}_1$ and $\mathcal{T}_2$, it is trivial to invert the composition. This allows the construction of complex transformations by composing several simple invertible building blocks. If each building block on its own has a tractable Jacobian determinant, the Jacobian determinant of the full normalizing flow is also tractable. This inherent characteristic serves as the foundational principle for building normalizing flows in practice. Normalizing flows constructed as a composition of a finite number of building blocks are sometimes referred to as *finite* normalizing flows [151]. Throughout

this thesis, we will refer to them as normalizing flows and introduce the term *finite* only when necessary for clarity. Further, the full probabilistic model $p_\theta(\mathbf{x})$ is sometimes referred to as a *flow-based model*.

The transformation $\mathcal{T}_\theta$ is parametrized by parameters $\theta \in \Theta$, where Θ is the space of admissible parameters. By adjusting the parameters of $\mathcal{T}_\theta$, we change the density $p_\theta(\mathbf{x})$ imposed by the flow-based model. The goal is to find the parameters θ such that the flow-based model approximates a given target distribution $p_{\text{data}}(\mathbf{x})$. To achieve this, we minimize the distance of $p_\theta(\mathbf{x})$ to $p_{\text{data}}(\mathbf{x})$ with respect to some distance in the space of probability densities. Usually, the KL divergence is employed to measure the distance. Minimization of the KL divergence recovers the negative log-likelihood loss used for fitting probabilistic models, cf. Section 1.3. The KL divergence can be written as

$$D_{\text{KL}}[p_{\text{data}}(\mathbf{x})||p_\theta(\mathbf{x})] = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[-\log p_\theta(\mathbf{x})] + \text{ const.}, \tag{2.5}$$

with a constant term not depending on the parameters θ. Using the expression of the flow-based density in Eqn. (2.2) and dropping constants that are not dependent on the parameters θ, we get the loss function

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[-\log p_{\mathbf{z}}(\mathcal{T}_\theta^{-1}(\mathbf{x})) - \log|\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x})|\right], \tag{2.6}$$

used to fit the flow-based model. The expectation over the data distribution $p_{\text{data}}(\mathbf{x})$ can be estimated with a dataset $\{\mathbf{x}^{(i)}\}_{i=1}^N$ of i.i.d. samples using Monte Carlo, i.e.,

$$\mathcal{L}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(-\log p_{\mathbf{z}}(\mathcal{T}_\theta^{-1}(\mathbf{x}^{(i)})) - \log|\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x}^{(i)})|\right). \tag{2.7}$$

In most applications, the base distribution is chosen as a standard Gaussian. In this case, the loss function can be further simplified as follows

$$\mathcal{L}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\|\mathcal{T}_\theta^{-1}(\mathbf{x}^{(i)})\|_2^2 - \log|\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x}^{(i)})|\right). \tag{2.8}$$

This loss function requires evaluation of the inverse $\mathcal{T}_\theta^{-1}$ and the Jacobian determinant of the inverse. Crucially, the forward pass is not used during training and only necessary for sampling in Eqn. (2.1).

2.1.1.1 Parametrize Forward or Inverse?

As we have seen in the previous section, there are different computational requirements for likelihood computation, necessary for training, and sampling:

1. *Likelihood computation and training:* Evaluating $\mathcal{T}_\theta^{-1}$ and the log-det Jacobian

2. *Sampling:* Evaluating $\mathcal{T}_\theta$ and sampling from the base distribution

Flow-based models employing coupling layers (see Section 2.2.2), are generally computationally symmetric, i.e., the computational costs for both the forward and inverse passes are identical. However, other architectures, for example, invertible residual networks (see Section 2.2.3) or auto-regressive networks [93], do not share this property. In these cases, it is of importance if $\mathcal{T}_\theta$ or $\mathcal{T}_\theta^{-1}$ is parametrized by an invertible neural network.

Assume that we have an invertible neural network f_θ, which is invertible in theory, but evaluating the inverse of the network is very expensive or even impossible. For these types of architectures, the inverse flow, i.e., $f_\theta = \mathcal{T}_\theta^{-1}$, can be parametrized instead. This results in the following loss function

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2} \| f_\theta(\mathbf{x}^{(i)}) \|_2^2 - \log |\det J_{f_\theta}(\mathbf{x}^{(i)})| \right), \tag{2.9}$$

for a Gaussian base distribution. Invertible models, without an analytical inverse or an expensive computation of the inverse, can still be used for efficient likelihood estimation. This kind of model can be used for out-of-distribution detection [163] or to approximate posteriors in VAEs [151].

2.1.1.2 Sampling Error

We are often interested in estimating specific moments of the target distribution $p_{\text{data}}(\mathbf{x})$. In particular, in statistical inverse problems, the conditional mean or pointwise conditional variance is of interest. We can apply the trained flow-based model $p_\theta = (\mathcal{T}_\theta)_\# p_{\mathbf{z}}$ as a surrogate to estimate specific moments $g(\mathbf{x})$

$$\begin{aligned} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[g(\mathbf{x})] &\overset{(1)}{\approx} \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[g(\mathbf{x})] \\ &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[g(\mathcal{T}_\theta(\mathbf{z}))] \overset{(2)}{\approx} \frac{1}{N} \sum_{i=1}^{N} g(\mathcal{T}_\theta(\mathbf{z}^{(i)})) \quad \mathbf{z}^{(i)} \sim p_{\mathbf{z}}, \end{aligned} \tag{2.10}$$

where we take advantage of the fact that sampling from the base distribution $p_{\mathbf{z}}(\mathbf{z})$ is usually cheap. The error in the Monte Carlo approximation (2) depends on the number of samples N and can thus be reduced by drawing more samples from the base distribution and

transforming them with the normalizing flow. The approximation (1) can be bounded using the KL divergence, i.e.,

$$\|\mathbb{E}_{p_{\text{data}}}[g(\mathbf{x})] - \mathbb{E}_{p_\theta}[g(\mathbf{x})]\| \leq C(p_{\text{data}}, p_\theta)\sqrt{D_{\text{KL}}(p_{\text{data}}||(\mathcal{T}_\theta)_{\#}p_{\mathbf{z}})}, \tag{2.11}$$

with $C(p_{\text{data}}, p_\theta) = \sqrt{2(\|\mathbb{E}_{p_{\text{data}}}[g(\mathbf{x})]\|^2 + \|\mathbb{E}_{p_\theta}[g(\mathbf{x})]\|^2)}$ [132]. This inequality tells us, that the approximation error is bounded by the KL divergence, which is exactly the quantity that is minimized during training.

2.1.1.3 Instability

Behrmann et al. [26] showed that stability issues can arise in flow-based models when applied to real-world datasets. Typically, these models employ a standard Gaussian as the base distribution. When dealing with multi-modal distributions that exhibit low-density regions between modes, a high Lipschitz constant is necessary to map the standard Gaussian to separate modes in the target distribution. As an example, Hagemann et al. [82] showed that for a standard Gaussian base distribution and a target $\mathbf{x} \sim 1/2\mathcal{N}(1, \sigma^2) + 1/2\mathcal{N}(-1, \sigma^2)$ the Lipschitz constant $\text{Lip}(\mathcal{T}_\theta)$ explodes[2] as $\sigma^2 \to 0$, i.e., as the density between the modes decreases. The authors further show that using a multi-modal base distribution can help alleviate these stability issues. As during training only the inverse $\mathcal{T}_\theta^{-1}$ is needed, the exploding Lipschitz constant for $\mathcal{T}_\theta$ can go unnoticed.

2.1.2 Conditional Normalizing Flows

Normalizing flows allow us to model densities as transformations of a simple base density $p_{\mathbf{z}}(\mathbf{z})$. In inverse problems, we are often interested in modeling the posterior $p^{\text{post}}(\mathbf{x}|\mathbf{y})$, i.e., the conditional density of the image $\mathbf{x}$ given measurements $\mathbf{y}$. Modeling conditional densities can be achieved with *conditional* normalizing flows [12, 214]. A conditional normalizing flow is defined by a transformation $\mathcal{T}_\theta : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ that takes $\mathbf{y} \in \mathbb{R}^m$ as an additional input to transform a base variable $\mathbf{z}$

$$\mathbf{x} = \mathcal{T}_\theta(\mathbf{z}, \mathbf{y}), \tag{2.12}$$

where $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ is sampled from a pre-specified base distribution. If, for every fixed $\mathbf{y} \in \mathbb{R}^m$, the forward mapping $T_\theta(\cdot, \mathbf{y}) : \mathbb{R}^n \to \mathbb{R}^n$ is a diffeomorphism, we can define the pushforward $p_\theta^{\mathbf{x}|\mathbf{y}} = (\mathcal{T}_\theta(\cdot, \mathbf{y}))_{\#}p_{\mathbf{z}}$ and calculate the density

[2] The term *exploding inverse* is used by Behrmann et al. [26] to refer to the unstable inverse due to a high Lipschitz constant, see also Figure 1 in [26].

via change-of-variables

$$p_\theta^{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y}) = p_{\mathbf{z}}(\mathcal{T}_\theta^{-1}(\mathbf{x},\mathbf{y}))|\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x};\mathbf{y})|, \tag{2.13}$$

where $J_{\mathcal{T}_\theta^{-1}}$ denotes the Jacobian matrix of $\mathcal{T}_\theta^{-1}(\cdot,\mathbf{y})$ given a fixed input $\mathbf{y}$. Put differently, the transformation $\mathcal{T}_\theta(\cdot,\mathbf{y})$ defines a family of normalizing flows.

The process of training a conditional normalizing flow is similar to the training of normalizing flows. The goal is to fit the flow parameters θ such that the conditional density $p_\theta^{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y})$ defined by the conditional flow-based model approaches a given posterior $p^{\text{post}}(\mathbf{x}|\mathbf{y})$. Here, we can minimize the expected KL divergence over measurements $\mathbf{y} \sim p(\mathbf{y})$. This results in the loss function

$$\begin{aligned}
\operatorname*{argmin}_{\theta\in\Theta} \mathcal{L}_{\text{CNF}}(\theta) &= \operatorname*{argmin}_{\theta\in\Theta} \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y})}[D_{\text{KL}}[p^{\text{post}}(\mathbf{x}|\mathbf{y})||p_\theta^{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y})]] \\
&= \operatorname*{argmin}_{\theta\in\Theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p(\mathbf{x},\mathbf{y})}[-\log p_\theta^{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y})] \\
&= \operatorname*{argmin}_{\theta\in\Theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p(\mathbf{x},\mathbf{y})}\left[-\log p_{\mathbf{z}}(\mathcal{T}_\theta^{-1}(\mathbf{x},\mathbf{y})) - \log|\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x};\mathbf{y})|\right],
\end{aligned} \tag{2.14}$$

where we used the change-of-variable formula and dropped all constants independent of θ in the last step and set $p(\mathbf{x},\mathbf{y}) = p^{\text{post}}(\mathbf{x}|\mathbf{y})p_{\mathbf{y}}(\mathbf{y})$.

The conditional transformation $\mathcal{T}_\theta$ is modeled as a neural network with two inputs. In many medical imaging applications, the structure of the measurements $\mathbf{y}$ can be highly complicated. For example, in computed tomography, the measurements represent line integrals taken from various angles. Constructing a neural network to directly handle such inputs is very challenging and highly problem depending. To address this problem and simplify the network construction, the measurements are often preprocessed using the adjoint $\mathbf{A}^* : \mathbb{R}^m \to \mathbb{R}^n$ or a similar method to project the measurements into the image domain [52, 53] or [150]. For linear inverse problems with additive Gaussian noise, it can be shown that the posterior $p^{\text{post}}(\mathbf{x}|\mathbf{y}) = p^{\text{post}}(\mathbf{x}|\mathbf{A}^*\mathbf{y})$ remains the same if adjoint data $\mathbf{A}^*\mathbf{y}$ is used as an input (see [6, Proposition 1] or [150]). Furthermore, this adjoint preprocessing provides a physics-informed way of obtaining standardized data. The dimensionality of computed tomography measurements depends on the number of measured angles and the number of detector pixels, which means that the conditional flow-based model must be able to work with data of different shapes. On the other hand, the adjoint data $\mathbf{A}^*\mathbf{y}$ always has the same dimensionality, regardless of the measurement setup. In particular, this allows implementations based on convolutional layers.

2.1.3 Continuous Normalizing Flows

Finite normalizing flows are constructed as a composition of a fixed number of simple transformations. To achieve a high level of expressiveness, a large number of simple transformations is needed, resulting in very deep neural networks. Instead, Chen et al. [40] observed that the normalizing flow can be modeled in continuous time by defining the transformation as the solution of an ordinary differential equation (ODE)

$$\frac{d\mathbf{z}(t)}{dt} = f_\theta(\mathbf{z}(t), t) \quad t \in [0, T], \tag{2.15}$$

where $f_\theta : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is implemented as a time-dependent neural network. ODEs of this type, where the dynamics are governed by a neural network, are referred to as *neural ODEs* and are studied in many areas of physical, financial, or time series modeling (see the overview [105]). Similar to finite normalizing flow, we define a density for each intermediate $\mathbf{z}(t)$ by imposing a base distribution for $\mathbf{z}(0) \sim p_\mathbf{z}(\mathbf{z}(0))$ at $t = 0$. All intermediate densities can be evaluated using a continuous variation of change-of-variables. We denote the time-dependent density with $p_\theta : [0, T] \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ and the likelihood is given by

$$\frac{d \log p_\theta(t, \mathbf{z}(t))}{dt} = -\mathrm{Tr}\left[J_{f_\theta(\cdot,t)}(\mathbf{z}(t))\right], \tag{2.16}$$

which is equivalent to the Fokker-Planck equation with a deterministic dynamic and random initial conditions [77, 105]. For continuous change-of-variables, we require that f_θ is continuous in t and uniformly Lipschitz continuous in $\mathbf{z}$ [151]. Importantly, no invertibility is required. However, continuous normalizing flows often have a higher computational cost compared to their finite counterparts. Instead of one pass through the network, the dynamical system in Eqn. (2.16) has to be solved from $t = 0$ to $t = T$ to draw one sample. Further, evaluating the trace of the Jacobian is computationally expensive, in particular in high-dimensional settings. To alleviate the computational cost, either the Hutchinson trace estimator [94] (cf. Section 2.2.3) or special neural network architectures [39] are employed.

Continuous normalizing flows can be trained using the maximum likelihood objective. This requires the evaluation of the log-likelihood of a data sample $\mathbf{x}$ at $t = T$. Using Eqn. (2.16), the log-likelihood is given by

$$\log p_\theta(T, \mathbf{x}) = \log p_\theta(0, \mathbf{z}(0)) - \int_0^T \mathrm{Tr}\left[J_{f_\theta(\cdot,t)}(\mathbf{z}(t))\right] dt, \tag{2.17}$$

with $\log p_\theta(0, \mathbf{z}(0)) = \log p_z(\mathbf{z}(0))$ as the base distribution and $\mathbf{z}(0)$ as the

solution to the initial value problem

$$\frac{d\mathbf{z}(t)}{dt} = f_\theta(\mathbf{z}(t), t) \quad \mathbf{z}(T) = \mathbf{x}, \ t \in [T, 0], \tag{2.18}$$

running backwards in time from $t = T$ to $t = 0$. This initial value problem and the evaluation of the integral on the RHS in Eqn. (2.17) can be written compactly as one system of ODEs. Evaluating the likelihood thus requires one call to an ODE solver. Training the network with the maximum likelihood objective demands to backpropagate through the ODE to collect all necessary gradients. There are generally three ways to accomplish this [105].

Full discretization The initial value problem is fully discretized and the gradients are computed with respect to the discretized ODE, i.e., using Euler's method

$$\mathbf{z}^{(t-\epsilon)} = \mathbf{z}^{(t)} - \epsilon f_\theta(\mathbf{z}^{(t)}, t), \tag{2.19}$$

with a small step size $\epsilon > 0$. For a step size $\epsilon < 1/L$ with L being the Lipschitz constant of $f_\theta(\cdot, t)$, this resembles the architecture used in invertible residual networks (see Section 2.2.3) [151]. For the gradient computation, we can make use of automatic differentiation and the backpropagation algorithm. However, this full discretization approach is typically memory intensive as each intermediate $\mathbf{z}^{(t-\epsilon)}$, and all network activations, have to be kept in memory to compute the gradients. The memory cost can be alleviated by using time reversible ODE solver, e.g., the reversible Heun method or the asynchronous leapfrog method [105].

Adjoint ODE Chen et al. [40] define an adjoint ODE, which has to be solved to recover the gradients. In optimal control theory this is known as the adjoint sensitivity method [157]. This approach has a higher computational cost as two ODEs, i.e., Eqn. (2.18) from $t = T$ to $t = 0$ and the adjoint ODE from $t = 0$ to $t = T$, have to be solved for each optimization step. However, the memory cost is lower than for the full discretization approach.

Flow Matching Recently, Lipman et al. [124] proposed a training objective called *flow matching* inspired by the denoising score matching used for score-based diffusion models [189]. Flow matching is a simulation-free framework and does not require any calls to an ODE solver during training. This is a promising new direction, which allows to train continuous normalizing flows at a larger scale.

	Inverse	**Computing the log-det Jacobian**
Linear Flow		
Inv. Matrix	matrix inversion, $\mathcal{O}(n^3)$	full determinant, $\mathcal{O}(n^3)$
QR Flow	fwd/bwd substitution, $\mathcal{O}(n^2)$	sum over diagonal
PLU Flow	fwd/bwd substitution, $\mathcal{O}(n^2)$	sum over diagonal
Permutation	transpose	constant
Residual Flow		
Contractive	fixed-point iteration	stochastic approximation
Matrix-Det-Lemma	no analytical formulation	$\mathcal{O}(M^3 + nM^2)$[1]
Coupling Flow		
Additive	analytical inverse	constant
Affine	analytical inverse	sum over diagonal
Other Blocks		
ActNorm	analytical inverse	sum over channels
Down/Upsampling	analytical inverse	constant

[1] $M \in \mathbb{N}, M < n$ is the bottleneck dimension of the layer

Table 2.1: Comparison of the different invertible components discussed in Section 2.2 with respect to the computation of the inverse pass and the log-det of the Jacobian. The dimensionality of the data is $n \in \mathbb{N}$.

2.2 Construction of Invertible Networks

Normalizing flows require invertible neural networks to facilitate likelihood computation and training. As invertible transformations are composable, we can build an expressive invertible network by stacking invertible layers on top of each other. The full inverse can then be computed by inverting one layer at a time. In this section, we will discuss several invertible components used for normalizing flows. An overview of invertible components discussed in this section is provided in Table 2.1. The focus of this section lies on architectures used in imaging applications, in particular for imaging inverse problems. As a result, autoregressive flows [110] are not covered. Here, we refer to the review [151].

2.2.1 Linear Flow Layers

Many invertible networks employ invertible *linear* layers in the architecture, e.g., for a generalization of a permutation [107], as a scaling layer [57] or to build orthogonal convolutions [67]. A linear flow is defined by a transformation

$$\mathbf{z}^{\text{out}} = \mathbf{W}\mathbf{z}^{\text{in}}, \tag{2.20}$$

with an invertible matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ and $\mathbf{z}^{\text{in}}, \mathbf{z}^{\text{out}} \in \mathbb{R}^n$. The determinant of the Jacobian is given by the determinant of the matrix, i.e., $\det J_{\mathcal{T}}(\mathbf{z}^{\text{in}}) = \det(\mathbf{W})$. However, learning an invertible matrix is a hard problem, as no continuous surjective parametrization of invertible matrices exists [151]. Further, in general, inverting a linear layer is expensive as the linear system $\mathbf{z}^{\text{out}} = \mathbf{W}\mathbf{z}^{\text{in}}$ has to be solved for each inverse pass. To address these challenges, most authors resort to *structured* invertible matrices, which allow for efficient invertibility and a simple calculation of the determinant. As a first example, triangular matrices with non-negative diagonal entries are employed as scaling layers [57]. Triangular matrices have the advantage of a trivial determinant and the inverse has a similar computational complexity (using forward/backward substitution) as matrix multiplication. Other approaches are based on matrix factorization [151], i.e., using a PLU [148] or QR decomposition [92]. For the PLU approach, we set $\mathbf{W} = \mathbf{PLU}$ with a permutation matrix $\mathbf{P}$ and a lower and upper triangular matrix $\mathbf{L}$ and $\mathbf{U}$, respectively. Generally, only $\mathbf{L}$ and $\mathbf{U}$ are learned, while the permutation matrix $\mathbf{P}$ is drawn randomly and then fixed [148].

The QR flow requires a parametrization of orthogonal matrices. Similar to invertible matrices, there is no general representation. An orthogonal matrix either has a determinant of 1 or -1, i.e., there exist two separate islands of orthogonal matrices. Common parametrizations include the exponential, $\mathbf{Q} = \exp(\mathbf{B})$, or Caley map, $\mathbf{Q} = (\mathbf{I}+\mathbf{B})(\mathbf{I}-\mathbf{B})^{-1}$, for a skew-symmetric matrix $\mathbf{B}$ [74]. A skew-symmetric matrix can be parametrized as $\mathbf{B} = \tilde{\mathbf{B}} - \tilde{\mathbf{B}}^T$ for any $\tilde{\mathbf{B}} \in \mathbb{R}^{n\times n}$. However, both transformations have a computational complexity of $\mathcal{O}(n^3)$ and thus do not scale to high dimensions. To alleviate this problem, Tomczak et al. [202] use the householder transformation. The matrix $\mathbf{Q}$ is given by a product of $K \in \mathbb{N}$ householder matrices

$$\mathbf{Q} = \prod_{k=1}^{K} \mathbf{H}^{(k)}, \quad \mathbf{H}^{(k)} = \mathbf{I} - 2\frac{\mathbf{v}^{(k)}(\mathbf{v}^{(k)})^T}{\|\mathbf{v}^{(k)}\|_2^2}, \tag{2.21}$$

where $\mathbf{v}^{(k)}$ is a vector with non-zero entries. Note that this decomposition is not unique, e.g., any permutation of the vectors $\mathbf{v}^{(k)}$ results in the same matrix $\mathbf{Q}$. This can result in difficulties during optimization [151]. The application of orthogonal matrices in flows is of interest due to the simple inverse, i.e., $\mathbf{Q}^{-1} = \mathbf{Q}^T$, and the trivial determinant. In particular, orthogonal layers are often used as learnable generalizations of permutations.

Application to Images Matrix factorization approaches are generally not efficient when dealing with image data, i.e., in cases where $\mathbf{W}$ is a convolution matrix. While the forward pass is fast, calculating the inverse or determinant requires a high computational effort. The *Glow* framework [107] introduces invertible 1×1 convolutions, which reduce to linear transformations applied across channels, see also Eqn. (1.30). For an input image of size $h \times w$ with c channels, the matrix $\mathbf{W}$ is only of size $c \times c$ and does not depend on the spatial size of the image. As the number of channels is usually smaller than the spatial dimension $h \times w$, any of the matrix decomposition techniques can be employed to parameterize $\mathbf{W}$. This approach has further been extended for general convolutional layers in a framework called *emerging convolutions* [92].

2.2.2 Coupling Layers

Coupling layers are analytically invertible and computationally symmetric, i.e., evaluation of the forward or inverse pass share the same complexity [57, 58]. This property is particularly useful for the design of normalizing flows, as we do not have to balance sampling and training speed. The invertibility is enforced by splitting the input into two parts, commonly the input is split evenly. The first part is left unchanged, while the second part is transformed based on the first part. Let $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2] \in \mathbb{R}^n$ be the input into a coupling layer, such that $\mathbf{z}$ is split into $\mathbf{z}_1 \in \mathbb{R}^d$ and $\mathbf{z}_2 \in \mathbb{R}^{n-d}$. The output of a coupling layer is given by

$$\begin{aligned} \mathbf{z}_1^{\text{out}} &= \mathbf{z}_1^{\text{in}} \\ \mathbf{z}_2^{\text{out}} &= G(\mathbf{z}_2^{\text{in}}, m(\mathbf{z}_1^{\text{in}})), \end{aligned} \tag{2.22}$$

where $G : \mathbb{R}^{n-d} \times \mathbb{V} \to \mathbb{R}^{n-d}$ is an invertible coupling law with respect to the first argument. The transformation $m : \mathbb{R}^d \to \mathbb{V}$ has no constraints. In particular, it does not need to be invertible and can be implemented as an arbitrary neural network. Two main types of coupling laws have been studied in the literature: additive couplings [57] and affine couplings [58]. For an additive coupling layer, we have $\mathbb{V} = \mathbb{R}^{n-d}$ and the forward and inverse

pass are given by

$$\begin{aligned} \mathbf{z}_1^{\text{out}} &= \mathbf{z}_1^{\text{in}} \\ \mathbf{z}_2^{\text{out}} &= \mathbf{z}_2^{\text{in}} + m(\mathbf{z}_1^{\text{in}}) \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} \mathbf{z}_1^{\text{in}} &= \mathbf{z}_1^{\text{out}} \\ \mathbf{z}_2^{\text{in}} &= \mathbf{z}_2^{\text{out}} - m(\mathbf{z}_1^{\text{out}}). \end{aligned} \tag{2.23}$$

Additive coupling layers are volume preserving, i.e., have a unit Jacobian determinant. This hinders the expressivity of additive coupling layers and they are often used in combination with scaling layers, i.e., an diagonal matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ with non-zero diagonal entries [57]. A more expressive coupling is given by the affine coupling law. Here, we set the codomain as $\mathbb{V} = \mathbb{R}^{n-d} \times \mathbb{R}^{n-d}$ with $m(\mathbf{z}) = [s(\mathbf{z}), t(\mathbf{z})]$, where $s(\cdot)$ represents a scaling and $t(\cdot)$ a translation. The forward pass is given by

$$\begin{aligned} \mathbf{z}_1^{\text{out}} &= \mathbf{z}_1^{\text{in}} \\ \mathbf{z}_2^{\text{out}} &= \mathbf{z}_2^{\text{in}} \odot \exp\big(s(\mathbf{z}_1^{\text{in}})\big) + t(\mathbf{z}_1^{\text{in}}), \end{aligned} \tag{2.24}$$

with an analytical inverse pass

$$\begin{aligned} \mathbf{z}_1^{\text{in}} &= \mathbf{z}_1^{\text{out}} \\ \mathbf{z}_2^{\text{in}} &= \exp(-s(\mathbf{z}_1^{\text{out}})) \odot (\mathbf{z}_2^{\text{out}} - t(\mathbf{z}_1^{\text{out}})). \end{aligned} \tag{2.25}$$

Let $\mathbf{z}^{\text{out}} = \mathcal{T}(\mathbf{z}^{\text{in}})$ denote the forward pass of the coupling layer. The Jacobian

$$J_{\mathcal{T}}(\mathbf{z}^{\text{in}}) = \begin{pmatrix} \mathbf{I}_d & \mathbf{0}_{n-d} \\ \frac{d\mathbf{z}_2^{\text{out}}}{d\mathbf{z}_1^{\text{in}}} & \frac{d\mathbf{z}_2^{\text{out}}}{d\mathbf{z}_2^{\text{in}}} \end{pmatrix} \tag{2.26}$$

is a block diagonal matrix and the Jacobian determinant can be computed by only considering the lower right block, i.e.,

$$\det J_{\mathcal{T}}(\mathbf{z}^{\text{in}}) = \det \frac{d\mathbf{z}_2^{\text{out}}}{d\mathbf{z}_2^{\text{in}}} := \det J_{\mathbf{z}_2}. \tag{2.27}$$

Both the additive and affine coupling law are designed such that the Jacobian $J_{\mathcal{T}}(\mathbf{z}^{\text{in}})$ is a lower triangular matrix. In particular, for the additive coupling law $J_{\mathbf{z}_2}$ is the identity and thus offers an efficient determinant. For the affine coupling law, the Jacobian determinant can be computed as

$$\det \frac{d\mathbf{z}_2^{\text{out}}}{d\mathbf{z}_2^{\text{in}}} = \det \operatorname{diag}\left(\exp\big(s(\mathbf{z}_1^{\text{in}})\big)\right) = \prod_{i=1}^{n-d} \exp\big(s(\mathbf{z}_1^{\text{in}})_i\big), \tag{2.28}$$

which does not require any additional computation as $s(\mathbf{z}_1^{\text{in}})$ is already computed in the forward pass of the coupling layer. Both coupling laws result in a very sparse Jacobian with only a dense lower left block.

The first input $\mathbf{z}_1^{\text{in}}$ to a coupling layer is always unchanged. This requires that after each coupling layer some permutation of the entries in $\mathbf{z}^{\text{out}}$ has to be performed [57]. Further, due to the sparsity of the Jacobian coupling-based

models require a large number of blocks, for example *Glow* has 96 coupling blocks [107], to build expressive transformations, which slows down training. The *HINT* [115] framework defines a recursive coupling block by repeatedly splitting the subsets and defining sub-couplings for each split, to enable expressive transformations with a smaller number of coupling blocks.

2.2.2.1 Application to Images

Many successful normalizing flow architectures for images are based on coupling layers, see [12, 58, 107] or [52]. In imaging applications the input to the coupling layer $\mathbf{z}^{\text{in}}$ is an image with spatial dimensions $h \times w$ and c channels. The *RealNVP* framework [58], which popularized the use of coupling layers for normalizing flows, proposed two types of splitting the input image. For the first type, the input $\mathbf{z}^{\text{in}}$ is split across the channels to produce $\mathbf{z}_1^{\text{in}}$ and $\mathbf{z}_2^{\text{in}}$ with c_1 and c_2 (with $c_1 + c_2 = c$) channels, respectively. The second splitting type is based on a checkerboard pattern, similar to the checkerboard downsampling to be discussed in Section 2.2.4. However, this second splitting was not used in later architectures, see for example Glow [107]. Further, the transformation $m(\cdot)$, i.e., $s(\cdot)$ and $t(\cdot)$ for affine coupling, are implemented as CNNs. Figure 2.1 is a schematic description of the additive coupling layer with channel splitting.

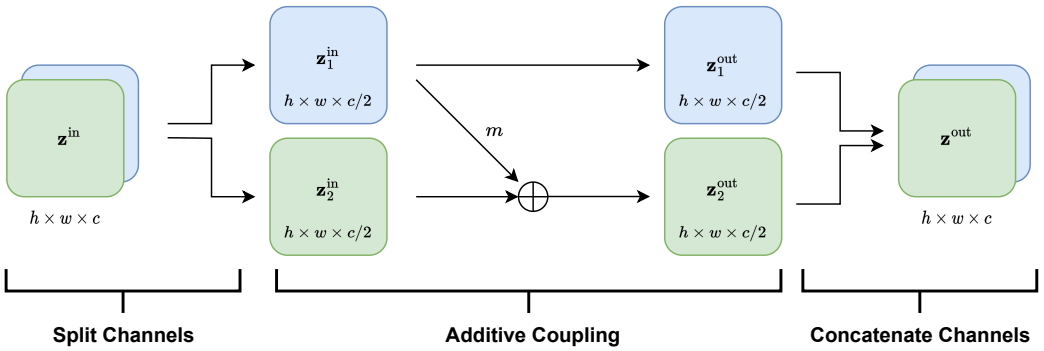

Figure 2.1: Schematic description of the additive coupling block for images. Here, we assume that the number of channels is even. In the first step, the input image $\mathbf{z}^{\text{in}}$ is split across channels into $\mathbf{z}_1^{\text{in}}$ and $\mathbf{z}_2^{\text{in}}$. This serves as the input to the additive coupling block. The output is then again concatenate on the channels.

2.2.2.2 Conditional Coupling Layers

To introduce conditional variables $\mathbf{y} \in \mathbb{R}^m$ to the coupling layer, we can extend the sub-network $m : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{V}$ with an additional input. A conditional coupling block is then defined by

$$\begin{aligned} \mathbf{z}_1^{\text{out}} &= \mathbf{z}_1^{\text{in}} \\ \mathbf{z}_2^{\text{out}} &= G(\mathbf{z}_2^{\text{in}}, m(\mathbf{z}_1^{\text{in}}, \mathbf{y})), \end{aligned} \tag{2.29}$$

with the same properties as the unconditional coupling layer in Eqn. (2.22), i.e., analytical inverse and block diagonal Jacobian. As discussed in Section 2.1.2, in many applications the measurements $\mathbf{y}$ are not directly used as an input to the flow model, but rather some preprocessed inputs $h(\mathbf{y})$, for example, adjoint data with $h(\mathbf{y}) = \mathbf{A}^*\mathbf{y}$ [150].

These conditional coupling layers resemble Feistel networks used in cryptography [135]. In this context $\mathbf{z}$ is the text to encrypt and $\mathbf{y}$ is the key. Feistel networks have an important advantage that they are trivially invertible, i.e. the key $\mathbf{y}$ is known, but the forward pass, i.e. the encryption, can be made as complicated as possible, as the transformation m does not have to be invertible.

2.2.3 Invertible Residual Layers

Another way of enforcing invertibility is by combining residual layers [87] with Lipschitz constraints to create *invertible residual layers* [25, 41]. These layers are of the form

$$\mathbf{z}^{\text{out}} = G(\mathbf{z}^{\text{in}}) = \mathbf{z}^{\text{in}} + f_\theta(\mathbf{z}^{\text{in}}), \tag{2.30}$$

with $\text{Lip}(f_\theta) = L < 1$. Invertible residual layers do not have to rely on dimension splitting, thus all dimensions can influence each other. Further, by the Lipschitz constraint, we get stability estimates for both the forward and inverse pass

$$\text{Lip}(G) \leq 1 + L, \quad \text{Lip}(G^{-1}) \leq \frac{1}{1-L}. \tag{2.31}$$

However, instead of having an explicit inverse, the inverse has to be computed via a fixed-point iteration for each inverse pass. We can recover $\mathbf{z}^{\text{in}}$ as the limit of

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{\text{out}} - f_\theta(\mathbf{z}^{(k)}) \quad k = 0, 1, 2, \ldots, \tag{2.32}$$

which converges for every $\mathbf{z}^{(0)} \in \mathbb{R}^n$. Behrmann et al. [25] propose to initialize the fixed point iteration with $\mathbf{z}^{(0)} = \mathbf{z}^{\text{out}}$.

2.2.3.1 Enforcing the Lipschitz constraint

To ensure invertibility, the residual function f_θ has to satisfy the Lipschitz constraint. The task of constraining the Lipschitz constant of a neural network has been explored in the broader domain of deep learning. In CLIP [34] an additional penalty term is added to the loss function during training, penalizing high Lipschitz constants. Similar penalties are also employed in the adversarial regularizer [128] or in Wasserstein GANs [79] to learn 1-Lipschitz functions. However, these penalty approaches do not guarantee that the Lipschitz constraint is fulfilled. Behrmann et al. [25] use simple feed-forward networks, i.e., $f_\theta(\mathbf{x}) = \mathbf{W}_3\phi_2(\mathbf{W}_2\phi_1(\mathbf{W}_1\mathbf{x}))$, with contractive non-linearities ϕ_1, ϕ_2 to parametrize the residual network. For networks structured in this manner, the Lipschitz constant can be estimated as the product of the weight matrices

$$\mathrm{Lip}(f_\theta) \leq \|\mathbf{W}_3\|_2\|\mathbf{W}_2\|_2\|\mathbf{W}_1\|_2. \tag{2.33}$$

By enforcing $\|\mathbf{W}_i\|_2 < 1$ for each layer, one can guarantee invertibility. Behrmann et al. [25] estimate the norm after each training step and renormalize the weights if necessary. Arndt et al. [14] extend this approach to directly parameterize the weight matrices so that they consistently satisfy this constraint.

In imaging applications, the residual network f_θ is usually implemented as a shallow CNN [25, 41]. The Lipschitz constraint is enforced for each layer by estimating the spectral norm using a power-iteration [138].

2.2.3.2 Jacobian determinant

There is no known efficient algorithm to compute the exact Jacobian determinant of an invertible residual layer. Explicitly constructing the full Jacobian, using automatic differentiation, and then computing the determinant is computationally infeasible for high dimensional data. Behrmann et al. [25] write the Jacobian determinant as a power series

$$\log|\det J_G(\mathbf{z})| = \log|\det(\mathbf{I} + J_{f_\theta}(\mathbf{z}))| = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \mathrm{Tr}\left(J_{f_\theta}^k(\mathbf{z})\right), \tag{2.34}$$

where the trace of the Jacobian can be estimated using the Hutchinson trace estimator [94]

$$\mathrm{Tr}\left(J_{f_\theta}^k(\mathbf{z})\right) \approx \mathbf{v}^T J_{f_\theta}^k(\mathbf{z})\mathbf{v}, \tag{2.35}$$

where $\mathbf{v}$ is a random vector with zero mean and unit covariance, typically chosen as a standard Gaussian or Rademacher distribution. The power series is approximated by only evaluating terms up to $K \in \mathbb{N}$ to obtain the estimate

$$\log|\det J_G(\mathbf{z})| \approx \sum_{k=1}^{K} \frac{(-1)^{k+1}}{k} \mathbf{v}_k^T J_{f_\theta}^k(\mathbf{z}) \mathbf{v}_k, \quad \mathbf{v}_k \sim p_{\mathbf{v}}(\mathbf{v}), \tag{2.36}$$

where the Jacobian-vector product can be computed using automatic differentiation functionalities, without the need to construct the full Jacobian [153]. The fixed truncation of the power series results in a biased estimator, which can hinder maximum likelihood training. By using a so-called Russian roulette estimator, i.e., letting the truncation be random and re-weighting the terms, the estimator can be made unbiased [41]. Similar techniques can be used to directly approximate the gradient with respect to the parameters θ of the Jacobian determinant [41].

2.2.3.3 Matrix Determinant Lemma

Another way of reducing the computational complexity of the Jacobian determinant is through the use of the matrix determinant lemma [151]. The matrix determinant lemma

$$\det\left(\mathbf{I}_n + \mathbf{V}\mathbf{W}^T\right) = \det\left(\mathbf{I}_M + \mathbf{W}^T\mathbf{V}\right), \tag{2.37}$$

with $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{n \times M}$ and $M << n$ [151], only requires to compute the determinant of a smaller $M \times M$ matrix. Thus, for transformations of the form

$$\mathbf{z}^{\text{out}} = \mathbf{z}^{\text{in}} + \mathbf{V}\sigma(\mathbf{W}^T\mathbf{z}^{\text{in}} + \mathbf{b}), \tag{2.38}$$

the computational complexity for evaluating the Jacobian determinant can be greatly reduced if M is chosen small. A variety of methods, e.g., *planar flows* [164], *Sylvester flows* [30] and *radial flows* [196], rely on this property. To ensure invertibility, techniques from linear flow layers (see Section 2.2.1) can be employed. Note, that while these layers are invertible, the inverse cannot be given analytically. Flow models based on the matrix determinant lemma commonly parameterize the inverse $\mathcal{T}_\theta^{-1}$, allowing for a fast training and likelihood computation.

2.2.3.4 Conditional Residual Layers

We can implement conditional residual layers by introducing an additional input $\mathbf{y}$ to the residual function, i.e.

$$\mathbf{z}^{\text{out}} = G(\mathbf{z}^{\text{in}}) = \mathbf{z}^{\text{in}} + f_\theta(\mathbf{z}^{\text{in}}, \mathbf{y}). \tag{2.39}$$

To ensure invertibility, the Lipschitz constant of the residual layer has to be less than one for all possible conditional inputs, i.e., $\text{Lip}(f_\theta(\cdot, \mathbf{y})) < 1$ for all $\mathbf{y} \in \mathbb{R}^m$.

2.2.4 Invertible Up- and Downsampling

Down- and upsampling operations are an important part of CNNs and are extensively used in many architectures, for example, the U-Net [168]. Thus, one wants to adopt these concepts for invertible architectures. However, by itself down- and upsampling is not invertible. However, there are invertible extensions. All of them have in common, that the total number of dimensions, i.e., the product of spatial dimensions and channels, has to be kept constant. In most practical applications, the spatial dimension is reduced by a factor of 2, and the channel dimension is increased by a factor of 4 for invertible downsampling operations. There are three main methods, which are currently in use by invertible architectures. Dinh et al. [58] propose a fixed pixel shuffle in a checkerboard pattern. However, this can result in artifacts in the final image, as values from different channels are mixed. Ardizzone et al. [12] propose to use an orthogonal convolution, based on haar-wavelets. This idea is generalized by Etmann et al. [67] to learnable down- and upsampling operations. Here, they make use of the Caley map to parametrize orthogonal matrices, as discussed in Section 2.2.1 for linear flow layers. As the resulting convolution is orthogonal, the inverse is given by the transposed convolution and the log-determinant of the Jacobian is constant.

2.2.5 Normalization

Normalization is often used to accelerate and stabilize the training of deep neural networks [75]. Most normalization techniques, like batch normalization [96], are trivial to invert. Batch normalization can be expressed as a linear layer

$$\mathbf{z}_c^{\text{out}} = \frac{\mathbf{z}_c^{\text{in}} - \tilde{\mu}_c}{\sqrt{\tilde{\sigma}_c^2 + \epsilon}}, \quad c = 1, \dots, C, \tag{2.40}$$

where the batch statistics, i.e., the mean $\tilde{\mu}_c$ and variance $\tilde{\sigma}_c^2$, are calculated for each channel c and $\epsilon > 0$ is a small numerical constant. The Jacobian determinant is given by

$$\prod_{c=1}^{C} (\tilde{\sigma}_c^2 + \epsilon)^{-1/2}. \tag{2.41}$$

This form of Batch normalization was successfully employed in the RealNVP framework [58]. However, due to memory constraints normalizing flows are often trained with small batches, which can lead to an inaccurate estimate of the batch statistics. To address this problem *ActNorm* [107] was proposed. Instead of tracking the batch statistics $\tilde{\mu}_c$ and $\tilde{\sigma}_c^2$ during training, they are initialized using the first mini-batch and then treated as regular trainable parameters. By integrating ActNorm layers into the architecture, the intermediate outputs have a mean of zero and a standard deviation of one at the start of training.

2.2.6 Multi-scale Architecture

Multi-scale image representation is a powerful concept in image processing that involves representing the image at different scales or levels of detail. It is widely used in image pyramids [2], scale-space theory [123], or wavelet processing [126]. This type of multi-scale processing is also an integral part of convolutional neural networks, which often use pooling layers to reduce the spatial dimensions of the image [182], see also the design of the U-Net [168]. With a similar reasoning, multi-scale architectures for normalizing flows were proposed in the RealNVP framework [58] and are an integral part of many successful invertible neural networks, for example, Glow [107] or the invertible U-Net [67].

The multi-scale architecture defines a hierarchical feature space, where some variables are factored out at an earlier level of the flow, see Figure 2.2 for a schematic visualization. At each scale, multiple invertible transformations are combined in a transformation $f^{(i)}$. The transformation $f^{(i)}$ commonly includes an invertible downsampling operation to transform the $h \times w \times c$ image to a shape of $h/2 \times w/2 \times 4c$. In RealNVP, half of the channels are directly forwarded to the output, and the other half is passed to the next

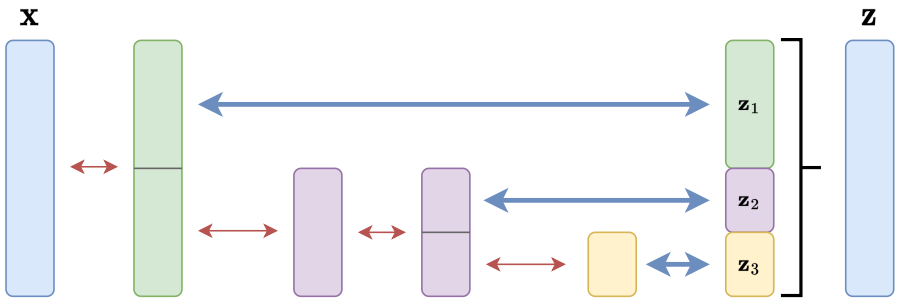

Figure 2.2: Schematic example of a three level multi-scale architecture. The red arrows denote invertible transformations and the blue arrows denote the splitting of dimensions and forwarding to the output.

transformation. This can be recursively defined as

$$\begin{aligned} \mathbf{h}^{(0)} &= \mathbf{x} \\ (\mathbf{h}^{(i+1)}, \mathbf{z}^{(i+1)}) &= f^{(i+1)}(\mathbf{h}^{(i)}), \quad i = 0, \dots, L-2 \\ \mathbf{z}^{(L)} &= f^{(L)}(\mathbf{h}^{(L-1)}) \\ \mathbf{z} &= (\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(L)}), \end{aligned} \tag{2.42}$$

where the final output $\mathbf{z}$ is defined as the concatenation of all intermediate $\mathbf{z}^{(i)}$. An example of a combination of invertible transformation used in RealNVP is provided in Figure 2.3.

As a result of this architecture, the model has to differentiate between finer features, factored out earlier, and coarser features, factored out later. As an additional practical benefit, the loss is distributed across the network, similar to intermediate layer guidance used in deeply supervised networks [120, 122]. Moreover, due to the smaller spatial size of intermediate activations, these architectures have a reduced memory cost and computation time. As a consequence, using this type of multi-scale architecture makes it possible to train deeper flow-based models. As an alternative, a fully invertible variation of the U-Net has been proposed by Etmann et al. [67] sharing similar computational advantages.

2.3 Application to Inverse Problems

In this section, we will discuss several ways of exploiting invertible neural networks for inverse problems. The applications can generally be classified into two groups: generative modeling and operator learning. For generative

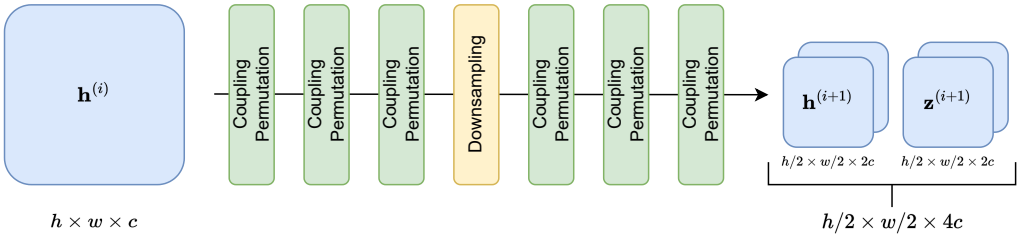

Figure 2.3: Operations for one scale in RealNVP [58]: First three coupling layers with permutations are used, then an invertible downsampling operation is performed, and finally three more coupling layers with permutations are applied. The first half of the output $\mathbf{h}^{(i+1)}$ is passed to the next scale, and the second half $\mathbf{z}^{(i+1)}$ is directly forwarded to the final output.

modeling, we can make use of invertible neural networks to build normalizing flow and use them for VI, learning the prior or estimating a posterior from a given dataset. Invertibility can also be used as an analytical tool to learn operators. In particular, as both the forward and inverse are Lipschitz continuous, we can estimate bi-Lipschitz function and learn regularizers for inverse problems, see Section 2.3.4.

2.3.1 Variational Inference

Normalizing flows are a promising class of probabilistic models for VI [180, 194]. As discussed in Section 1.1.2, the goal in VI is to find a tractable approximation of some known, but intractable, posterior $p^{\text{post}}(\mathbf{x}|\mathbf{y})$ by minimizing the discrepancy between the probabilistic model and the posterior. This discrepancy is usually measured with respect to the reverse KL divergence

$$D_{\text{KL}}(p_\theta(\mathbf{x})||p^{\text{post}}(\mathbf{x}|\mathbf{y})) = \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[-\log p^{\text{post}}(\mathbf{x}|\mathbf{y}) + \log p_\theta(\mathbf{x})], \tag{2.43}$$

where $p_\theta(\mathbf{x})$ is defined by a normalizing flow [31]. Given an invertible neural network $\mathcal{T}_\theta : \mathbb{R}^n \to \mathbb{R}^n$, with $\mathbf{x} = \mathcal{T}_\theta(\mathbf{z})$ and $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$, the optimization problem reduces to

$$\min_\theta \left\{\mathcal{L}_{\text{VI}}(\theta) = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[-\log p^{\text{post}}(\mathbf{y}|\mathcal{T}_\theta(\mathbf{z})) - \log|\det J_{\mathcal{T}_\theta}(\mathbf{z})|]\right\}, \tag{2.44}$$

where all term independent of θ have been dropped. The negative log-determinant of the Jacobian in Eqn. (2.44) is related to the negative entropy of the flow-based model and acts as a regularization term. Without taking this term into account, the normalizing flow would recover the MAP

solution. Sun et al. [194] introduce an additional parameter $\beta > 0$

$$\min_\theta \left\{ \mathcal{L}_{\beta\text{-VI}}(\theta) = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[-\log p^{\text{post}}(\mathbf{y}|\mathcal{T}_\theta(\mathbf{z})) - \beta \log|\det J_{\mathcal{T}_\theta}(\mathbf{z})|] \right\}, \quad (2.45)$$

where a large β results in a larger entropy for $p_\theta(\mathbf{x})$, thus increasing the diversity of samples.

The optimization is usually performed with gradient descent, requiring an estimating of the gradient $\nabla_\theta \mathcal{L}_{\text{VI}}(\theta)$. Given a set of samples $\{\mathbf{z}^{(i)}\}_{i=1}^N$ from the base distribution, this gradient can be obtained via a Monte Carlo estimation

$$\nabla_\theta \mathcal{L}_{\text{VI}}(\theta) \approx \sum_{i=1}^N \left(-\nabla_\theta \log p^{\text{post}}(\mathbf{y}|\mathcal{T}_\theta(\mathbf{z}^{(i)})) - \nabla_\theta \log\det J_{\mathcal{T}_\theta}(\mathbf{z}^{(i)}) \right), \quad (2.46)$$

where both the gradient of the posterior and the gradient of the log-det term can be computed via backpropagation. Usually, for each training step a new set of samples $\{\mathbf{z}^{(i)}\}_{i=1}^N$ is drawn to estimate the gradient [151].

A particularly interesting special case are linear inverse problems with additive Gaussian noise, where we again exploit Bayes' theorem to decompose the posterior into likelihood and prior. Assuming additive Gaussian noise and a Gibbs prior [103], the likelihood and prior are given by

$$p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}) \propto \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 \right), \quad \pi(\mathbf{x}) \propto \exp\left(-\lambda\|\mathbf{L}\mathbf{x}\|_2^2\right), \quad (2.47)$$

for some matrix $\mathbf{L}$. By imposing a base distribution $p_{\mathbf{z}} \sim \mathcal{N}(0, I)$, we arrive at the objective

$$\min_\theta \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,I)} \left[\frac{1}{2\sigma}\|\mathbf{A}\mathcal{T}_\theta(\mathbf{z}) - \mathbf{y}\|_2^2 + \lambda\|\mathbf{L}\mathcal{T}_\theta(\mathbf{z})\|_2^2 - \log\det J_{\mathcal{T}_\theta}(\mathbf{z}) \right]. \quad (2.48)$$

VI only requires a single measurement $\mathbf{y}$ and has similar computational requirements as the DIP (cf. Section 1.4.4). The VI framework can be extended to a variety of priors. For example, Siahkoohi et al. [180] first learn model the prior $\pi(\mathbf{x})$ with a flow-based models given a dataset $\{\mathbf{x}^{(i)}\}_{i=1}^N$ of images from our target domain. In the second step, this learned prior is fixed and a normalizing flow is trained to approximate the resulting posterior.

A drawback of VI is the high computational cost, as the normalizing flow has to be retrained for every new measurement $\mathbf{y}$. To reduce the computational cost, conditional flow-based models can be employed to perform amortized VI [108, 131, 150]. In amortized VI, we assume access to a dataset of measurements $\{\mathbf{y}^{(i)}\}_{i=1}^N$ and learn a conditional normalizing flow with $\mathbf{y}^{(i)}$ as an additional input. The objective function in Eqn. (2.44)

has to be changed to include the measurements

$$\begin{aligned}\min_\theta \{&\mathcal{L}_{\text{a-VI}}(\theta) = \\ &\mathbb{E}_{\mathbf{y}\sim p_{\mathbf{y}}(\mathbf{y})}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[-\log p^{\text{post}}(\mathcal{T}_\theta(\mathbf{z},\mathbf{y})|\mathbf{y}) - \log|\det J_{\mathcal{T}_\theta}(\mathbf{z};\mathbf{y})|]\},\end{aligned} \tag{2.49}$$

where the Jacobian is only computed with respect to the base variable $\mathbf{z}$. After the initial training phase, the posterior for a new measurement can be approximated with the trained conditional normalizing flow, without the need for re-training.

2.3.2 Learning the Prior

We follow the setting introduced in Section 1.4.2. We assume, that a dataset $\{\mathbf{x}^{(i)}\}_{i=1}^N$ with $\mathbf{x}^{(i)} \sim p_{\text{data}}(\mathbf{x})$ i.i.d., is available. By minimizing the negative log-likelihood

$$\mathcal{L}_{\text{nll}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\|\mathcal{T}_\theta^{-1}(\mathbf{x}^{(i)})\|_2^2 - \log|\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x}^{(i)})|\right), \tag{2.50}$$

with a Gaussian base distribution, we can train a normalizing flow to approximate the unknown data distribution, see also Section 2.1. This trained prior can be integrated in several downstream applications. Most works focus on the integration of a normalizing flow prior in MAP estimation, see [18, 89, 114, 211, 212] or [7]. However, normalizing flow priors have recently also been used in Langevin sampling [35]. In the MAP approach, the negative log-likelihood of the trained normalizing flow is used as a regularizer:

$$\hat{\mathbf{x}} = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n} -\log p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}) - \log p_\theta(\mathbf{x}). \tag{2.51}$$

Due to the invertibility of the normalizing flow, one can formulate this optimization problem in the latent space [18, 89], i.e.,

$$\hat{\mathbf{z}} = \operatorname*{argmin}_{\mathbf{z}\in\mathbb{R}^n} -\log p^{\text{lkhd}}(\mathbf{y}|\mathcal{T}_\theta(\mathbf{z})) - \log p_{\mathbf{z}}(\mathbf{z}), \tag{2.52}$$

and obtain the final reconstruction as $\hat{\mathbf{x}} = \mathcal{T}_\theta(\hat{\mathbf{z}})$. In the case of a Gaussian base distribution and a Gaussian noise model Eqn. (2.52) reduces to

$$\hat{\mathbf{z}} = \operatorname*{argmin}_{\mathbf{z}\in\mathbb{R}^n} \frac{1}{2\sigma_y^2}\|\mathbf{A}\mathcal{T}_\theta(\mathbf{z}) - \mathbf{y})\|_2^2 + \gamma\|\mathbf{z}\|_2^2, \tag{2.53}$$

where $\gamma > 0$ is an additional penalty parameter. It has been observed that optimizing over the latent space, instead of the pixel domain, is less challenging [211]. However, as the normalizing flow $\mathcal{T}_\theta$ can become unstable (cf. exploding Lipschitz constant, see Section 2.1.1.3) when learning

multi-modal distributions, naively optimizing over the latent space may lead to degraded or distorted reconstructions [18, 89].

To address this problem, several regularization strategies and optimization techniques have been proposed. For example, the non-linear optimization problem in Eqn. (2.52) is initialized with $\mathbf{z}_0 = \mathbf{0}$ [18], which corresponds to the maximum likelihood $\mathbf{z}$ in latent space. This initialization ensures a starting point in a relatively high likelihood region. Another approach is to regularize the learned mapping $\mathcal{T}_\theta$ by adding specific regularization to the training loss. For example, in addition to the negative-log likelihood $\mathcal{L}_{\text{nll}}(\theta)$ a *latent-noise* loss $\mathcal{L}_{\text{ln}}(\theta) = \|\mathcal{T}_\theta(\mathbf{z}+\eta) - \mathcal{T}_\theta(\mathbf{z})\|$ with $\mathbf{z} = \mathcal{T}_\theta^{-1}(\mathbf{x})$ and random noise η has been used [89]. This latent-noise loss encourages similar elements in latent space, to be mapped to similar images, essentially constraining the Lipschitz constant of $\mathcal{T}_\theta$. The magnitude of the noise η connects to the strength of this regularization.

Further, some techniques specifically exploit the multi-scale architecture used for many normalizing flow models. In *coarse-to-fine optimization* [89] the authors directly use the decomposition of the base variable $\mathbf{z} = (\mathbf{z}_1, \dots, \mathbf{z}_L)$, where $\mathbf{z}_L$ is the split output from the last layer relating to the coarse structure of the image. They define a sequence of optimization problems

$$\begin{aligned} \hat{\mathbf{z}}_k = \underset{\mathbf{z}_k}{\operatorname{argmin}} \frac{1}{2\sigma_y^2} \|\mathbf{A}\mathcal{T}_\theta(\mathbf{z}) - \mathbf{y})\|_2^2 + \gamma \|\mathbf{z}_k\|_2^2 \\ \text{with } \mathbf{z} = (\bar{\mathbf{z}}_1, \dots, \bar{\mathbf{z}}_{k-1}, \mathbf{z}_k, \hat{\mathbf{z}}_{k+1}, \dots, \hat{\mathbf{z}}_L), \end{aligned} \tag{2.54}$$

for $k = L, \dots, 1$. Here $\hat{\mathbf{z}}_{k+1}, \dots, \hat{\mathbf{z}}_L$ are fixed from the prior optimization problems and $\bar{\mathbf{z}}_1, \dots, \bar{\mathbf{z}}_{k-1}$ are taken as the mean values from the training data. This means, that for each optimization problem in this sequence, only one part of the base variable is optimized.

A large collection of images $\{\mathbf{x}^{(i)}\}_{i=1}^N$ is needed to train a suitable prior. This problem is not inherent to normalizing flows, but rather has been observed in many image prior models, where it is often tackled by the use of patch-based priors [170, 220]. In the field of normalizing flows, patch-based methods have been explored by Helminger et al. [89]. However, in the reconstruction step, they reconstruct all image patches individually, which only works for local image degradations, i.e., denoising, deblurring, or inpainting. In PatchNR [7], we also train a normalizing flow on image patches but use an all-at-once approach to reconstruction.

2.3.3 Learning the Posterior

The task of learning the posterior with normalizing flows is similar to learning a prior. As discussed in Section 2.1.2, for each fixed $\mathbf{y} \in \mathbb{R}^m$ the transformation $\mathcal{T}_\theta(\cdot, \mathbf{y})$ defines a normalizing flow. Using a paired dataset $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$, we can train the conditional flow-based model using an empirical approximation to Eqn. (2.14)

$$\mathcal{L}_{\text{nll}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(-\log p_{\mathbf{z}}(\mathcal{T}_\theta^{-1}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})) - \log |\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x}^{(i)}; \mathbf{y}^{(i)})| \right). \tag{2.55}$$

For the typical choice of a Gaussian base distribution, we get:

$$\mathcal{L}_{\text{nll}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2} \|\mathcal{T}_\theta^{-1}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\|_2^2 - \log |\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x}^{(i)}; \mathbf{y}^{(i)})| \right). \tag{2.56}$$

This approach is entirely data-driven as in this training loss no information of the forward operator $\mathbf{A}$ is used. As already pointed out in Section 2.1.2, most approaches do not use the measurements $\mathbf{y}$ directly as an input, but rather some sort of preprocessing. Different forms of preprocessing were for example applied for computed tomography reconstruction [52], magnetic resonance imaging [53], photoacoustic imaging [150] and seismic imaging [181]. Using adjoint preprocessing, the loss function further reduces to

$$\mathcal{L}_{\text{nll}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2} \|\mathcal{T}_\theta^{-1}(\mathbf{x}^{(i)}, \mathbf{A}^* \mathbf{y}^{(i)})\|_2^2 - \log |\det J_{\mathcal{T}_\theta^{-1}}(\mathbf{x}^{(i)}; \mathbf{A}^* \mathbf{y}^{(i)})| \right). \tag{2.57}$$

The trained conditional normalizing flow can be used to compute the conditional mean as the reconstruction. The conditional mean can be computed as

$$\hat{\mathbf{x}}_{\text{CM}} = \frac{1}{K} \sum_{i=1}^{K} \mathcal{T}_\theta(\mathbf{z}^{(i)}, \mathbf{A}^* \mathbf{y}), \quad \mathbf{z}^{(i)} \sim p_{\mathbf{z}}(\mathbf{z}), \tag{2.58}$$

for a new measurement $\mathbf{y} \in \mathbb{R}^m$. Even with this preprocessing, there is no part in the training process directly enforcing data consistency. Nonetheless, in the evaluation of the LoDoPab-CT challenge, we found that this conditional mean reconstruction has a similar data consistency to other data-driven approaches, see Table 3 in [121].

It is important to distinguish the difference between general conditional density estimation and posterior estimation. Estimating the posterior $p^{\text{post}}(\mathbf{x}|\mathbf{y})$ given samples is a different task than estimating some

other conditional density as the posterior comes with a lot of stability estimates. In most statistical inverse problems, the posterior is robust to changes in $\mathbf{y}$ with respect to the Hellinger metric [193]. Latz [117] extends this stability also to other metrics. These estimates hold in particular for the example of linear inverse problems with additive Gaussian noise. These stability estimates were recently extended to other conditional generative models [8]. However, the extension to conditional normalizing flows is still missing.

2.3.4 Operator Learning

Besides the applications for statistical inverse problems, invertible neural networks can also be used in the functional analytic framework, where some works studying the approximation capabilities of invertible networks. Teshima et al. [199] show that affine coupling flows are L^p-universal approximators for C^2 diffeomorphism. Roughly speaking, for any C^2 diffeomorphism g, there exist an affine flow f, such that $\|f - g\|_{p,K} < \epsilon$ for any $\epsilon > 0$ and any compact subset K. Here $\|\cdot\|_{p,K}$ is the L^p norm restricted to K. This result also extends to distributional universality, which is of interest for generative modeling. In a recent work, Lyu et al. [130] claim even universality for C^k diffeomorphisms even in the C^k norm. However, they need a lifting to higher dimensions, i.e., a padding with zeros, to achieve this. Both of these results give no rate of the approximation, for example in terms of the number of affine coupling blocks.

These theoretical works give justification that bi-Lipschitz functions can be reasonably well approximated using invertible neural networks. Recently Jin et al. [98] showed specific approximation rates for bi-Lipschitz functions by an explicitly constructing of the invertible mapping. They use an invertible neural network $\mathcal{T}_\theta$ to learn both the forward operator $F : \mathcal{X} \to \mathcal{Y}$ and the inverse $F^{-1} : \mathcal{Y} \to \mathcal{X}$ at the same time using a dataset $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$ with a joint reconstruction loss

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \left(\|\mathcal{T}_\theta(\mathbf{x}^{(i)}) - \mathbf{y}^{(i)}\|_{\mathcal{Y}}^2 + \|\mathcal{T}_\theta^{-1}(\mathbf{y}^{(i)}) - \mathbf{x}^{(i)}\|_{\mathcal{X}}^2 \right). \tag{2.59}$$

Further, they propose an approach for approximating bi-Lipschitz functions on infinite-dimensional spaces by combining model reduction using principal component analysis with an invertible neural network mapping.

An application to linear inverse problems has been explored by Arndt et al. [14]. In the discrete setting, we work with a linear forward

operator $\mathbf{A} : \mathbb{R}^n \to \mathbb{R}^m$, usually with $m < n$. Approximating the forward operator in itself using an invertible neural network is only feasible in the setting of $n = m$, i.e., if the dimensions of the input and output space are the same. As a consequence, in our work, we study approximations to the normal equation $\mathcal{T}_\theta \approx \mathbf{A}^T\mathbf{A}$ and implement $\mathcal{T}_\theta$ as a one-block invertible residual network. Solving the inverse problem is then a two-step process:

1) Training the invertible neural network on a supervised dataset $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$:

$$\begin{aligned} &\min_\theta \frac{1}{N}\sum_{i=1}^{N} \|\mathcal{T}_\theta(\mathbf{x}^{(i)}) - \mathbf{A}^T\mathbf{y}^{(i)}\|_2^2 \\ &\text{with } \mathcal{T}_\theta(\mathbf{x}) = \mathbf{x} - f_\theta(\mathbf{x}), \quad \mathrm{Lip}(f_\theta) \leq L < 1. \end{aligned} \tag{2.60}$$

2) Using $\mathcal{T}_\theta^{-1}$ to obtain a reconstruction for new measurements $\mathbf{y}$:

$$\hat{\mathbf{x}} = \mathcal{T}_\theta^{-1}(\mathbf{A}^T\mathbf{y}). \tag{2.61}$$

The maximum Lipschitz constant L is a design choice and is related to the Lipschitz constant of the inverse via $\mathrm{Lip}(\mathcal{T}_\theta^{-1}) \leq 1/(1-L)$. Increasing the Lipschitz constant leads thus to a higher expressiveness, but also decreases robustness and thus sensitivity to noise. By varying the Lipschitz constant L, we define a family of reconstructions $\mathcal{R}_L = \mathcal{T}_{\theta,L}^{-1} \circ \mathbf{A}^T$, where L acts as a regularization parameter and $\mathcal{R}_L$ is a regularizer. This regularization depends both on the architecture of the residual network f_θ and the training data used. Recently, this analysis was extended to different training approaches and loss functions [15].

2.4 Score-based Diffusion Models

Score-based diffusion models are the current state of the art for generative modeling [55, 189]. They emerged as a new interpretation of denoising diffusion probabilistic models [91, 184] and score-matching Langevin dynamics [187]. The training of a normalizing flow is set up in a way, that images are transformed into noise by $\mathcal{T}_\theta^{-1}$. By learning this image-to-noise map with an invertible transformation, we implicitly get access to the noise-to-image mapping, necessary for sampling. For score-based diffusion models, we go the other way around: We set up a fixed forward process to transform images into noise and learn to invert this process step by step. For this presentation, we follow the notation introduced in [189].

The forward process, i.e., transforming images into noise, is defined as an Itô stochastic differential equation (SDE) [175]. This forward diffusion process takes a sample of the data distribution and perturbs it with noise according to

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + g(t)d\mathbf{w}_t, \quad \mathbf{x}_0 \sim p_0 \coloneqq p_{\text{data}}, \tag{2.62}$$

where $\{\mathbf{x}_t\}_t$ is a stochastic process indexed by time t and $\{\mathbf{w}_t\}_t$ is Brownian motion. Through this SDE a time-dependent density $p(\mathbf{x}_t)$ is imposed, where we write $p_t(\mathbf{x}_t) \coloneqq p(\mathbf{x}_t)$ to explicitly emphasize this time dependency. The drift function $\mathbf{f}(\cdot, t) : \mathbb{R}^n \to \mathbb{R}^n$ and the diffusion function $g : \mathbb{R}^n \to \mathbb{R}$ define how the time-dependent density $p_t(\mathbf{x}_t)$ evolves over time.[3] Drift and diffusion functions are chosen in such a way that the terminal distribution at $t = T$ approximates a standard Gaussian, i.e., $p_T \approx \mathcal{N}(0, \mathbf{I})$, or another tractable distribution.

Under weak assumptions[4], there exist a reverse diffusion process [10], mapping noise to the data distribution. This reverse diffusion process is given as

$$d\mathbf{x}_t = [\mathbf{f}(\mathbf{x}_t, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)]dt + g(t)d\tilde{\mathbf{w}}_t, \tag{2.63}$$

where $\{\tilde{\mathbf{w}}_t\}_t$ is a time-reverse Brownian motion and the SDE is solved backward in time. The term $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$ is called the *score function* and plays a central role in score-based diffusion models.

The goal in score-based generative modeling is to train a neural network, called the score model, to approximate this score function by minimizing the explicit score matching (ESM) objective

$$L_{\text{ESM}}(\theta) = \mathbb{E}_{t\sim U[0,T]} \mathbb{E}_{\mathbf{x}_t \sim p_t(\mathbf{x}_t)} \left[\omega_t \| s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t) \|_2^2 \right]. \tag{2.64}$$

However, as the score function is generally unknown, this optimization problem is intractable. Already in 2005, Hyvarinen [95] proposed an implicit score matching objective, circumventing the evaluation of the score function. Vincent [210] provides a connection between implicit score matching and training a denoiser, proposing denoising score matching (DSM). In the DSM

[3] Note that the diffusion function can be further generalized to matrix-valued functions $G : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^{n\times n}$, with an additional dependence on $\mathbf{x}_t$, see Appendix A in [189].

[4] The drift function $\mathbf{f}$ is usually chosen as $\mathbf{f} \equiv 0$ [189] or as a linear function in $\mathbf{x}$ [91]. Both choices satisfy the assumptions in [10].

framework, the time-conditional neural network $s_\theta(\mathbf{x}_t, t)$ is trained using

$$\min_\theta \Big\{ L_{\mathrm{DSM}}(\theta) = \mathbb{E}_{t\sim U[0,T]} \mathbb{E}_{\mathbf{x}_0 \sim p_{\mathrm{data}}(\mathbf{x})} \mathbb{E}_{\mathbf{x}_t \sim p_t(\mathbf{x}_t|\mathbf{x}_0)} \left[\omega_t \| s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t|\mathbf{x}_0) \|_2^2 \right] \Big\}, \tag{2.65}$$

where $\omega_t > 0$ are weighting factors, balancing the different time steps. Vincent [210] proves, that $L_{\mathrm{DSM}}(\theta)$ and $L_{\mathrm{ESM}}(\theta)$ have the same minimizer. This result is remarkable as it shows that it is sufficient to only match the transition densities $p_t(\mathbf{x}_t|\mathbf{x}_0)$ to approximate the full density $p_t(\mathbf{x}_t)$. For SDEs with an affine linear drift the transition densities are Gaussians and have closed-form expressions, making Eqn. (2.65) efficient to evaluate [175]. The two most widely used SDEs are the variance preserving SDE [91] and the variance exploding SDE [187], both of which result in transitions densities $p_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \gamma_t \mathbf{x}_0, \nu_t^2 \mathbf{I})$, where γ_t, ν_t can be computed from the drift and diffusion functions. For these SDEs, we can simplify the loss function to

$$L_{\mathrm{DSM}}(\theta) = \mathbb{E}_{t\sim U[0,T]} \mathbb{E}_{\mathbf{x}_0 \sim p_{\mathrm{data}}} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})} \left[\omega_t \left\| s_\theta(\mathbf{x}_t, t) + \frac{\mathbf{z}}{\nu_t} \right\|_2^2 \right], \tag{2.66}$$

with $\mathbf{x}_t = \gamma_t \mathbf{x}_0 + \nu_t \mathbf{z}$ and $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. For the choice of $\omega_t = \nu_t^2$, the DSM objective (plus an additional constant independent of θ) is an upper bound of the negative log-likelihood

$$-\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}(\mathbf{x})} [\log p_\theta(\mathbf{x})] \leq L_{\mathrm{DSM}}(\theta) + C, \tag{2.67}$$

where C is a constant and $p_\theta(\mathbf{x})$ the probabilistic model defined by the reverse SDE, see [188, Corollary 1].

To highlight the connection between denoising and training the score model, we can make use of a result by Tweedie [62]. Given a trained score model, the minimum mean-squared-error denoiser is given by

$$\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \frac{\mathbf{x}_t + \nu_t^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)}{\gamma_t} \approx \frac{\mathbf{x}_t + \nu_t^2 s_\theta(\mathbf{x}_t, t)}{\gamma_t} := \hat{\mathbf{x}}_0(\mathbf{x}_t). \tag{2.68}$$

Further, given a trained denoiser $D(\mathbf{x}_t, t) \approx \mathbf{x}_0$, we can recover the score model as

$$s_\theta(\mathbf{x}_t, t) \approx \frac{\gamma_t D(\mathbf{x}_t, t) - \mathbf{x}_t}{\nu_t^2}. \tag{2.69}$$

The equivalence between denoising and score modeling has been successfully utilized for Bayesian inverse problems and included in Langevin sampling algorithms [101, 102, 118]. Sampling from a score-based diffusion model

requires solving the reverse SDE with the score model in-place of $\nabla_x \log p_t(\mathbf{x}_t)$

$$d\mathbf{x}_t = [\mathbf{f}(\mathbf{x}_t, t) - g(t)^2 s_\theta(\mathbf{x}_t, t)]dt + g(t)d\bar{\mathbf{w}}_t. \tag{2.70}$$

Here, Euler-Maruyama [189] is one of the most basic methods for discretization of the resulting SDE. Starting with $\mathbf{x}_T \sim p_T$, the Euler-Maruyama sampling update is given by

$$\mathbf{x}_{t-\Delta t} = \mathbf{x}_t - [\mathbf{f}(\mathbf{x}_t, t) - g(t)^2 s_\theta(\mathbf{x}_t, t)]|\Delta t| + g(t)\sqrt{|\Delta t|}\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}), \tag{2.71}$$

which decomposes into an explicit Euler update and adding noise at each iteration. For high-quality samples, usually a small time step $|\Delta t|$ is required, requiring a high number of iterations and evaluations of the score model. There has been a lot of research in reducing the number of sampling, for example by using denoising diffusion implicit models [185].

2.4.1 Connection to Continuous Normalizing Flows

The perturbation with the SDE is connected to a diffusion of the density function via the Fokker-Planck equation [32, 175]:

$$\frac{dp(\mathbf{x}, t)}{dt} = -\sum_{i=1}^{n} \frac{\partial}{\partial x_i}(f_i(\mathbf{x}, t)p(\mathbf{x}, t)) + \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2}{\partial x_j \partial x_i}\frac{g(t)^2}{2}p(\mathbf{x}, t), \tag{2.72}$$

where $p(\mathbf{x}, t)$ is the (now) time dependent density with $p(\mathbf{x}, t = 0) = p_{\text{data}}(\mathbf{x})$ and $\mathbf{f} = (f_1, \ldots, f_n)$ and g are drift and diffusion functions, respectively. The Fokker-Planck equation can be reformulated as

$$\frac{dp(\mathbf{x}, t)}{dt} = -\sum_{i=1}^{n} \frac{\partial}{\partial x_i}\left[\left(f_i(\mathbf{x}, t) - \frac{g(t)^2}{2}(\nabla_{\mathbf{x}} \log p(\mathbf{x}, t))_i\right) p(\mathbf{x}, t)\right]. \tag{2.73}$$

This means that the SDE in Eqn. (2.62) has the same marginal densities $p(\mathbf{x}, t)$ as the following ODE

$$d\mathbf{x} = \left[f(\mathbf{x}, t) - \frac{g(t)^2}{2}\nabla_{\mathbf{x}} \log p(\mathbf{x}, t)\right] dt, \tag{2.74}$$

which is referred to as the *probability flow ODE* [189]. In this way, the probability flow ODE defines a continuous normalizing flow. The continuous normalizing flow framework, as presented in Section 2.1.3, has the disadvantage of a slow maximum likelihood training, i.e., two ODEs have to be solved for each optimization step when using the adjoint ODE for gradient computation. Using this probability flow ODE together with denoising score matching offers a simulation-free training of the model.

By exploiting the connection of ODEs to SDEs one can compute the likelihood as

$$\log p_0(\mathbf{x}(0)) = \log p_T(\mathbf{x}(T)) + \int_0^T \operatorname{div}(\tilde{f}(\mathbf{x}, t))dt, \tag{2.75}$$

with $\tilde{f}(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}, t) - \frac{g(t)^2}{2}\nabla_{\mathbf{x}} \log p_t(\mathbf{x}, t)$. Note that continuous normalizing flows and score-based generative models usually define the time the other way around. For score-based diffusion models, we start at $t = 0$ with the data distribution $p_{\text{data}}(\mathbf{x})$, and for $t = T$, we arrive at $p_T(\mathbf{x}(T), T) \approx \mathcal{N}(\mathbf{x}(T); \mu, \Sigma)$ which should approach some Gaussian (the final form depends on the forward SDE). For continuous normalizing flows, we usually define it the other way around, i.e., we start at $t = 0$ with the base distribution. However, this simply corresponds to a different parametrization.

Sampling from the corresponding probability flow ODE requires to simulate Eqn. (2.74) backwards in time, starting at $\mathbf{x}(T) \sim p_T$. For the ODE simulation, any off-the-shelf ODE solver can be used. However, samples from the corresponding probability ODE often have a lower visual quality than samples from the reverse SDE. The gap between the ODE and SDE formulation, and particularly the impact on sampling, has recently been explored [54].

2.4.2 Application to Inverse Problems

Score-based diffusion models can be used to estimate the posterior $p^{\text{post}}(\mathbf{x}|\mathbf{y})$. Either the posterior can be directly modeled by a conditional score-based diffusion model (see e.g. [24]) or the score-based diffusion model can be used to learn the prior $\pi(\mathbf{x})$ [189]. A score-based diffusion model for the posterior requires drawing samples from the reverse SDE

$$d\mathbf{x}_t = [\mathbf{f}(\mathbf{x}_t, t) - g(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y})]dt + g(t)d\bar{\mathbf{w}}_t, \tag{2.76}$$

with a time-dependent posterior score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y})$. Using Bayes' theorem, the time-dependent posterior can be decomposed in

$$\begin{aligned} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{y}) &= \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t) \\ &\approx s_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t), \end{aligned} \tag{2.77}$$

where $s_\theta(\mathbf{x}_t, t)$ is a score model trained to approximate the prior $\pi(\mathbf{x})$ and $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t)$ is the time-dependent likelihood. In particular, $s_\theta(\mathbf{x}_t, t)$ can be pre-trained and is independent of the inverse problem under consideration, i.e., it can be reused for different tasks.

Using the pre-trained unconditional score model $s_\theta(\mathbf{x}_t, t)$ for posterior

estimation, requires to evaluate the time-dependent likelihood. However, this time-dependent likelihood is intractable as it can be factorized as

$$p_t(\mathbf{y}|\mathbf{x}_t) = \int_{\mathbb{R}^n} p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}_0) p(\mathbf{x}_0|\mathbf{x}_t) d\mathbf{x}_0 \tag{2.78}$$

and this high-dimensional integral is hard to compute. To still enable posterior sampling, different approximations were proposed. For example in some works [97, 160] the time-dependent likelihood is simply approximated with the scaled likelihood as

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t) = \lambda_t \nabla_{\mathbf{x}_t} \log p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}_t), \tag{2.79}$$

with a time-dependent penalty λ_t. Chung et al. [44] propose DDS and make use of Tweedie's formula, cf. Eqn.(2.68), to obtain $\hat{\mathbf{x}_0}(\mathbf{x}_t) \approx \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$ and use the approximation

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t) \approx \nabla_{\mathbf{x}_t} \log p^{\text{lkhd}}(\mathbf{y}|\hat{\mathbf{x}_0}(\mathbf{x}_t)). \tag{2.80}$$

However, this approximation comes with a higher computational cost as the gradients have to be computed through the score model. In the case of a linear inverse problem with Gaussian noise, $p^{\text{lkhd}}(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}; \mathbf{A}\mathbf{x}, \sigma_{\mathbf{y}}^2 \mathbf{I}_m)$, Boys et al. [33] use a Gaussian approximation to $p(\mathbf{x}_0|\mathbf{x}_t)$, enabling them to analytically evaluate the time-dependent likelihood in Eqn. (2.78).

The Euler-Maruyama discretization usually needs about 1000 time-steps to produce realistic results [91, 185, 190, 189]. Another sampling scheme, referred to as denoising diffusion implicit models (DDIM), was proposed to speed up the sampling process, requiring fewer iterations [185]. This sampling process does not make use of the Euler-Maruyama discretization. Recently, a new framework was proposed by Chung et al. [45] was developed to modify the DDIM sampling update for linear inverse problems.

Bibliography

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

[2] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6): 33–41, 1984.

[3] J. Adler and O. Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

[4] J. Adler and O. Öktem. Learned primal-dual reconstruction. *IEEE transactions on medical imaging*, 37(6):1322–1332, 2018.

[5] J. Adler and O. Öktem. Deep posterior sampling: Uncertainty quantification for large scale inverse problems. *Medical Imaging with Deep Learning*, 2019.

[6] J. Adler, S. Lunz, O. Verdier, C.-B. Schönlieb, and O. Öktem. Task adapted reconstruction for inverse problems. *Inverse Problems*, 38(7): 075006, 2022.

[7] F. Altekrüger, A. Denker, P. Hagemann, J. Hertrich, P. Maass, and G. Steidl. Patchnr: learning from very few images by patch normalizing flow regularization. *Inverse Problems*, 39:125018, 2023.

[8] F. Altekrüger, P. Hagemann, and G. Steidl. Conditional generative models are provably robust: Pointwise guarantees for bayesian inverse problems. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

[9] B. Amos, L. Xu, and J. Z. Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

[10] B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[11] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen. On instabilities of deep learning in image reconstruction and the potential costs of ai. *Proceedings of the National Academy of Sciences*, 117(48): 30088–30095, 2020.

[12] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019.

[13] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[14] C. Arndt, A. Denker, S. Dittmer, N. Heilenkötter, M. Iske, T. Kluth, P. Maass, and J. Nickel. Invertible residual networks in the context of regularization theory for linear inverse problems. *Inverse Problems*, 39 (10):104004, 2023.

[15] C. Arndt, S. Dittmer, N. Heilenkötter, M. Iske, T. Kluth, and J. Nickel. Bayesian view on the training of invertible residual networks for solving linear inverse problems. *arXiv preprint arXiv:2307.10431*, 2023.

[16] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019.

[17] V. Y. Arsenin. On ill-posed problems. *Russian Mathematical Surveys*, 31(6):93, 1976.

[18] M. Asim, M. Daniels, O. Leong, A. Ahmed, and P. Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *International Conference on Machine Learning*, pages 399–409. PMLR, 2020.

[19] D. O. Baguer, J. Leuschner, and M. Schmidt. Computed tomography reconstruction using deep image prior and learned reconstruction methods. *Inverse Problems*, 36(9):094004, 2020.

[20] R. Barbano, Ž. Kereta, A. Hauptmann, S. R. Arridge, and B. Jin. Unsupervised knowledge-transfer for learned image reconstruction. *Inverse Problems*, 38(10):104004, 2022.

[21] R. Barbano, J. Leuschner, J. Antorán, B. Jin, and J. M. Hernández-Lobato. Bayesian experimental design for computed tomography with the linearised deep image prior. *arXiv preprint arXiv:2207.05714*, 2022.

[22] R. Barbano, J. Leuschner, M. Schmidt, A. Denker, A. Hauptmann, P. Maass, and B. Jin. An educated warm start for deep image prior-based micro ct reconstruction. *IEEE Transactions on Computational Imaging*, 2022.

[23] R. Barbano, A. Denker, H. Chung, T. H. Roh, S. Arrdige, P. Maass, B. Jin, and J. C. Ye. Steerable conditional diffusion for out-of-distribution adaptation in imaging inverse problems. *arXiv preprint arXiv:2308.14409*, 2023.

[24] G. Batzolis, J. Stanczuk, C.-B. Schönlieb, and C. Etmann. Conditional image generation with score-based diffusion models. *arXiv preprint arXiv:2111.13606*, 2021.

[25] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen. Invertible residual networks. In *International conference on machine learning*, pages 573–582. PMLR, 2019.

[26] J. Behrmann, P. Vicol, K.-C. Wang, R. Grosse, and J.-H. Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR, 2021.

[27] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[28] M. Benning and M. Burger. Modern regularization methods for inverse problems. *Acta numerica*, 27:1–111, 2018.

[29] M. Benning, G. Gilboa, and C.-B. Schönlieb. Learning parametrised regularisation functions via quotient minimisation. *PAMM*, 16(1): 933–936, 2016.

[30] R. v. d. Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. In *34th Conference on Uncertainty in Artificial Intelligence*, 2018.

[31] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[32] N. M. Boffi and E. Vanden-Eijnden. Probability flow solution of the fokker–planck equation. *Machine Learning: Science and Technology*, 4 (3):035012, 2023.

[33] B. Boys, M. Girolami, J. Pidstrigach, S. Reich, A. Mosca, and O. D. Akyildiz. Tweedie moment projected diffusions for inverse problems. *arXiv preprint arXiv:2310.06721*, 2023.

[34] L. Bungert, R. Raab, T. Roith, L. Schwinn, and D. Tenbrinck. Clip: Cheap lipschitz training of neural networks. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 307–319. Springer, 2021.

[35] Z. Cai, J. Tang, S. Mukherjee, J. Li, C. B. Schönlieb, and X. Zhang. Nf-ula: Langevin monte carlo with normalizing flow prior for imaging inverse problems. *arXiv preprint arXiv:2304.08342*, 2023.

[36] L. Calatroni, C. Cao, J. C. De Los Reyes, C.-B. Schönlieb, and T. Valkonen. Bilevel approaches for learning of variational imaging models. *Variational Methods: In Imaging and Geometric Control*, 18 (252):2, 2017.

[37] D. Chen, J. Tachella, and M. E. Davies. Equivariant imaging: Learning beyond the range space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4379–4388, 2021.

[38] D. Chen, M. Davies, M. J. Ehrhardt, C.-B. Schönlieb, F. Sherry, and J. Tachella. Imaging with equivariant deep learning: From unrolled network design to fully unsupervised learning. *IEEE Signal Processing Magazine*, 40(1):134–147, 2023.

[39] R. T. Chen and D. K. Duvenaud. Neural networks with cheap differential operators. *Advances in Neural Information Processing Systems*, 32, 2019.

[40] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[41] R. T. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019.

[42] S. Chen and R. Gopinath. Gaussianization. *Advances in neural information processing systems*, 13, 2000.

[43] D. Chu, I. Demir, K. Eichensehr, J. G. Foster, M. L. Green, K. Lerman, F. Menczer, C. O'Connor, E. Parson, L. Ruthotto, et al. White paper: Deep fakery–an action plan. *Institute for Pure and Applied Mathematics (IPAM)*, 2020.

[44] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

[45] H. Chung, S. Lee, and J. C. Ye. Fast diffusion sampler for inverse problems by geometric decomposition. *arXiv preprint arXiv:2303.05754*, 2023.

[46] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[47] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[48] M. Z. Darestani, J. Liu, and R. Heckel. Test-time training can close the natural distribution shift performance gap in deep learning based compressed sensing. In *International Conference on Machine Learning*, pages 4754–4776. PMLR, 2022.

[49] M. Dashti and A. M. Stuart. The bayesian approach to inverse problems. *arXiv preprint arXiv:1302.6989*, 2013.

[50] J. C. De los Reyes, C.-B. Schönlieb, and T. Valkonen. Bilevel parameter learning for higher-order total variation regularisation models. *Journal of Mathematical Imaging and Vision*, 57(1):1–25, 2017.

[51] F. S. de Moura, S. Siltanen, and M. Juvonen. Helsinki deblur challenge 2021 (hdc20201) ipi special issue preface. *Inverse Problems and Imaging*, 17(5):i–iii, 2023.

[52] A. Denker, M. Schmidt, J. Leuschner, P. Maass, and J. Behrmann. Conditional normalizing flows for low-dose computed tomography image reconstruction. *Second workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (ICML)*, 2020.

[53] A. Denker, M. Schmidt, J. Leuschner, and P. Maass. Conditional invertible neural networks for medical imaging. *Journal of Imaging*, 7(11):243, 2021.

[54] T. Deveney, J. Stanczuk, L. M. Kreusser, C. Budd, and C.-B. Schönlieb. Closing the ode-sde gap in score-based diffusion models through the fokker-planck equation. *arXiv preprint arXiv:2311.15996*, 2023.

[55] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34: 8780–8794, 2021.

[56] A. G. Dimakis, A. Bora, D. Van Veen, A. Jalal, S. Vishwanath, and E. Price. Deep generative models and inverse problems. *Mathematical Aspects of Deep Learning*, page 400, 2022.

[57] L. Dinh, D. Krueger, and Y. Bengio. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[58] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR*, 2017.

[59] S. Dittmer. *On Deep Learning Applied to Inverse Problems: A Chicken-and-egg Problem*. Dissertation, Universität Bremen, 2020.

[60] F. Draxler, P. Sorrenson, L. Zimmermann, A. Rousselot, and U. Köthe. Free-form flows: Make any architecture a normalizing flow. *arXiv preprint arXiv:2310.16624*, 2023.

[61] M. Duff, N. D. Campbell, and M. J. Ehrhardt. Regularising inverse problems with generative machine learning models. *Journal of Mathematical Imaging and Vision*, pages 1573–7683, 2023.

[62] B. Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[63] B. Efron. *Large-scale inference: empirical Bayes methods for estimation, testing, and prediction*, volume 1. Cambridge University Press, 2012.

[64] H. W. Engl, K. Kunisch, and A. Neubauer. Convergence rates for tikhonov regularisation of non-linear ill-posed problems. *Inverse problems*, 5(4):523, 1989.

[65] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[66] L. eon Bottou. Online learning and stochastic approximations. *Online learning in neural networks*, 17(9):142, 1998.

[67] C. Etmann, R. Ke, and C.-B. Schönlieb. iUNets: Learnable invertible up- and downsampling for large-scale inverse problems. In *IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2020.

[68] B. G. Fitzpatrick. Bayesian analysis in inverse problems. *Inverse problems*, 7(5):675, 1991.

[69] M. Fornasier and H. Rauhut. Iterative thresholding algorithms. *Applied and Computational Harmonic Analysis*, 25(2):187–208, 2008.

[70] R. G. Gavaskar and K. N. Chaudhury. On the proof of fixed-point convergence for plug-and-play admm. *IEEE Signal Processing Letters*, 26(12):1817–1821, 2019.

[71] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[72] T. Germer, J. Robine, S. Konietzny, S. Harmeling, and T. Uelwer. Limited-angle tomography reconstruction via deep end-to-end learning on synthetic data. *Applied Mathematics for Modern Challenges*, 2023.

[73] D. Gilton, G. Ongie, and R. Willett. Model adaptation for inverse problems in imaging. *IEEE Transactions on Computational Imaging*, 7:661–674, 2021.

[74] A. Golinski, M. Lezcano-Casado, and T. Rainforth. Improving normalizing flows via better orthogonal parameterizations. In *ICML Workshop on Invertible Neural Networks and Normalizing Flows*, 2019.

[75] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[76] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[77] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *7th International Conference on Learning Representations, ICLR*, 2019.

[78] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.

[79] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[80] E. Haber and L. Tenorio. Learning regularization functionals—a supervised training approach. *Inverse Problems*, 19(3):611, 2003.

[81] J. Hadamard. Lectures on cauchy's problem in linear partial differential equations. *Press. New Haven*, 1923.

[82] P. Hagemann and S. Neumayer. Stabilizing invertible neural networks using mixture models. *Inverse Problems*, 37(8):085002, 2021.

[83] Y. Han, J. Yoo, H. H. Kim, H. J. Shin, K. Sung, and J. C. Ye. Deep learning with domain adaptation for accelerated projection-reconstruction mr. *Magnetic resonance in medicine*, 80(3): 1189–1205, 2018.

[84] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[85] A. Hauptmann and J. Poimala. Model-corrected learned primal-dual models for fast limited-view photoacoustic tomography. *arXiv preprint arXiv:2304.01963*, 2023.

[86] J. He, Y. Wang, and J. Ma. Radon inversion via deep learning. *IEEE transactions on medical imaging*, 39(6):2076–2087, 2020.

[87] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[88] R. Heckel and P. Hand. Deep decoder: Concise image representations from untrained non-convolutional networks. In *7th International Conference on Learning Representations, ICLR*, 2019.

[89] L. Helminger, M. Bernasconi, A. Djelouah, M. Gross, and C. Schroers. Generic image restoration with flow based priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 334–343, 2021.

[90] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[91] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[92] E. Hoogeboom, R. Van Den Berg, and M. Welling. Emerging convolutions for generative normalizing flows. In *International conference on machine learning*, pages 2771–2780. PMLR, 2019.

[93] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR, 2018.

[94] M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[95] A. Hyvärinen and P. Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6 (4), 2005.

[96] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[97] A. Jalal, M. Arvinte, G. Daras, E. Price, A. G. Dimakis, and J. Tamir. Robust compressed sensing mri with deep generative priors. *Advances in Neural Information Processing Systems*, 34:14938–14954, 2021.

[98] B. Jin, Z. Zhou, and J. Zou. On the approximation of bi-lipschitz maps by invertible neural networks. *arXiv preprint arXiv:2308.09367*, 2023.

[99] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE transactions on image processing*, 26(9):4509–4522, 2017.

[100] R. M. Johnson. The minimal transformation to orthonormality. *Psychometrika*, 31(1):61–66, 1966.

[101] Z. Kadkhodaie and E. Simoncelli. Stochastic solutions for linear inverse problems using the prior implicit in a denoiser. *Advances in Neural Information Processing Systems*, 34:13242–13254, 2021.

[102] Z. Kadkhodaie and E. P. Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser. In *NeurIPS Workshop on Deep Learning and Inverse Problems*, 2020.

[103] J. Kaipio and E. Somersalo. *Statistical and computational inverse problems*, volume 160. Springer Science & Business Media, 2006.

[104] B. Kawar, N. Elata, T. Michaeli, and M. Elad. Gsure-based diffusion model training with corrupted data. *arXiv preprint arXiv:2305.13128*, 2023.

[105] P. Kidger. *On neural differential equations*. Dissertation, University of Oxford, 2022.

[106] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[107] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[108] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR*, 2014.

[109] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[110] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

[111] A. Klenke. *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013.

[112] F. Knoll, T. Murrell, A. Sriram, N. Yakubova, J. Zbontar, M. Rabbat, A. Defazio, M. J. Muckley, D. K. Sodickson, C. L. Zitnick, et al. Advancing machine learning for mr image reconstruction with an open competition: Overview of the 2019 fastmri challenge. *Magnetic resonance in medicine*, 84(6):3054–3070, 2020.

[113] I. Kobyzev, S. J. Prince, and M. A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[114] K. Kothari, A. Khorashadizadeh, M. de Hoop, and I. Dokmanić. Trumpets: Injective flows for inference and inverse problems. In *Uncertainty in Artificial Intelligence*, pages 1269–1278. PMLR, 2021.

[115] J. Kruse, G. Detommaso, U. Köthe, and R. Scheichl. Hint: Hierarchical invertible neural transport for density estimation and bayesian inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8191–8199, 2021.

[116] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[117] J. Latz. On the well-posedness of bayesian inverse problems. *SIAM/ASA Journal on Uncertainty Quantification*, 8(1):451–482, 2020.

[118] R. Laumont, V. D. Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra. Bayesian imaging using plug & play priors: when langevin meets tweedie. *SIAM Journal on Imaging Sciences*, 15(2):701–737, 2022.

[119] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[120] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. Pmlr, 2015.

[121] J. Leuschner, M. Schmidt, P. S. Ganguly, V. Andriiashen, S. B. Coban, A. Denker, D. Bauer, A. Hadjifaradji, K. J. Batenburg, P. Maass, et al. Quantitative comparison of deep learning-based image reconstruction methods for low-dose and sparse-angle ct applications. *Journal of Imaging*, 7(3):44, 2021.

[122] J. Le'Clerc Arrastia, N. Heilenkötter, D. Otero Baguer, L. Hauberg-Lotte, T. Boskamp, S. Hetzer, N. Duschner, J. Schaller, and P. Maass. Deeply supervised unet for semantic segmentation to assist dermatopathological assessment of basal cell carcinoma. *Journal of imaging*, 7(4):71, 2021.

[123] T. Lindeberg. *Scale-space theory in computer vision*, volume 256. Springer Science & Business Media, 2013.

[124] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR*, 2022.

[125] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[126] A. K. Louis, P. Maaß, and A. Rieder. *Wavelets*. Springer-Verlag, 1998.

[127] F. Lucka. Bayesian inversion in biomedical imaging. *Dissertation, Westfälische Wilhelms-Universität Münster*, 2014.

[128] S. Lunz, O. Öktem, and C.-B. Schönlieb. Adversarial regularizers in inverse problems. *Advances in neural information processing systems*, 31, 2018.

[129] S. Lunz, A. Hauptmann, T. Tarvainen, C.-B. Schonlieb, and S. Arridge. On learned operator correction in inverse problems. *SIAM Journal on Imaging Sciences*, 14(1):92–127, 2021.

[130] J. Lyu, Z. Chen, C. Feng, W. Cun, S. Zhu, Y. Geng, Z. Xu, and C. Yongwei. Para-cflows: c^k-universal diffeomorphism approximators as superior neural surrogates. *Advances in Neural Information Processing Systems*, 35:28829–28841, 2022.

[131] C. C. Margossian and D. M. Blei. Amortized variational inference: When and why? *arXiv preprint arXiv:2307.11018*, 2023.

[132] Y. Marzouk, T. Moselhy, M. Parno, and A. Spantini. Sampling via measure transport: An introduction. *Handbook of uncertainty quantification*, 1:2, 2016.

[133] M. T. McCann, K. H. Jin, and M. Unser. Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine*, 34(6):85–95, 2017.

[134] T. Meinhardt, M. Moller, C. Hazirbas, and D. Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1781–1790, 2017.

[135] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[136] C. A. Metzler, A. Mousavi, R. Heckel, and R. G. Baraniuk. Unsupervised learning with stein's unbiased risk estimator. *arXiv preprint arXiv:1805.10531*, 2018.

[137] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[138] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR*, 2018.

[139] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[140] V. A. Morozov. *Methods for Solving Incorrectly Posed Problems*, volume 1. Springer, 1984.

[141] F. Mosteller and J. W. Tukey. Data analysis, including statistics. *Handbook of social psychology*, 2:80–203, 1968.

[142] S. Mukherjee, S. Dittmer, Z. Shumaylov, S. Lunz, O. Öktem, and C.-B. Schönlieb. Learned convex regularizers for inverse problems. *arXiv preprint arXiv:2008.02839*, 2020.

[143] S. Mukherjee, A. Hauptmann, O. Öktem, M. Pereyra, and C.-B. Schönlieb. Learned reconstruction methods with convergence guarantees: a survey of concepts and applications. *IEEE Signal Processing Magazine*, 40(1):164–182, 2023.

[144] K. P. Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[145] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[146] F. Natterer. *The mathematics of computerized tomography*. SIAM, 2001.

[147] R. Nickl. On bayesian inference for some statistical inverse problems with partial differential equations. *Bernoulli News*, 24(2):5–9, 2017.

[148] J. Oliva, A. Dubey, M. Zaheer, B. Poczos, R. Salakhutdinov, E. Xing, and J. Schneider. Transformation autoregressive networks. In *International Conference on Machine Learning*, pages 3898–3907. PMLR, 2018.

[149] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

[150] R. Orozco, A. Siahkoohi, G. Rizzuti, T. van Leeuwen, and F. J. Herrmann. Adjoint operators enable fast and amortized machine learning based bayesian uncertainty quantification. In *Medical Imaging 2023: Image Processing*, volume 12464, pages 357–367. SPIE, 2023.

[151] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1): 2617–2680, 2021.

[152] N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

[153] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[154] M. Pereyra. Proximal markov chain monte carlo algorithms. *Statistics and Computing*, 26:745–760, 2016.

[155] M. Pereyra, P. Schniter, E. Chouzenoux, J.-C. Pesquet, J.-Y. Tourneret, A. O. Hero, and S. McLaughlin. A survey of stochastic simulation and optimization methods in signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):224–241, 2015.

[156] J.-C. Pesquet, A. Repetti, M. Terris, and Y. Wiaux. Learning maximally monotone operators for image recovery. *SIAM Journal on Imaging Sciences*, 14(3):1206–1237, 2021.

[157] L. S. Pontrjagin, E. Mishchenko, V. Boltyanskii, and R. Gamkrelidze. *The mathematical theory of optimal processes*. Interscience, 1962.

[158] S. J. Prince. *Understanding Deep Learning*. MIT Press, 2023. URL http://udlbook.com.

[159] P. Putzky and M. Welling. Recurrent inference machines for solving inverse problems. *arXiv preprint arXiv:1706.04008*, 2017.

[160] Z. Ramzi, B. Remy, F. Lanusse, J.-L. Starck, and P. Ciuciu. Denoising score-matching for uncertainty quantification in inverse problems. In *34th Conference on Neural Information Processing Systems, Workshop on Deep Learning and Inverse Problems*, 2020.

[161] M. Ranzato, Y.-L. Boureau, S. Chopra, and Y. LeCun. A unified energy-based framework for unsupervised learning. In *Artificial Intelligence and Statistics*, pages 371–379. PMLR, 2007.

[162] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

[163] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Advances in neural information processing systems*, 32, 2019.

[164] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[165] A. Rieder. *Keine Probleme mit inversen Problemen: eine Einführung in ihre stabile Lösung*. Springer-Verlag, 2013.

[166] O. Rippel and R. P. Adams. High-dimensional probability estimation with deep density models, 2013.

[167] R. Rojas and R. Rojas. The backpropagation algorithm. *Neural networks: a systematic introduction*, pages 149–182, 1996.

[168] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[169] L. Rosasco, A. Caponnetto, E. Vito, F. Odone, and U. Giovannini. Learning, regularization and ill-posed inverse problems. *Advances in Neural Information Processing Systems*, 17, 2004.

[170] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 860–867. IEEE, 2005.

[171] N. Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. *Advances in neural information processing systems*, 25, 2012.

[172] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4): 259–268, 1992.

[173] L. Ruthotto and E. Haber. An introduction to deep generative modeling. *GAMM-Mitteilungen*, 44(2):e202100008, 2021.

[174] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[175] S. Särkkä and A. Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[176] J. Scarlett, R. Heckel, M. R. Rodrigues, P. Hand, and Y. C. Eldar. Theoretical perspectives on deep learning methods in inverse problems. *IEEE journal on selected areas in information theory*, 3(3):433–453, 2022.

[177] O. Scherzer, M. Grasmair, H. Grossauer, M. Haltmeier, and F. Lenzen. *Variational methods in imaging*, volume 167. Springer, 2009.

[178] T. Schuster, B. Kaltenbacher, B. Hofmann, and K. S. Kazimierski. *Regularization methods in Banach spaces*, volume 10. Walter de Gruyter, 2012.

[179] J. Schwab, S. Antholzer, and M. Haltmeier. Deep null space learning for inverse problems: convergence analysis and rates. *Inverse Problems*, 35(2):025008, 2019.

[180] A. Siahkoohi, G. Rizzuti, M. Louboutin, P. Witte, and F. Herrmann. Preconditioned training of normalizing flows for variational inference in inverse problems. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021.

[181] A. Siahkoohi, G. Rizzuti, R. Orozco, and F. J. Herrmann. Reliable amortized variational inference with physics-based latent distribution correction. *Geophysics*, 88(3):R297–R322, 2023.

[182] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[183] I. Singh, R. Barbano, Z. Kereta, B. Jin, K. Thielemans, and S. Arridge. 3d pet-dip reconstruction with relative difference prior using a sirf-based objective. Fully3D, 2023.

[184] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics.

In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[185] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations, ICLR*, 2021.

[186] Y. Song. Learning to generate data by estimating gradients of the data distribution. *Dissertation, Stanford University*, 2022.

[187] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[188] Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34:1415–1428, 2021.

[189] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR*, 2021.

[190] Y. Song, L. Shen, L. Xing, and S. Ermon. Solving inverse problems in medical imaging with score-based generative models. In *International Conference on Learning Representations, ICLR*, 2022.

[191] C. M. Stein. Estimation of the mean of a multivariate normal distribution. *The annals of Statistics*, pages 1135–1151, 1981.

[192] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, 36(2):111–133, 1974.

[193] A. M. Stuart. Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559, 2010.

[194] H. Sun and K. L. Bouman. Deep probabilistic imaging: Uncertainty quantification and multi-modal solution characterization for computational imaging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2628–2637, 2021.

[195] J. Sun, H. Li, Z. Xu, et al. Deep admm-net for compressive sensing mri. *Advances in neural information processing systems*, 29, 2016.

[196] E. G. Tabak and C. V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[197] A. Tarantola, B. Valette, et al. Inverse problems= quest for information. *Journal of geophysics*, 50(1):159–170, 1982.

[198] L. Tenorio. Statistical regularization of inverse problems. *SIAM review*, 43(2):347–366, 2001.

[199] T. Teshima, I. Ishikawa, K. Tojo, K. Oono, M. Ikeda, and M. Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators. *Advances in Neural Information Processing Systems*, 33:3362–3373, 2020.

[200] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

[201] A. N. Tikhonov et al. On the stability of inverse problems. In *Dokl. akad. nauk sssr*, volume 39, pages 195–198, 1943.

[202] J. M. Tomczak and M. Welling. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

[203] I. Tošić and P. Frossard. Dictionary learning. *IEEE Signal Processing Magazine*, 28(2):27–38, 2011.

[204] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[205] D. Van Veen, A. Jalal, M. Soltanolkotabi, E. Price, S. Vishwanath, and A. G. Dimakis. Compressed sensing with deep image prior and learned regularization. *arXiv preprint arXiv:1806.06438*, 2018.

[206] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[207] M. Vauhkonen, J. Kaipio, E. Somersalo, and P. Karjalainen. Electrical impedance tomography with basis constraints. *Inverse problems*, 13 (2):523, 1997.

[208] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE global conference on signal and information processing*, pages 945–948. IEEE, 2013.

[209] C. Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

[210] P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[211] X. Wei, H. van Gorp, L. Gonzalez-Carabarin, D. Freedman, Y. C. Eldar, and R. J. van Sloun. Deep unfolding with normalizing flow priors for inverse problems. *IEEE Transactions on Signal Processing*, 70:2962–2971, 2022.

[212] J. Whang, Q. Lei, and A. Dimakis. Solving inverse problems with a flow-based noise model. In *International Conference on Machine Learning*, pages 11146–11157. PMLR, 2021.

[213] D. R. Wilson and T. R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural networks*, 16(10): 1429–1451, 2003.

[214] C. Winkler, D. Worrall, E. Hoogeboom, and M. Welling. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*, 2019.

[215] C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon. Text-to-image diffusion model in generative ai: A survey. *arXiv preprint arXiv:2303.07909*, 2023.

[216] S. Zhao, J. Song, and S. Ermon. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017.

[217] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555 (7697):487–492, 2018.

[218] S. C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27:107–126, 1998.

[219] E. Zisselman and A. Tamar. Deep residual flow for out of distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13994–14003, 2020.

[220] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *2011 international conference on computer vision*, pages 479–486. IEEE, 2011.

Part II

Papers

Contents