Injectivity and Locality: Robust Deep Learning for Bayesian Imaging

---

Inauguraldissertation
zur
Erlangung der Würde eines Doktors der Philosophie
vorgelegt der
Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel

von

AmirEhsan Khorashadizadeh

Basel, 2024

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät auf Antrag von

Erstbetreuer: Prof. Dr. Ivan Dokmanić

Zweitbetreuer: Prof. Dr. Volker Roth

Externer Experte: Prof. Dr. Ender Konukoglu

Basel, 17.09.2024

Prof. Dr. Marcel Mayor, Dekan

Abstract

Imaging inverse problems are crucial in exploring and understanding various phenomena in our universe. Astronomers decode light signals from distant galaxies, physicians use imaging to reveal the internal body structures for clinical analysis, and geophysicists process seismic waves to model the Earth's interior. Each case involves reconstructing images of hidden objects from observed data. Recently, data-driven methods based on deep learning have shown great success in solving various imaging inverse problems resulting in high-quality and fast image reconstruction with fewer noisy observations. One major concern when solving inverse problems with deep learning is generalization; we expect the deep neural network to perform well on data other than training samples. Lack of generalization may lead to unstable reconstructions and wrong interpretations which is problematic, particularly for medical applications.

In the first part of this thesis, we build deep-learning architectures based on implicit neural representation. We show that these coordinate-based reconstruction pipelines including MLPatch, Glimpse, and FunkNN for various imaging modalities like image denoising, super-resolution, computed tomography, and magnetic resonance imaging can produce high-quality reconstructions with strong generalization.

While deep learning models with strong generalization can improve the reliability of reconstructions and downstream interpretations, the estimated images can significantly deviate from the true image. Moreover, due to the noise and ill-posedness of the forward operator, there may exist many images that align with our observations, each resulting in a different scientific interpretation. One way to address this is to learn a distribution of possible reconstructions instead of computing a single estimate. This strategy can also help us evaluate an uncertainty map to pinpoint the regions of the recovered image estimated with lower confidence.

To this end, in the second part of this thesis, we develop Bayesian frameworks based on injective neural networks to learn the distribution of reconstructions for solving ill-posed inverse problems. We show that our Bayesian architectures can generate multiple high-quality reconstructions and evaluate physically meaningful uncertainty estimates for various imaging problems including inverse scattering and computed tomography.

To my parents and my wife for their love, support and encouragement

Acknowledgements

I am deeply grateful to my wife for her unwavering support and encouragement throughout my PhD journey. Her belief in me kept me moving forward during challenging times. I am truly thankful to my mother, whose unconditional love has been an incentive, from my earliest school days through the completion of my PhD. I am grateful to my late father, who was my first mathematics teacher in childhood. His encouraging words have always motivated me to keep moving forward. Without their encouragement, it would have been hard to persevere and continue this journey.

I spent four fantastic years at the SADA group. I feel incredibly fortunate to have had Ivan as my advisor. Not only is he an exceptional scientist, but he is also a kind and generous mentor. He allowed me the freedom to explore my ideas while providing invaluable guidance to bring them to fruition. I would also like to thank Professor Volker Roth for his feedback and comments during my PhD. Additionally, I extend my sincere thanks to my external reviewer, Professor Ender Konukoglu, for his interest in my work and the time and effort he dedicated to reviewing this thesis. Furthermore, I feel fortunate to have had the opportunity to collaborate with Professor Maarten de Hoop during our wave-based imaging projects.

I am fortunate to have been surrounded by my dear friends at the University of Basel. My PhD began with a collaboration with Konik Kothari which was an educational experience in both mathematics and programming, laying a solid foundation for my future research in machine learning. Special thanks go (in no particular order) to my friends at the University of Basel, Valentin Debarnot, Vinith Kishore, Tianlin Liu, Anadi Chaman, Cheng Shi, Jonas Linkerhägner, Fabian Kruse, Alexandra Spitzer, Lorenzo Baldassari, Jonathan Aellen, Fabricio Arend Torres, Manuel Jahn, Monika Nagy-Huber, Shijian Xu, and Marcello Negri who made my experience in Basel truly enjoyable. Our trip to Rovinj after my defense always remains a fond memory. A special thanks to my cousin and friend, Vahid Khorashadizadeh, whose support over the years—from high school to co-authoring a paper together—has been invaluable. I must also acknowledge Ali Aghababaei-Harandi and Sepehr Eskandari, whose friendship and collaboration have been with me since our undergraduate days.

Lastly, I am very thankful for the support from the University of Basel's promotion of young talents, which awarded me a grant to spend nine months at University College

London (UCL) as a visiting researcher. This grant enabled me to collaborate with Professor Jason McEwen and his research group. I am particularly grateful to Tobias Liaudat and the rest of the group, I will never forget the joy of our trip to Khania in May 2024 for the Cosmo21 conference—it was a perfect blend of science and friendship.

Contents

Chapter 1

Introduction

In everyday life, we often encounter situations where we need to deduce hidden information from indirect observations. As an intuitive example, imagine you receive a sealed package in the mail. You are curious about what is inside; you try to gather clues like weight, shape, size, and sound. Using these observations, you start to make guesses about what might be inside the package. This process of inferring hidden information from observations is analogous to solving an inverse problem.

Inverse problems are prevalent in various applications including medicine [1], material science [2], remote sensing [3] and cosmology [4]. In these problems, the goal is to reconstruct an unknown signal, image, or 3D volume $\mathbf{f} \in \mathcal{F}$ from noisy measurements $\mathbf{q} \in \mathcal{Q}$ where the measurements are obtained through the imaging process modeled as follows,

$$\mathbf{q} = \mathbf{A}\mathbf{f} + \mathbf{n}, \tag{1.1}$$

where $\mathbf{A} : \mathcal{F} \to \mathcal{Q}$ is the forward operator modeling the physics of the imaging system that probes the object $\mathbf{f}$ and we consider additive noise $\mathbf{n}$ for simplicity. It is worth noting that more complicated scenarios like stochastic forward operators or non-additive noise can also be modeled. This general framework underpins numerous inverse problems in imaging including,

- **Image denoising:** Reconstructing a clean image from a noisy one. This is considered the simplest inverse problem with an identity forward operator [5]. A practical example of image denoising is low-dose computed tomography.
- **Image super-resolution:** Reconstructing a high-resolution image from a low-resolution one. This technique is used because acquiring high-resolution images can be costly for imaging systems [6].
- **Sparse- and limited-view computed tomography (CT):** Reconstructing the object's internal structure from measurements of wave attenuation. The object is illuminated by X-rays from multiple angles [7].

- **Magnetic resonance imaging (MRI):** Reconstructing the internal structure of an object from signals generated by the nuclei in the body's tissues. This medical imaging technique uses magnetic fields and radio waves [8].
- **Inverse scattering:** Reconstructing the object's wave speed from measurements of scattered waves. The object is illuminated by electromagnetic or X-ray waves from different angles [9].

These problems are often ill-posed, meaning that many images might be compatible with the given measurements, though most of these solutions are not plausible. For example, in image super-resolution, many high-resolution images may result in the same low-resolution image.

Another concern is the stability of the inverse problem. The measurements $\mathbf{q}$ are often contaminated by noise $\mathbf{n}$. Unstable forward operators can result in significantly different images from the modified measurements, which is undesirable. Ill-posedness and poor instability together make inversion challenging. To tackle these issues, *regularization* techniques can be used to constrain our search space to target image solutions and can be formulated as follows,

$$\mathbf{f}^* = \underset{\mathbf{f}}{\operatorname{argmin}}\, \mathcal{D}(\mathbf{q}, \mathbf{A}\mathbf{f}) + \lambda \mathcal{R}(\mathbf{f}). \tag{1.2}$$

Here, $\mathcal{D}$ specifies the discrepancy function between the measurements $\mathbf{q}$ and the application of the forward operator $\mathbf{A}$ to the estimated image $\mathbf{f}$. The choice of $\mathcal{D}$ depends on the noise model $\mathbf{n}$; for instance, Gaussian noise corresponds to the L_2 norm for $\mathcal{D}$. In (1.2), $\mathcal{R}$ is the regularization operator that incorporates our prior knowledge of the target image $\mathbf{f}$, and λ is a pre-specified parameter that adjusts the amount of regularization. When $\lambda = 0$, there is no regularization. The choice of regularization can significantly influence the quality of the reconstruction.

Traditional regularization methods use hand-crafted regularizers. Total variation (TV) regularization [10] leverages the smoothness of natural images, resulting in edge-preserving reconstruction algorithms. While these methods perform well on images with simple structures, hand-crafted priors often lead to poor reconstructions for complex natural images, which are common in practice.

As of recently, methods with data-driven priors based on deep neural networks significantly outperform traditional solvers for ill-posed inverse problems [11]. Deep neural networks have shown promising results across various imaging problems including image denoising [12], image super-resolution [13], CT [14], MRI [15] and inverse scattering [16]. Despite their impressive performance, the reliability of deep learning architectures remains a significant concern in practical applications [17]. One critical issue is generalization, while trained on specific datasets, neural networks are expected to generalize well to new, unseen data [18]. Poor generalization can be problematic, especially in sensitive domains like medical imaging and autonomous driving.

An effective strategy to design reliable neural networks is to endow them with capability for *uncertainty quantification*; the model needs to provide an uncertainty map to pinpoint the regions of the recovered image estimated with lower confidence.

This thesis has two main parts: in Chapters 2, 3, and 4, we design our coordinate-based image recovery pipelines for building neural networks with strong generalization. Chapters 5, 6, 7 and 8 leverage deep generative models for uncertainty quantification.

- **MLPatch: Scalable Coordinate-based Image Reconstruction**
  Convolutional neural networks (CNNs) have become the de facto standard machine learning approach for solving imaging inverse problems. One limitation of CNNs is that the memory required for training scales unfavorably with image resolution, making them unsuitable for high-resolution problems. Another issue is that they tend to overfit the large-scale image features; preventing this requires elaborate designs that automatically adjust the receptive field. In this chapter, we design a local processing network termed MLPatch, which recovers the image intensity at a target pixel by processing a small neighborhood of the pixel in the observed image using a simple multi-layer perception (MLP). With comparable performance to state-of-the-art CNNs, MLPatch achieves excellent **generalization to out-of-distribution data** and memory usage independent of image resolution. Remarkably, a mere 3GB of memory suffices to train on 1024×1024; standard CNNs require more than 80GB. Moreover, MLPatch can effectively recover the image at *any* continuous coordinate. Finally, we generalize this framework to non-localized inverse problems by employing MLPatch denoiser as a prior in a plug-and-play framework. Our experiments show that this framework can achieve excellent performance on non-localized problems like image in-painting, magnetic resonance imaging (MRI), and radio interferometry.

- **GLIMPSE: Generalized Local Imaging with MLPs**
  In this chapter which is based on our preprint paper [19], we generalize our local coordinated-based reconstruction pipeline to computed tomography (CT) where the notion of locality must be adapted to the non-cartesian geometry of the CT sinograms. We introduce GLIMPSE, a neural network that reconstructs a pixel value by processing only the measurements in the sinogram associated with the neighborhood of the pixel with a simple multi-layer perception (MLP). While achieving performance comparable to or better than successful CNNs like U-Net on in-distribution test data, GLIMPSE significantly outperforms them on OOD samples while maintaining a memory footprint almost independent of image resolution; 5GB memory suffices to train on 1024×1024 images, with each epoch requires 420 seconds. Because we built GLIMPSE to be fully differentiable it can also be used as a plug-in component of arbitrary deep learning architectures, enabling feats such as correction of miscalibrated projection orientations.

- **FunkNN: Neural Interpolation for Functional Generation**

In this chapter which is based on our published paper [20], we showcase an important feature of the developed coordinate-based framework, continuous image representation for learning the distribution of functions, and answer the following question: can we build continuous generative models that generalize across scales, can be evaluated at any coordinate, admit calculation of exact derivatives, and are conceptually simple? Existing MLP-based architectures generate worse samples than the grid-based generators with favorable convolutional inductive biases. Models that focus on generating images at different scales do better but employ complex architectures not designed for continuous evaluation of images and derivatives. We take a signal-processing perspective and treat continuous image generation as interpolation from samples. Indeed, correctly sampled discrete images contain all information about the low spatial frequencies. The question is then how to extrapolate the spectrum in a data-driven way while meeting the above design criteria. Our answer is FunkNN—a new convolutional network that learns how to reconstruct continuous images at arbitrary coordinates and can be applied to any image dataset. Combined with a discrete generative model it becomes a functional generator that can act as a prior in continuous ill-posed inverse problems. We show that FunkNN generates high-quality continuous images and exhibits strong out-of-distribution performance thanks to its patch-based design. We further showcase its performance in several stylized inverse problems with exact spatial derivatives.

- **Trumpets: Injective Flows for Inference and Inverse Problems**
  Starting with this chapter, we focus on the Bayesian approach for solving ill-posed inverse problems using deep generative models enabling uncertainty quantification. Deep learning models for computational imaging often regress a single reconstructed image. In practice, however, ill-posedness, nonlinearity, model mismatch, and noise often conspire to make such *point estimates* misleading or insufficient. The Bayesian approach models images and (noisy) measurements as jointly distributed random vectors and aims to approximate the posterior distribution of unknowns. In this chapter which is based on our published paper [21], we design a Bayesian framework, injective generative models called TRUMPET that generalize invertible normalizing flows. The injective generators progressively increase dimension from a low-dimensional latent space. We demonstrate that TRUMPETs can be trained orders of magnitudes faster than standard flows while yielding samples of comparable or better quality. They retain many of the advantages of the standard flows such as training based on maximum likelihood and a fast, exact inverse of the generator. Since TRUMPETs are injective and have fast inverses, they can be effectively used for downstream Bayesian inference. To wit, we use TRUMPET priors for solving ill-posed inverse problems, outperforming competitive baselines in terms of reconstruction quality and speed. We then develop an efficient method for posterior characterization and uncertainty quantification with TRUMPETs by taking advantage of the low-dimensional latent space.

- **Deep Injective Prior for Inverse Scattering**
  This chapter applies our unsupervised algorithm based on Trumpets to solve inverse scattering. In electromagnetic inverse scattering, the goal is to reconstruct object permittivity using scattered waves. While deep learning has shown promise as an alternative to iterative solvers, it is primarily used in supervised frameworks that are sensitive to distribution drift of the scattered fields, common in practice. Moreover, these methods typically provide a single estimate of the permittivity pattern, which may be inadequate or misleading due to noise and the ill-posedness of the problem. In this chapter which is based on our published paper [22], we develop a data-driven framework designed for inverse scattering based on Trumpets. Unlike supervised methods that necessitate both scattered fields and target permittivities, our method only requires the target permittivities for training; it can then be used with any experimental setup. We also introduce a Bayesian framework for approximating the posterior distribution of the target permittivity, enabling multiple estimates and uncertainty quantification. Extensive experiments with synthetic and experimental data demonstrate that our framework outperforms traditional iterative solvers, particularly for strong scatterers, while achieving comparable reconstruction quality to state-of-the-art supervised learning methods like the U-Net.

- **Conditional Injective Flows for Bayesian Imaging**
  The developed Bayesian frameworks in Chapters 5 and 6 involve an iterative process to learn the posterior distribution for each new measurement which might be slow. In this chapter which is based on our published paper [23], we design C-Trumpets—conditional injective flows specifically designed for imaging problems for fast and efficient posterior learning. Injectivity reduces memory footprint and training time while low-dimensional latent space together with architectural innovations like fixed-volume-change layers and skip-connection revnet layers, C-Trumpets outperform regular conditional flow models on a variety of imaging and image restoration tasks, including limited-view CT and nonlinear inverse scattering, with a lower compute and memory budget. C-Trumpets enable fast approximation of point estimates like MMSE or MAP as well as physically meaningful uncertainty quantification.

- **Deep Variational Inverse Scattering**
  Despite good performance for uncertainty quantification, C-Trumpets are unable to generate high-quality posterior samples and MMSE estimates, significantly worse than that of successful point estimators like U-Net. This chapter addresses this issue and applies our method to the inverse scattering problem. In this chapter which is based on our published paper [24], we design U-Flow, a Bayesian U-Net based on conditional normalizing flows, which generates high-quality posterior samples and estimates physically meaningful uncertainty. We show that U-Flow significantly outperforms C-Trumpets in terms of posterior sample quality while having comparable performance with the U-Net in point estimation.

Chapter 2

Scalable Local Image Reconstruction with Implicit Neural Representation

Deep learning has been, for a considerable time, the method of choice for solving major imaging problems. Convolutional neural networks like U-Net [25] have shown remarkable performance across diverse tasks including computed tomography (CT) [14], magnetic resonance imaging (MRI) [15], photoacoustic imaging [26] and inverse scattering [16]. This success is primarily attributed to its multiscale architecture with a large receptive field that extracts features from the input image at different scales [27].

Despite their strong performance on low-dimensional 2D imaging problems, deep learning architectures can become computationally expensive for high-dimensional images. This inefficiency arises because current deep neural networks reconstruct the entire image simultaneously, necessitating substantial memory for back-propagation during training. Additionally, interpreting models and analyzing reconstructions can be challenging. Simplified neural architectures can enhance our understanding of reconstruction mechanisms, enabling the design of robust models with strong generalization.

Recently, implicit neural representations (INRs) [28–30] have emerged as a promising tool for representing continuous signals, images, and 3D volumes. Unlike most existing deep learning models that treat signals as discrete arrays, INRs map coordinates to signal values using a deep neural network, typically a multi-layer perceptron (MLP), resulting in a *continuous* signal representation. INRs have been widely applied to various tasks, including solving partial differential equations (PDEs) [20, 28, 31], 3D shape modeling [32–34], and image super-resolution [20, 35]. One of the most well-known applications of INRs is 3D scene representation using neural radiance fields (NeRF) [36], which has received significant attention in recent years. The recent work [37] endows INRs with uncertainty quantification for 3D reconstruction from sparse and corrupted data.

INRs have several advantages over standard deep learning models. Rather than

representing signals at a single resolution, INRs can conveniently interpolate signals in a continuous space. This capability is particularly interesting because we can conveniently adjust the required memory making INRs well-suited for high-dimensional 3D reconstructions [38, 39].

In this chapter, we leverage INRs to build a scalable coordinate-based reconstruction pipeline for solving imaging inverse problems with local forward operators. In tasks such as image denoising and super-resolution, the image intensity at a specific pixel is primarily influenced by the observed image in the pixel's neighborhood. Our proposed model, termed MLPatch, recovers the image intensity at each pixel *individually*, using local information extracted from the input image around that pixel. This information is processed by a neural network to determine the image intensity at the target pixel, enabling image reconstruction at any resolution or arbitrary continuous coordinate.

MLPatch leverages the benefits of INRs through its coordinate-based synthesis pipeline. This allows training on mini-batches of both objects and pixels, resulting in resolution-agnostic memory usage. Consequently, MLPatch can handle high-resolution images and 3D shapes with minimal memory requirements. Remarkably, it requires only 3GB of memory to train on 1024×1024 images.

MLPatch facilitates model interpretation and reconstruction analysis. Thanks to its coordinate-based synthesis pipeline, MLPatch is a shift and rotation equivariant model, which enhances its ability to generalize across out-of-distribution data. Additionally, we propose a novel adaptive patch geometry framework to learn the position of the relevant features in the input image for each pixel, providing valuable insights for image analysis.

We then show how to extend the applicability of MLPatch to non-localized inverse problems, benefiting from its robust generalization and computational efficiency. We pre-train MLPatch as a denoiser and incorporate it as a denoising prior in a plug-and-play (PnP) framework [40]. We show that the resulting unsupervised framework effectively solves diverse imaging problems, including image in-painting, MRI and radio interferometry imaging. Remarkably, the proposed framework inherits the strong generalization capability of MLPatch on OOD data. At the same time, its coordinate-based synthesis pipeline allows a pre-trained MLPatch model, initially trained on lower dimension images, to handle inverse problems at arbitrary dimension with modest memory usage.

2.1 Background and related works

We consider imaging inverse problems discussed in (1.1) where we assume that $\mathbf{n}$ is additive white Gaussian noise (AWGN), $\mathbf{n} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$. With this assumption, (1.2) can be written as,

$$\mathbf{f}^* = \underset{\mathbf{f}}{\operatorname{argmin}} \frac{1}{2\sigma^2} \|\mathbf{q} - \mathbf{A}\mathbf{f}\|_2^2 + R(\mathbf{f}), \tag{2.1}$$

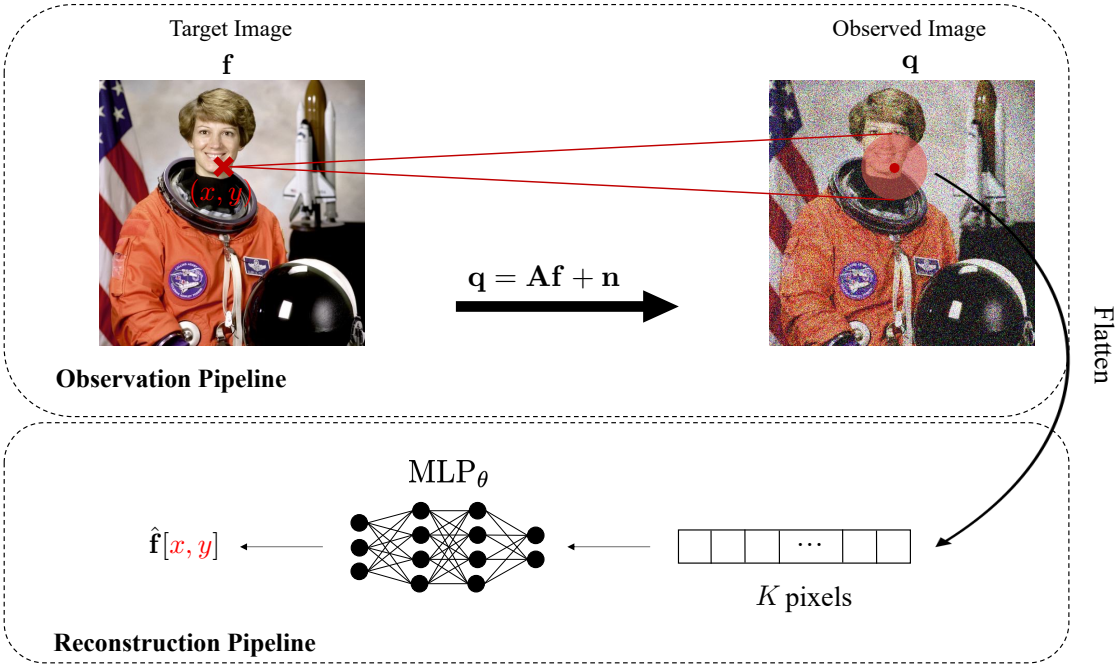


Figure 2.1: MLPatch; a small MLP processes the patch extracted from the observed image around the given pixel (x, y). MLPatch has strong generalization on OOD data and requires a small memory when training on high-resolution images like 1024×1024 thanks to its coordinate-based synthesis pipeline.

where $R(\cdot)$ represents the regularization term encompassing prior knowledge about the image structure. While traditional hand-crafted priors such as total-variation (TV) [10] can yield good reconstructions, learning-based priors have brought about substantial improvements [21, 41, 42].

2.1.1 local forward operators

In many inverse problems, the forward operator is localized; an image pixel primarily relates to its neighboring pixels in the observed image. Here the assumption is that the observed data is shaped like an image with the same coordinate system as the target. This local forward operator suggests an opportunity to design efficient deep-learning architectures. In the following, we discuss two inverse problems characterized by this localization property.

Image denoising

Image denoising is the most straightforward inverse problem, where its forward operator $\mathbf{A}$ in (1.1) is the identity matrix; each pixel in the clean image is solely connected to the

corresponding pixel in the noisy image, assuming uncorrelated noise and natural images. Despite this simplicity, the presence of noise prompts the incorporation of 'contextual' information from neighboring noisy pixels. In principle, the required local neighborhood size depends on the noise level and the prior distribution; stronger noises often necessitate larger receptive fields for natural images [43].

Over the last decade, various deep learning approaches have achieved state-of-the-art (SOTA) performance in image denoising. Some early approaches segmented the image into patches and employed an MLP network to denoise each patch individually [44]. However, the optimal patch size for high-quality reconstructions depends on the noise level, which makes it challenging to estimate. Zhang et al. [12] introduced DnCNN, an image-to-image CNN for image denoising where the network depth is adjusted based on the required receptive field size suggested by traditional methods [45]. To expand the CNN's receptive field and exploit more contextual information, the authors of [46] used dilated convolutions. Concurrently, the U-Net [25] has been the backbone model for state-of-the-art image denoising architectures [47–49], in particular, the DRU-Net [50] achieved remarkable performance by integrating residual blocks [51] into the U-Net architecture. Despite their success, these CNNs lack a systematic procedure for automatic adjustment of the required receptive field size which can lead to overfitting the global features and poor generalization. Moreover, their required memory scales with image resolution making them computationally expensive for high-resolution image reconstruction.

Dark matter mapping in cosmology

In addition to denoising, we will consider a more complicated inverse problem, dark matter mapping [4], which is of utmost importance for our understanding of weak gravitational lensing [52] in cosmology. We will study the convergence that traces the projected mass distribution of the Universe, effectively providing a map of dark matter. The goal is to estimate the convergence field κ, the target image, from a related observable, the shear field γ. The convergence is associated with the magnification of observed galaxies, while the shear corresponds to the image's stretching due to ellipticity changes. The two fields are related as

$$\gamma = \mathbf{A}\kappa + \mathbf{n}, \tag{2.2}$$

where $\mathbf{n}$ is the shear field noise, and the forward operator $\mathbf{A}$ is a convolutional filter (a Fourier multiplier) with the 2D Fourier transform given by[1]

$$\mathbf{D}_{k_x,k_y} = \frac{k_x^2 - k_y^2 + 2ik_xk_y}{k_x^2 + k_y^2}, \quad \text{for } k_x^2 + k_y^2 \neq 0. \tag{2.3}$$

Clearly, this inverse problem has a well-posed forward operator and we can compute the inverse of the forward operator $\mathbf{A}^{-1}$

[1]This is known as the weak lensing planar forward model.

A simple approach to recover the convergence field κ from the shear field γ is the Kaiser-Squires (KS) method [4], the current standard in cosmology. The method consists of directly applying $\mathbf{A}^{-1}$, the inverse of the forward operator, to the shear field, with a subsequent Gaussian smoothing with an ad hoc setting of the smoothing scale. We consider the naive KS inversion, without the Gaussian smoothing step, as follows,

$$\kappa_{\mathrm{KS}} = \mathbf{A}^{-1}\mathbf{A}\gamma = \kappa + \mathbf{A}^{-1}\mathbf{A}\mathbf{n} = \kappa + \tilde{\mathbf{n}}, \tag{2.4}$$

where $\tilde{\mathbf{n}} = \mathbf{A}^{-1}\mathbf{A}\mathbf{n}$ is colored noise. The recovery of the convergence field from the naive inverted KS image is thus a denoising problem with colored noise and can be interpreted as an inverse problem with a *local* forward operator. In Section 2.3.2, we later exploit this representation to solve the problem using MLPatch.

Several methods have been proposed to incorporate prior information for dark matter mapping, including Gaussian priors [53], sparsity [54–57], and more recently deep learning, including CNNs [58] and score matching techniques [59].

2.1.2 Non-local forward operators

In general, inverse problems might have non-local forward operators where the image intensity at a given pixel could be related to features globally distributed over the observed image. One important example of this family is magnetic resonance imaging (MRI) [8, 60, 61] where the forward operator is given as follows,

$$\mathbf{A} = \mathbf{M}_{\Omega}\mathbf{S}\mathbf{F}, \tag{2.5}$$

where M_Ω is a mask indicating the sampled Fourier points, S corresponds to the sensitivity maps of each coil and $\mathbf{F}$ is the Fourier transform. The locality of the forward operator is lost with the Fourier transform.

A similar problem is radio interferometric imaging where we also acquire samples of the Fourier transform of the image along curves [62–64]. In radio interferometric imaging, aperture synthesis techniques are used to acquire specific Fourier measurements giving an incomplete coverage of the Fourier plane. The inverse problem consists of recovering the entire intensity image from Fourier measurements. In this case, the forward operator in a simplified form writes

$$\mathbf{A} = \mathbf{M}_{\mathrm{RI}}\mathbf{F} \tag{2.6}$$

where $\mathbf{M}_{\mathrm{RI}}$ is a Fourier mask indicating the Fourier coverage of the observations. We have omitted gridding and degridding operations in between other corrections from the forward operator for the sake of simplicity; a more detailed description of the operator can be found in [63, §3.2].

2.1.3 Supervised vs unsupervised learning for inverse problems

A standard approach for solving imaging inverse problems is regression using CNNs in a *supervised* learning framework. Supervised learning enables fast image recovery from observations [14, 16, 23, 27, 65]. However, it necessitates paired training data comprising measurements and corresponding target images which is hard to obtain in many real scientific problems. Additionally, the related models require re-training even for small changes in the forward model.

To address this issue, unsupervised methods learn a data-driven prior from target images. Deep generative models have been extensively used for solving imaging inverse problems where a pre-trained image generator is used as the prior [21, 22, 66–69]. However, training deep generative models often requires large-scale datasets and substantial computational resources [70, 71].

An alternative is to use a Gaussian denoiser as a prior in a plug-and-play (PnP) framework [40, 72, 73]. PnP algorithms achieved SOTA performance across a variety of image reconstruction tasks including image super-resolution and deblurring [50, 74], computed tomography (CT) [75] and magnetic resonance imaging (MRI) [76, 77]. Since PnP algorithms decouple the likelihood from the image prior they enable the use of pre-trained denoisers. Compared to generative priors they are often cheaper in both training and inference. Recently, the authors of [78] leveraged diffusion models [71, 79] as a pre-trained denoiser in PnP framework and achieved state-of-the-art performance for various image restoration tasks.

2.2 MLPatch: a local coordinate-based network

Our objective is to reconstruct the target image $\mathbf{f} \in \mathbb{R}^{N\times N\times C}$ from the observed image $\mathbf{q} \in \mathbb{R}^{M\times M\times C}$, as described in the forward equation (1.1), where C is the channel dimension, N and M denote the resolution of the target and observed images. The resolutions of $\mathbf{f}$ and $\mathbf{q}$ may be different but we assume that they share the same coordinate system and semantics; thus $\mathbf{f}[x, y]$ and $\mathbf{q}[x, y]$ refer to the same continuous location in both images. Example problems where this is meaningful are denoising and deblurring. More generally, for many linear inverse problems, including computed tomography, it will hold after applying the adjoint of the forward operator.

We then consider a local forward operator $\mathbf{A}$ where the intensity $\mathbf{f}[x, y]$ at pixel $(x, y) \in \mathbb{R}^2$ relates only to a small neighborhood of (x, y) in $\mathbf{q}$. We design an efficient deep neural network, MLPatch, based on this locality hypothesis.

To recover the image at the target pixel (x, y) *individually*, we extract a local patch around (x, y) from the observed image $\mathbf{q}$ containing K pixels. We denote the (flattened) extracted patch by $\mathbf{p}_{x,y} \in \mathbb{R}^{K\times C}$. As illustrated in Figure 2.1, we process $\mathbf{p}_{x,y}$ using a

$\text{MLP}_\theta : \mathbb{R}^{K\times C} \to \mathbb{R}^C$ parameterized by θ to approximate the image intensity $\mathbf{f}[x, y]$,

$$\hat{\mathbf{f}}[x, y] = \text{MLP}_\theta(\mathbf{p}_{x,y}). \tag{2.7}$$

This coordinate-based image representation enables image recovery at any continuous coordinate with small memory. We call our framework MLPatch. In the following sections, we provide further details regarding the MLPatch architecture and introduce the pre-processing filter, patch geometry, and training strategy.

2.2.1 Patch geometry and equivariance

A straightforward method to extract a patch is to directly select the pixels adjacent to the target pixel (x, y). This however has several drawbacks: it requires a manual selection of the optimal patch size, akin to traditional methods, which is hard to estimate; it only permits image evaluation at on-grid pixels resulting in a single-resolution image reconstruction.

Learnable patch geometry

To enable image reconstruction at arbitrary continuous coordinates, we define a patch with learnable geometry centered around the target coordinate (x, y) by first defining a set of *learnable* coordinate offsets,

$$\Delta I = \left[(x_n^\Delta, y_{n'}^\Delta)\right]_{n,n'=1}^K. \tag{2.8}$$

By learning the optimal coordinate offsets ΔI we can control the scale of the receptive field of the patch. We then extract the patch by evaluating $\mathbf{q}$ at coordinates $I_{(x,y)} = (x, y) + \Delta I$ to get

$$\mathbf{p}_{x,y} = \mathbf{q}[I_{(x,y)}]. \tag{2.9}$$

We emphasize that the coordinates $I_{(x,y)}$ need not be aligned with the pixel grid; we use differentiable bicubic interpolation to enable off-the-grid evaluation. Unlike most other work using patches, we initialize ΔI to a circular geometry; this results in better rotation invariance as discussed in the following section.

2.2.2 Coordinate-conditioned patch geometry

While the learnable patch geometry introduced in (2.8) can provide additional flexibility to the network, the learned patch geometry is fixed for all the pixels in the image. In practice, the location of the relevant information changes for different pixels. To enable coordinate-conditioned patch geometry (CCPG), we use another neural network

$\text{CCPG}_\psi : \mathbb{R}^{K\times C} \to \mathbb{R}^{2K}$ which takes a local patch around pixel (x, y) and estimates the position of the coordinates inside the patch as follows,

$$\Delta I_{CCPG}(x, y) = \text{CCPG}_\psi(p_{x,y}) \tag{2.10}$$

We then evaluate $\mathbf{q}$ at $(x, y) + \Delta I_{CCPG}(x, y)$. The learned patch geometry $\Delta I_{CCPG}(x, y)$ can localize the position of the relevant information for image recovery at pixel (x, y) for corrupted image $\mathbf{q}$. We can improve performance by repeating this process T times:

$$\Delta I^{(i)}_{CCPG}(x, y) = \text{CCPG}^{(i)}_\psi(p^{(i-1)}_{x,y}) \tag{2.11}$$

$$p^{(i)}_{x,y} = \mathbf{q}[(x, y) + \Delta I^{(i)}_{CCPG}(x, y)] \tag{2.12}$$

where in each iteration $1 \le i \le T$, we feed noisy image $\mathbf{q}$ evaluated at the estimated patch position in the previous step $p^{(i-1)}_{x,y}$ to a separate neural network $\text{CCPG}^{(i)}_\psi$. As we will show in our experiments in Section 2.3.4, the learned patch geometry indeed improves over iterations.

2.2.3 Equivariance to shifts and rotations

In many imaging and pattern recognition tasks, it is desirable to work with functions that are invariant or equivariant to shifts and rotations. Building such invariances into CNNs, typically via group representation theory [80, 81] improves generalization both in and out of distribution [82, 83].

It is easy to see that MLPatch is shift equivariant by design. Indeed, the patch at position $\mathbf{x} + \Delta\mathbf{x}$ in an image shifted by $\Delta\mathbf{x}$ is exactly the same as the patch at position $\mathbf{x}$ in a non-shifted image; the MLP thus receives the same input. This stands in contrast to most mainstream multi-scale CNNs which are surprisingly sensitive even to 1-pixel shifts, although equivariance can be restored by a careful design of pooling and downsampling layers [84, 85].

Rotation equivariance is precluded for square patches, except for angles that are multiples of $\frac{\pi}{2}$. With circular patches, when the image is rotated the MLP receives the same input only in a different order up to interpolation error. Rotation equivariance could be implemented by applying the MLP to rotation invariant features [86–89] extracted from the circular patch or by averaging the output over rotations. Even without this, however, MLPatch is approximately rotation equivariant because natural images are approximately *locally* rotation invariant and the patch geometry is initialized as circular; cf. Section 2.3.5 for an empirical validation of this claim.

2.2.4 Noise suppression filter

The observed image $\mathbf{q}$ is often corrupted with significant noise making the inversion challenging for the MLP_θ in (2.7). To address this, we apply pre-processing filters

$\mathbf{h} \in \mathbb{R}^{s \times s \times T}$ to the observed image before patch extraction,

$$\tilde{\mathbf{q}} = \mathbf{q} \star \mathbf{h}, \tag{2.13}$$

where $\star$ denotes the convolution operator, s is the filter size and T is the number of filters. This process is reminiscent of filtering techniques commonly used in computed tomography imaging [90]. We can also concatenate the filtered and noisy images along the channel dimension.

To avoid manual adjustment of the receptive field, we apply the filter in the frequency domain,

$$\tilde{\mathbf{q}} = \mathbf{F}^{-1}\mathbf{H}\mathbf{F}\mathbf{q}, \tag{2.14}$$

where $\mathbf{F}$ and $\mathbf{F}^{-1}$ are the forward and inverse 2D Fourier transforms, and the filter $\mathbf{H}$ is treated as a learnable parameter derived from the data. The filter $\mathbf{H}$ can have large support in image space, capturing global information, in contrast to the local convolutional operator in (2.13). This large filter support can allow the MLP to overfit global features and compromise model generalization. To mitigate this issue, we initialize the filter $\mathbf{H}$ with an all-one Fourier representation, which corresponds to the smallest support in pixel space. During training, the filter expands its support based on the image resolution, noise level, and data distribution while keeping the filter support small. To further control support size, we could add a regularization penalty based on the filter support but in subsequent experiments presented in Appendix 2.6.2, we noticed that the learned filter has small support even without any regularization.

2.2.5 Resolution-agnostic memory usage in training

For simplicity, we write the entire MLPatch pipeline as $\hat{\mathbf{f}}(\mathbf{x}) = \text{MLPatch}_\phi(\mathbf{x}, \mathbf{q})$ where $\mathbf{x} = (x, y)$ is the target pixel. The model approximates the image intensity $\hat{\mathbf{f}}(\mathbf{x})$ from the observed image $\mathbf{q}$. We denote all the trainable parameters of MLPatch by ϕ. This includes the MLP weights θ, pre-processing filter parameters $\mathbf{h}$, and CCPG weights ψ. We consider a set of training data $\{(\mathbf{q}_i, \mathbf{f}_i)\}_{i=1}^{L}$ from observed and target images. We optimize the MLPatch parameters ϕ using a gradient-based optimizer to solve,

$$\phi^\star = \underset{\phi}{\text{argmin}} \sum_{i=1}^{L} \sum_{j=1}^{N^2} |\mathbf{f}_i(\mathbf{x}_j) - \text{MLPatch}_\phi(\mathbf{x}_j, \mathbf{q}_i)|. \tag{2.15}$$

At inference time, we can recover the image at any target pixel $\mathbf{x}$ from the observed image $\mathbf{q}_{\text{test}}$ as $\hat{\mathbf{f}}_{\text{test}}(\mathbf{x}) = \text{MLPatch}_{\phi^\star}(\mathbf{x}, \mathbf{q}_{\text{test}})$. From (2.15), it is clear that MLPatch can be trained on mini-batches of both objects *and* pixels, leading to nearly constant memory usage during training and inference. This flexibility also allows us to train MLPatch on datasets with images in various resolutions.

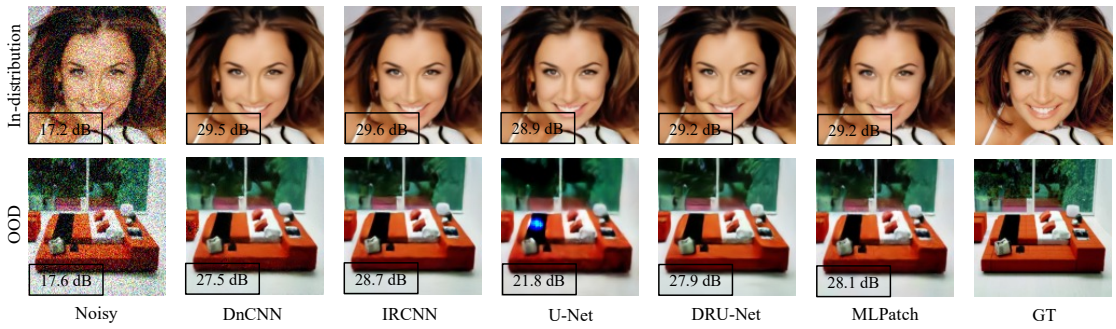


Figure 2.2: Performance comparison between different models on image denoising ($\sigma = 0.15$) in resolution 128×128.

2.3 MLPatch for inverse problems with local forward operators

We assess the performance of MLPatch on inverse problems with local forward operators including image denoising and dark matter mapping in cosmology. As baselines, we consider BM3D [5] and state-of-the-art CNNs including DnCNN [12], IRCNN [46], U-Net [25] and DRU-Net [50]. Further details regarding the MLPatch architecture and training details are available in Appendix 2.6.1.

2.3.1 Image denoising

We consider zero-mean Gaussian noise $\mathbf{n}$ with $\sigma = 0.15$ and normalize images between 0 and 1. We train different models on 29900 samples from the CelebA-HQ [91] dataset. Model performance is then assessed on 100 in-distribution test samples from CelebA-HQ, as well as 300 OOD samples from the LSUN-bedroom dataset [92].

Figure 2.2 illustrates a visual comparison of reconstructions obtained from various models on both in-distribution and OOD data on image denoising. It shows that MLPatch, which uses a single MLP, can achieve comparable performance on image denoising with SOTA CNNs on both in-distribution and OOD data and significantly outperform U-Net on OOD data. The U-Net's poor generalization can be attributed to its large receptive field due to multiscale architecture. As expected, the locality of MLPatch prevents the MLP from overfitting to global features resulting in a strong generalization out of distribution. Table 2.1 presents the quantitative results in terms of PSNR and structural similarity index measure (SSIM) [93]. It shows the importance of the pre-processing filters, learnable patch geometry, and CCPG module.

Table 2.1: Quantitative comparison between different models on image denoising ($\sigma = 0.15$)

	In-dist (CelebA-HQ)		OOD (LSUN-bedroom)	
	PSNR	**SSIM**	**PSNR**	**SSIM**
Noisy	16.0	0.29	16.3	0.30
BM3D [5]	28.3	0.84	27.6	0.81
DnCNN [12]	30.9	0.91	29.0	0.85
IRCNN [46]	31.1	0.91	29.6	0.87
U-Net [25]	30.2	0.90	24.1	0.81
DRU-Net [50]	30.7	0.91	29.0	0.86
MLPatch (no filter; ours)	30.3	0.88	29.0	0.85
MLPatch (no learnable patch shape; ours)	30.5	0.90	29.0	0.86
MLPatch (no CCPG; ours)	30.8	0.90	29.2	0.86
MLPatch (ours)	30.9	0.91	29.4	0.87

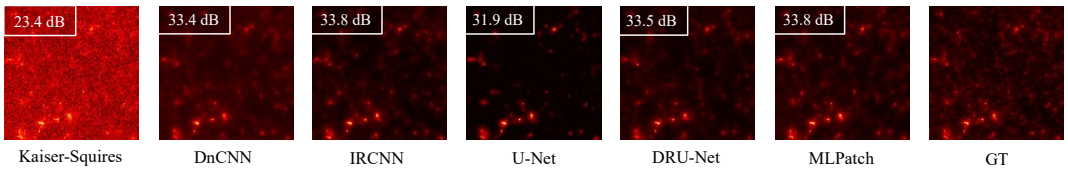


Figure 2.3: Performance comparison between different models on dark matter mapping in resolution 128×128.

2.3.2 Dark matter mapping

As shown in (2.4), the dark matter mapping problem can be interpreted as a denoising problem with colored noise characterized by a local forward operator. We apply MLPatch on the naive KS inversion κ_{KS} to recover the convergence field κ. For additional details regarding the experimental setup, please refer to Appendix 2.6.3.

In Figure 2.3, the reconstructed convergence fields from the KS image are presented for various models at resolution 128×128. The performance parity between MLPatch and SOTA CNNs indicates the relevance of the locality property derived from (2.4). In the next section, we show that MLPatch has much better memory efficiency, particularly for high-dimension image reconstruction tasks.

2.3.3 Computational efficiency and high dimensional image reconstruction

As discussed in Section 2.2.5, MLPatch can run on GPUs with small memory by reducing the pixel batch size in both training and inference. In Table 2.2, we compare the memory

Table 2.2: Model comparison for memory usage and training time on one epoch.

	Params	128×128	256×256	512×512	1024×1024
DnCNN [12]	3M	23GB / 200s	78GB / 1680s	$>$ 80GB	$>$ 80GB
IRCNN [46]	3M	16GB / 120s	55GB / 500s	$>$ 80GB	$>$ 80GB
U-Net [25]	8M	6GB / 40s	16GB / 100s	60GB / 380s	$>$ 80GB
DRU-Net [50]	8M	7GB / 60s	22GB / 220s	79GB / 800s	$>$ 80GB
MLPatch (with filter; ours)	4M	3GB / 80s	3GB / 100s	5GB / 140s	15GB / 300s
MLPatch (no filter; ours)	3M	2GB / 60s	2GB / 80s	2GB / 120s	3GB / 260s

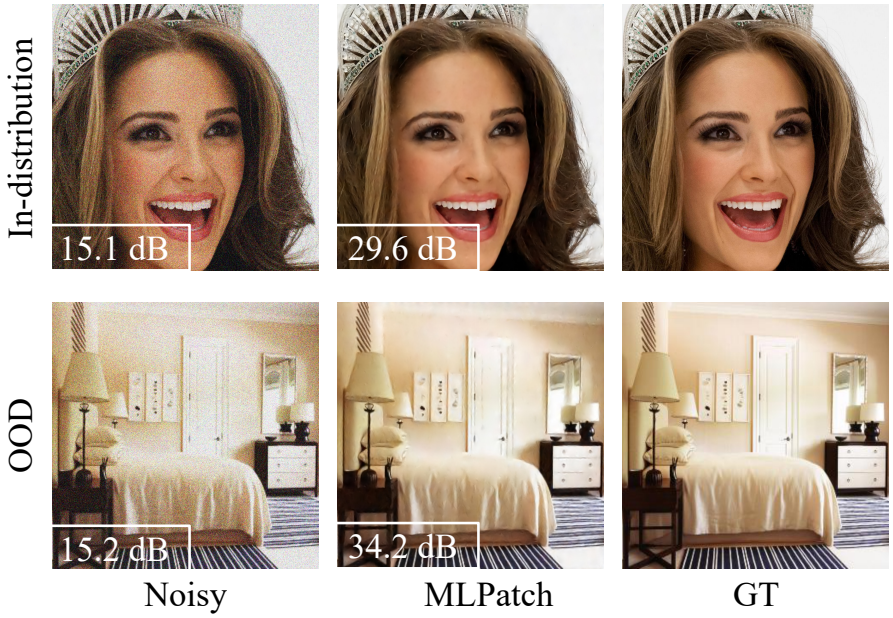
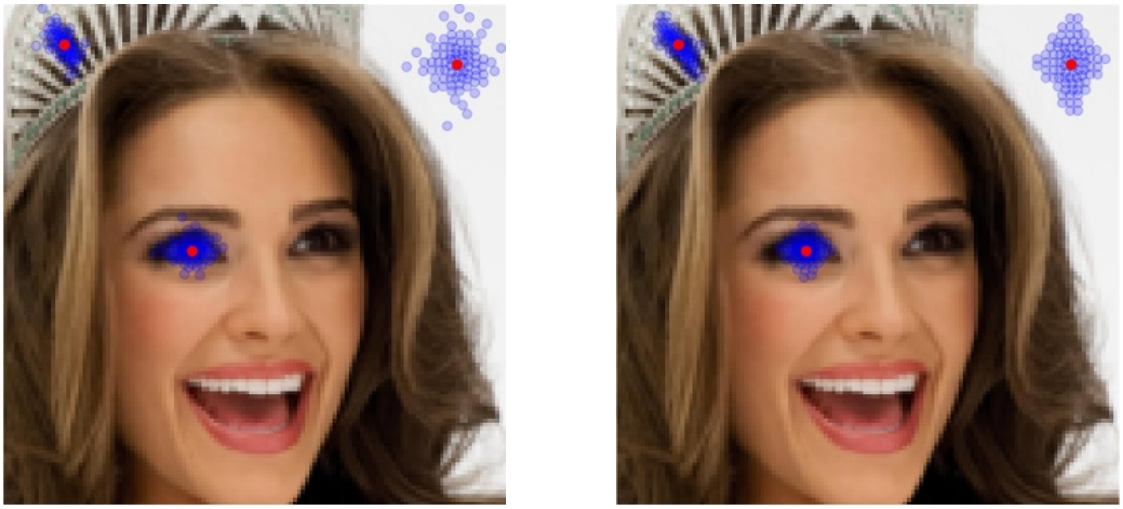


Figure 2.4: MLPatch reconstructions on in-distribution and OOD data for image denoising ($\sigma = 0.2$) in resolution 1024×1024.

usage and training time (per epoch) for various models when training on different image resolutions with object batch size 64 and on a dataset with around 30000 training samples. This comparative analysis clearly shows that MLPatch, with pixel batch size 512, has a small memory usage and fast training even in high-resolutions like 1024×1024, making it suitable for real-world applications. Figure 2.4 illustrates the images reconstructed by MLPatch on the image denoising task ($\sigma = 0.2$) in resolution 1024×1024.

2.3.4 Adaptive patch geometry

As discussed in Section 2.2.2, MLPatch can adaptively estimate the patch geometry based on the local information. Figure 2.5 illustrates the learned patch geometry for different pixels. The estimated position of the relevant features aligns with intuition. The



(a) T = 1 (b) T = 3

Figure 2.5: The learned patch geometry for different pixels. As expected, larger T results in more accurate localization of the relevant information.

possibility to identify features used to estimate a given pixel may be important for more informed analysis and interpretation of the results. Moreover, this Figure shows that larger T results in better localization of the relevant information.

2.3.5 Shift and rotation equivariance analysis

As discussed in Section 2.2.3, MLPatch is shift equivariant by design. In this section, we assess its performance on image denoising when the input image is shifted or rotated. As shown in Figure 2.6, we horizontally shift the input noisy image by 15 pixels. As expected, MLPatch's reconstruction is shifted by 15 pixels and exhibits no degradation in the reconstruction quality.

Figure 2.7 demonstrates the MLPatch performance under 90° rotations. We observe negligible degradation in reconstruction quality. Note that we did not use any data augmentation to make the model robust under translation and rotation during training.

2.4 MLPatch for general inverse problems

As detailed in Section 2.3, MLPatch performs well on inverse problems with local forward operators. In this section, we generalize it to non-local problems. Motivated by strong performance on image denoising presented in Section 2.3.1, we use a pre-trained MLPatch as a denoising prior in the plug-and-play (PnP) framework. In the following,

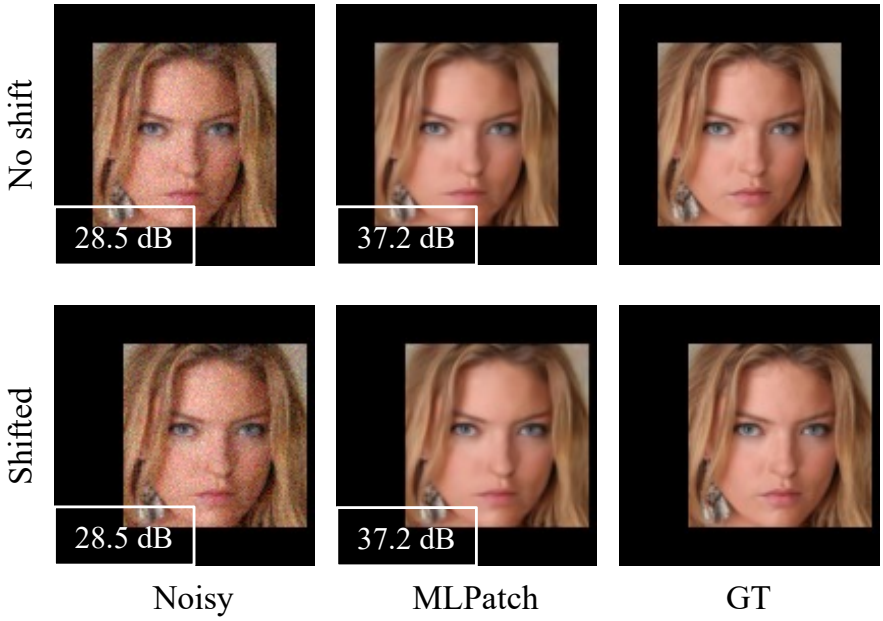


Figure 2.6: Shift equivariance of MLPatch; there is no degradation in reconstruction quality when the input noisy image is shifted by 15 pixels.

we provide a brief overview of PnP.

2.4.1 Plug-and-Play ADMM

PnP [40] is a powerful framework for image reconstruction that leverages the separation of data fidelity and regularization terms. PnP methods integrate pre-trained denoisers into the reconstruction process, allowing them to handle a wide range of imaging tasks without needing to train models for each specific task. The original PnP [40] framework is developed based on the alternating direction method of multipliers (ADMM) [94], where new variables $\mathbf{u}$ and $\mathbf{v}$ are introduced to decouple the data fidelity and regularization terms in (1.1) using augmented Lagrangian,

$$\min_{\mathbf{f},\mathbf{v}} \max_{\mathbf{u}} \left\{ \frac{1}{2\sigma^2}\|\mathbf{q}-\mathbf{A}\mathbf{f}\|_2^2 + R(\mathbf{v}) + \frac{1}{2\eta}\|\mathbf{f}-\mathbf{v}+\mathbf{u}\|_2^2 - \frac{1}{2\eta}\|\mathbf{u}\|_2^2 \right\}, \tag{2.16}$$

where η is a parameter that controls the convergence rate. We alternate the optimization over $\mathbf{f}$, $\mathbf{u}$ and $\mathbf{v}$ as follows,

$$\mathbf{f}_k = h(\mathbf{v}_{k-1} - \mathbf{u}_{k-1}; \alpha), \tag{2.17}$$
$$\mathbf{v}_k = \text{prox}_R(\mathbf{f}_k - \mathbf{u}_{k-1}; \eta), \tag{2.18}$$
$$\mathbf{u}_k = \mathbf{u}_{k-1} + (\mathbf{f}_k - \mathbf{v}_k), \tag{2.19}$$

Figure 2.7: Rotation equivariance of MLPatch; we observe negligible degradation in reconstruction quality when the input noisy image is rotated by $90°$.

where $\alpha = \dfrac{\sigma^2}{\eta}$,

$$h(\mathbf{z};\alpha) \triangleq (\mathbf{A}^*\mathbf{A} + \alpha)^{-1}(\mathbf{A}^*\mathbf{q} + \alpha\mathbf{z}), \tag{2.20}$$

$$\text{prox}_R(\mathbf{z};\eta) \triangleq \underset{\mathbf{f}}{\text{argmin}} \frac{1}{2\eta}\|\mathbf{f} - \mathbf{z}\|_2^2 + R(\mathbf{f}), \tag{2.21}$$

and $\mathbf{A}^*$ is the adjoint of the forward operator. The proximal operator in (2.21) can be interpreted as the denoiser of $\mathbf{z}$ with image prior $R(\cdot)$ and AWGN variance η. The key idea of the PnP algorithm is to heuristically apply a powerful image denoiser, like a pre-trained CNN, in (2.21). This has resulted in SOTA performance on various inverse problems [50].

2.4.2 MLPatch-ADMM

We use a pre-trained MLPatch denoiser in (2.21) to solve imaging problems. Algorithm 1 describes the proposed MLPatch-ADMM framework. MLPatch-ADMM can evaluate images at arbitrary coordinates thanks to the continuous image representation described in (2.7). We note that by reducing the pixel batch size, it can run on machines with small memory.

Image in-painting. We use the MLPatch denoiser trained on CelebA-HQ samples in resolution 128×128 as described in Section 2.3.1. We set $\alpha = 0.05$ and run the ADMM

Algorithm 1 MLPatch-ADMM

Input: $\mathbf{q}$, $\mathbf{A}$, $\phi^\star$

Parameter: α

$\mathbf{f} = 0$, $\mathbf{u} = 0$, $\mathbf{v} = 0$;

for $k = 0$ **to** $K - 1$ **do**

$\mathbf{f}_k \leftarrow h(\mathbf{v}_{k-1} - \mathbf{u}_{k-1}; \alpha)$;

$\mathbf{v}_k(\mathbf{x}) \leftarrow \text{MLPatch}_{\phi^\star}(\mathbf{x}, \mathbf{f}_k - \mathbf{u}_{k-1}), \quad \forall \mathbf{x} \in \mathbf{v}_k$;

$\mathbf{u}_k \leftarrow \mathbf{u}_{k-1} + (\mathbf{f}_k - \mathbf{v}_k)$;

end

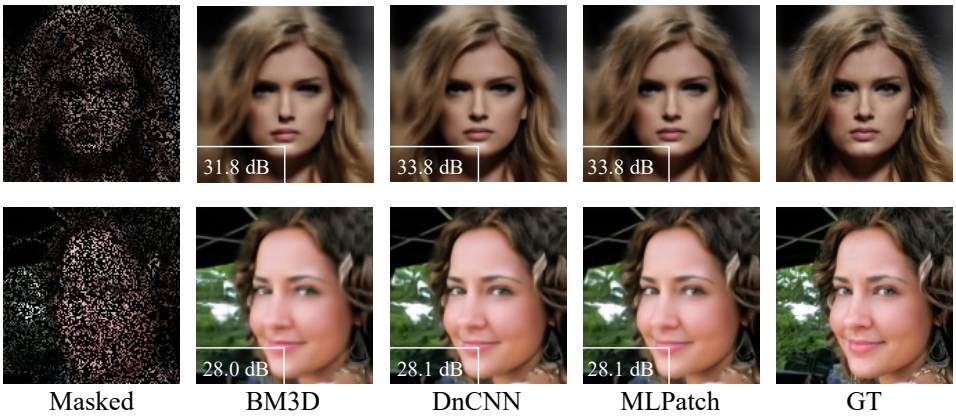


Figure 2.8: Comparison on image in-painting ($p = 30\%$) in resolution 128×128 for PnP algorithm with different denoising priors.

algorithm for a total of 90 iterations, a number we found sufficient to ensure convergence for all the models involved in this study. In Figure 2.8, we compare different denoising priors on in-painting problem at the original resolution 128×128 where $p = 30\%$ of the image pixels are randomly masked. We see that MLPatch-ADMM achieves high-quality reconstructions comparable to those of DnCNN. Figure 2.9 showcases the performance of MLPatch-ADMM at a higher resolution 512×512 on both in-distribution and OOD data. It shows that once MLPatch denoiser is trained, it can be used for solving general inverse problems at any resolution with strong generalization to OOD data.

Magnetic resonance imaging (MRI). We evaluate MLPatch-ADMM on 2D MRI where we train MLPatch denoiser on 7000 samples from single-coil brain images in FastMRI dataset [60]. We consider an acceleration factor 4 with a Cartesian undersampling pattern, where 4% of the central region is used, following [60]. Figure 2.10 shows MLPatch-ADMM reconstructions on in-distribution brain images and OOD knee images in resolution 128×128. We see that MLPatch-ADMM is robust across different anatomical regions.

Radio interferometry. We next apply MLPatch-ADMM to radio interferometric imaging. We use non-uniform fast Fourier transform (NUFFT) to simulate radio interferometric

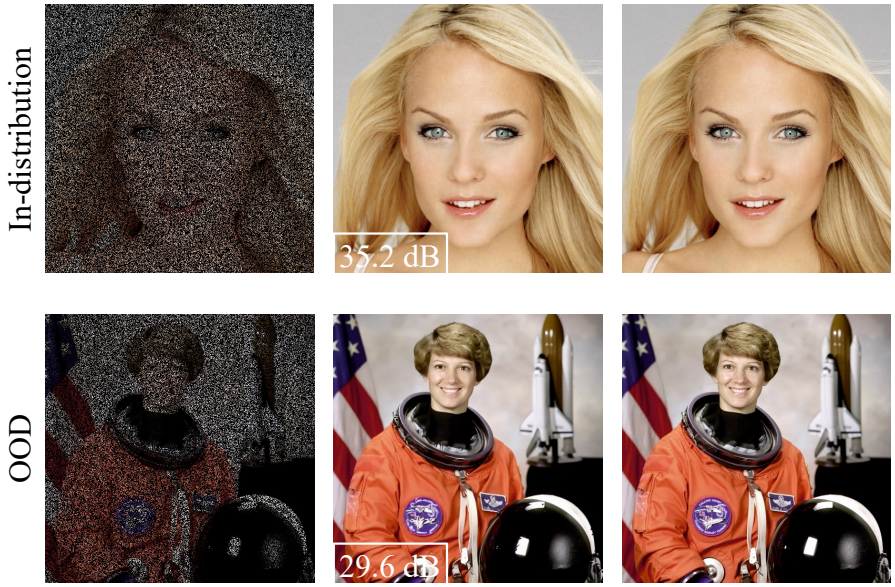


Figure 2.9: MLPatch-ADMM performance on image in-painting ($p = 30\%$) for in-distribution and OOD data in resolution 512×512 when MLPatch denoiser is trained on CelebA-HQ samples in low-resolution 128×128.

measurements from 128×128 images using an upsampling factor of 2 and Kaiser-Bessel kernels with a size of 6×6 pixels; We use the simulated uv-coverage of the MeerKAT telescope and the measurements with complex Gaussian noise at the SNR of 30dB. We train MLPatch denoiser on 13000 images of simulated galaxies created from the IllustrisTNG simulations [95]. Figure 2.11 illustrates the performance of the MLPatch-ADMM at different resolutions. We find that MLPatch-ADMM can successfully address inverse problems with non-local forward operators.

2.5 Summary

We introduced MLPatch, a coordinate-based local image reconstruction framework with strong generalization to OOD data and resolution-agnostic memory usage during training. Notably, MLPatch achieves SOTA performance on local inverse problems such as image denoising and dark matter mapping, all while maintaining a significantly smaller memory footprint compared to standard CNNs. MLPatch can generate reconstructions in higher resolutions than those encountered during training. In the next chapter, we show that this coordinate-based reconstruction pipeline can be generalized for CT image reconstruction resulting in an efficient framework with strong generalization.

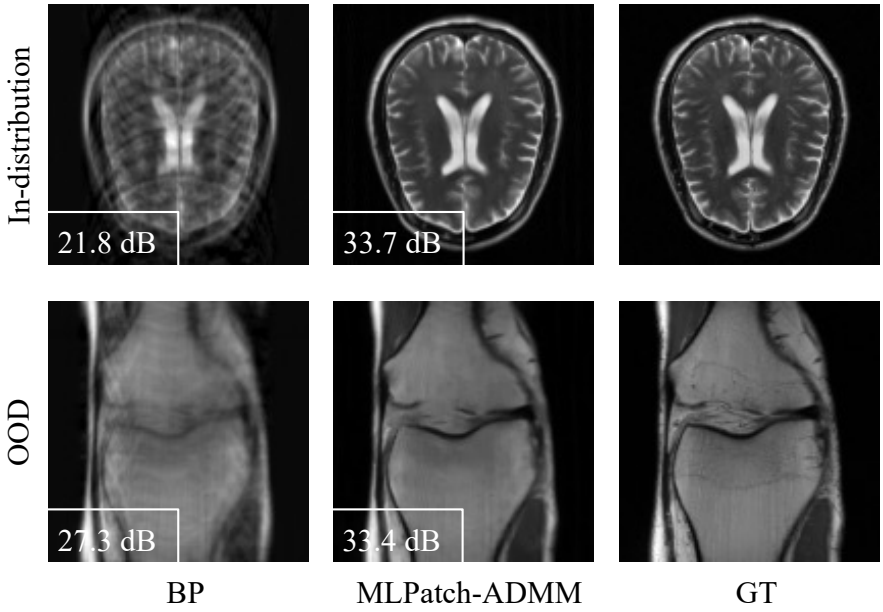


Figure 2.10: MLPatch-ADMM performance on MRI for in-distribution and OOD data in resolution 128×128.

2.6 Appendix

2.6.1 Network architecture and training details

For MLPatch architecture, we use a simple MLP comprising 8 hidden layers, each with dimensions [1024, 1024, 1024, 512, 512, 512, 256, 256] with ReLu activations. As shown in Figure 2.12, we used circular patch geometry consisting of 5 concentric circles each with 16 pixels plus the center pixel, in total 81 pixels. When learning the patch geometry, we used the circular patch as initialization. We concatenated the real and complex values of the filtered input image along the channel dimension before patch extraction.

Our model is implemented in PyTorch [96] and we use an Nvidia A100 GPU with 80GB memory. All models in Section 2.3 were trained for 200 epochs with ℓ_1 loss using Adam optimizer [97] with learning rate 10^{-4} and object batch size 64. In the case of MLPatch, for each mini-batch of random objects, we performed optimization on a random mini-batch of 512 pixels 2 times.

2.6.2 Learned filter

In this section, we analyze the learned filter $\mathbf{H}$ in (2.14) to understand how it processes noisy images. As shown in Figure 2.13, the learned filter remains localized in image space after training which prevents the MLP from overfitting the large-scale features.

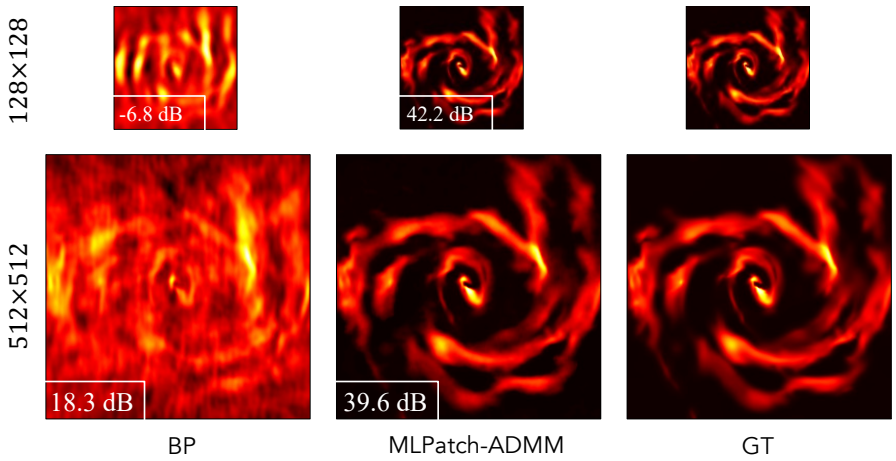


Figure 2.11: MLPatch-ADMM performance on radio interferometry imaging; the MLPatch denoiser is trained on 128×128 images, we can run the MLPatch-ADMM algorithm at any resolution.

2.6.3 Experimental details: dark matter mapping

We used the κTNG convergence maps [98], generated from the widely used IllustrisTNG hydrodynamical simulation suite [99–103], to build the train and test datasets for the models presented in Figure 2.3. The κTNG dataset consists of $10\,000$ realizations of $5 \times 5\text{deg}^2$ convergence maps for each of the 40 different source redshifts with $0 < z_s < 2.6$. For the sake of simplicity, we have selected a single redshift slice, the 20^{th} plane, corresponding to $z_s = 0.858$ for our experiments.

From the simulated convergence fields, κ, we have constructed the observed shear fields using Equation 2.2. We follow [104, §5.2.1] for the noise simulation. The shear noise $\mathbf{n}$ for each component of the shear and for each pixel i is simulated as $\mathbf{n}_i \sim \mathcal{N}(0, \sigma_i^2)$. The per-pixel standard deviation can be computed as follows

$$\sigma_i = \frac{\sigma_e}{\sqrt{(\theta^2 / n_{\text{grid}}^2) \times n_{\text{gal}}}}, \tag{2.22}$$

where σ_e is the standard deviation of the intrinsic ellipticity distribution, θ^2 is the area of the simulated field in arcmin^2, n_{grid}^2 is the total number of pixels in the field, and n_{gal} is the number density of galaxy observations given in number of galaxies per arcmin^2. We have used $\sigma_e = 0.37$, which is the typical intrinsic ellipticity standard deviation, $\theta = 300$ arcmin, which corresponds to the κTNG simulated area of $5 \times 5\text{deg}^2$, $n_{\text{gal}} = 30$ arcmin^{-2}, which corresponds to the projected number density expected in Stage IV surveys like *Euclid* [105], and $n_{\text{grid}} = 128$.

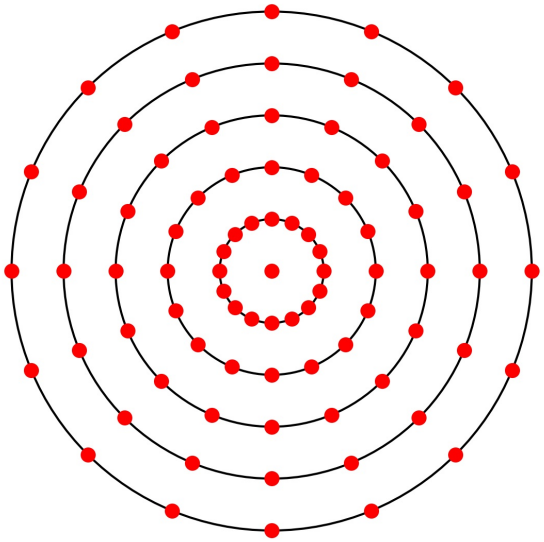
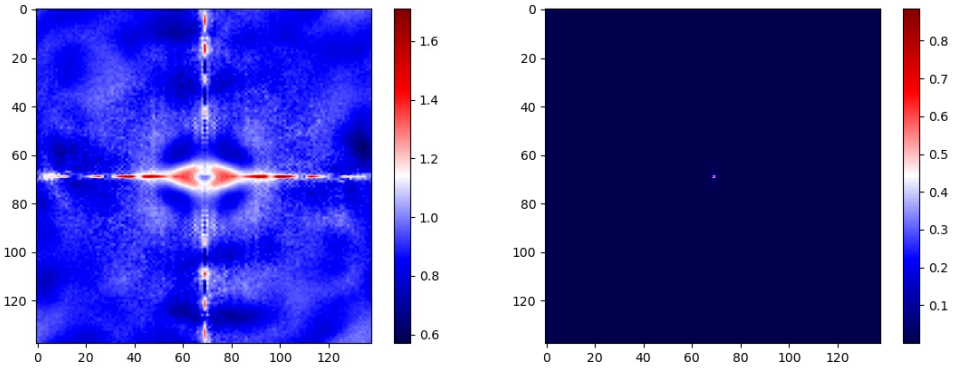


Figure 2.12: Circular patch geometry



Fourier Space Data Space

Figure 2.13: The learned filter $\mathbf{H}$ in (2.14); as expected, it has a small support in image space which prevents the MLP from overfitting large-scale features.

Chapter 3

GLIMPSE: Generalized Local Imaging with MLPs

In Chapter 2, we showed that our proposed local processing network can effectively solve a variety of imaging problems including image denoising, dark matter mapping and MRI with strong generalization on out-of-distribution data. In this chapter, we extend our local processing framework for an important imaging modality, computed tomography.

Convolutional neural networks (CNNs) have become the standard approach for tomographic image reconstruction [106]. U-Net [25] has emerged as an architecture underpinning numerous deep learning reconstruction methods, applied with great success to a variety of imaging problems including computed tomography (CT) [14], magnetic resonance imaging (MRI) [65] and photoacoustic tomography [26]. Its success is often attributed to its particular multi-scale architecture [27].

At the same time, certain aspects of multi-scale CNNs complicate their application to real problems. Despite good performance on in-distribution test images similar to the training data, they often overfit specific image content resulting in poor generalization to distribution shifts in image content and sensing as shown in this chapter. Model-based networks attempt to address this drawback by integrating the forward and adjoint operators into multiple network layers or iterations [107–112]. However, the required memory for CNNs directly scales with image resolution [113]. For instance, the widely used U-Net requires a substantial 140GB memory and 2600 seconds per epoch when training on 1024×1024 images using two Tesla A100 GPUs. This latter drawback is further exacerbated with model-based networks such as learned primal-dual (LPD) [108], which achieves strong performance but requires over 80GB memory and very long training time even at a lower resolution of 512×512. This increased memory demand is due to the repeated application of the forward model and its adjoint in forward and backward passes of the neural network. This makes standard CNN-based pipelines impractical for real-world scenarios involving resolutions beyond 512×512.

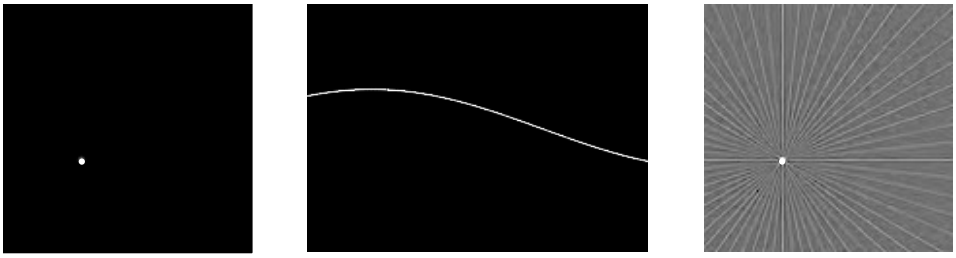


Image Sinogram FBP

Figure 3.1: A point source image, its sinogram, and the sparse view FBP reconstruction. While the corresponding measurements for this pixel have sinusoidal support in the sinogram, this information is diffused all over the FBP image. *The contrast of the FBP image has been stretched to emphasize this effect.*

To better understand the mechanics behind the poor generalization of U-Net-like CNNs which compute the reconstruction from filtered back-projections (FBP) [114], we designed an experiment as follows. Figure 3.1 shows an object with a point source, its sparse view sinogram measurements with sinusoidal support, and the FBP reconstruction. It is evident that the FBP is supported over the entire field of view. This observation raises the question of the ideal receptive field size for CNNs like U-Net: a large receptive field may be beneficial to capture all information correlated with the value of a target pixel [115].

However, models with large receptive fields often overfit specific image content in training data which leads to poor generalization on out-of-distribution samples [20]. Indeed, Figure 3.2 shows that while U-Net produces good results when tested on in-distribution data similar to training data (here chest images), it performs poorly on out-of-distribution (here brain images). This makes CNNs like U-Net problematic in domains such as medical imaging where robustness over distribution shifts and other uncertain and variable factors is of relevance [116].

In this chapter, we introduce a new deep learning imaging architecture termed GLIMPSE—a simple local processing neural network adapted to the geometry of computed tomography. As shown in Figure 3.3, to recover the image intensity at a given target pixel, we use an MLP that takes only the local sinogram measurements associated with this pixel and its neighbors. There is no back-projection step. This localization results in robust performance, particularly when dealing with OOD data.

At the same time, this design makes GLIMPSE highly computationally efficient; it permits training on mini-batches of both *pixels* and objects. This flexibility leads to fast and efficient training, requiring a small, fixed amount of memory almost independent from the image resolution. This allows training GLIMPSE on large, realistic images in resolution 1024×1024 and beyond.

We built GLIMPSE to be fully differentiable, all the way down to the sensing and

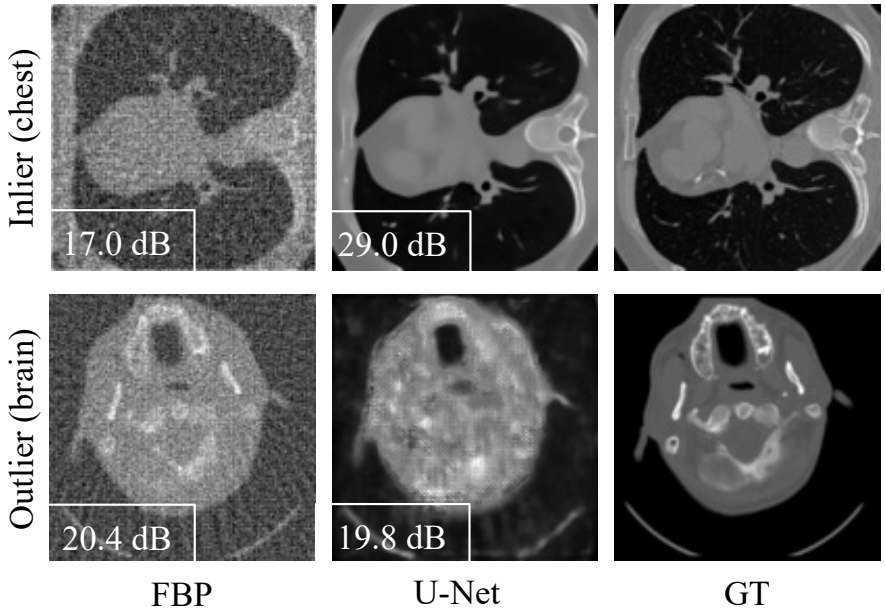


Figure 3.2: Performance of U-Net [25] trained on chest images: evaluation on in-distribution test data (chest samples) and OOD brain samples shows that the large receptive field of U-Net hinders its ability to generalize on OOD samples, with its PSNR even falling below that of FBP reconstruction.

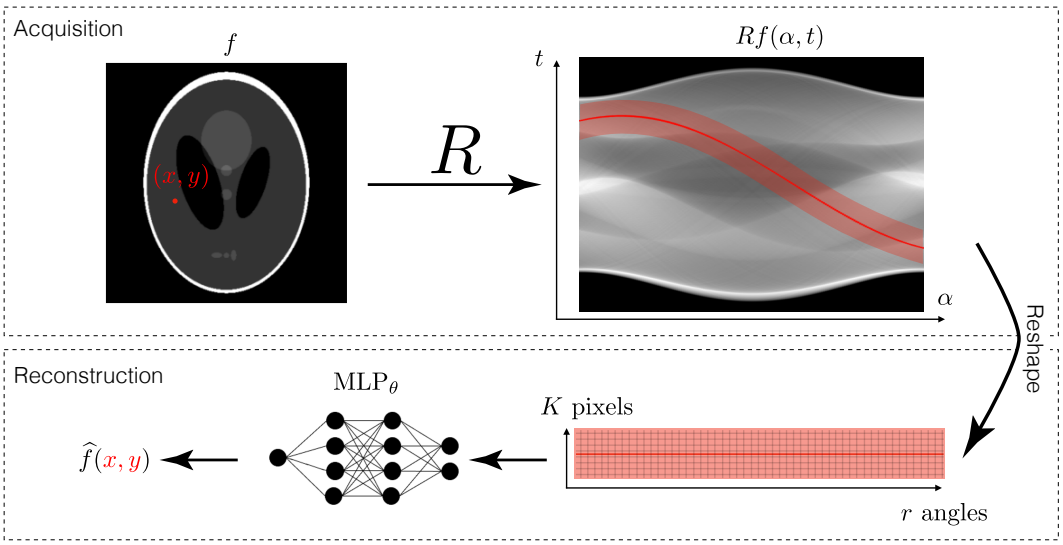


Figure 3.3: GLIMPSE; a single MLP processes the measurements associated with the pixel (x, y) and its neighbors extracted from the sinogram. This local processing network has promising performance on OOD data while being computationally efficient all due to its locality.

integration geometry. This has several advantages over the standard CNN-based architectures. For instance, most methods for CT image recovery strongly rely on the sensor geometry information encoded in the forward operator, whether explicitly, as seen in methods like FBP [114], SART [117], LGS [107], and LPD [108] or implicitly as used in U-Net [25] when taking FBP as input. This fixed geometry is a problem when faced with uncertainties in calibration or blind inversion problems where the sensor geometry information is entirely unavailable. While such uncertainties might degrade the quality of reconstructions of the standard methods [118, 119], our differentiable architecture allows the optimization of projection angles which can estimate the right projection angles and improve the quality of reconstructions.

3.1 Related works

Model-based vs model-free inversion. There are two major classes of deep-learning-based approaches to CT: *model-based* and *model-free* inversion. In the model-based approach, neural networks process raw sinograms and map them to the desired CT images while the Radon transform is integrated into multiple network layers or iterations [107, 108, 111]. These methods perform remarkably well across various inverse problems, but they are computationally expensive, especially during training [113]. The high computational cost is due, among other factors, to the repeated application of the Radon transform and its adjoint in forward and backward passes of the neural networks.

In contrast, model-free approaches offer a computationally cheaper alternative. The Radon transform (or its adjoint) is only used once in FBP computation before the neural network [14, 23, 120]. However, these models often require deep networks with a large receptive field to leverage the information delocalized across the FBP image. Recently, Hamoud et al. [121] used a measurement rearrangement technique to stratify back-projected features by angle and thus enable the use of smaller, shallower CNNs.

MLPs for imaging A multi-layer perceptron (MLP) is a fundamental neural architecture used in a great variety of applications. Recently, vision transformers [122] and MLP-mixers [123] have shown promising performance in various computer vision tasks like image classification [124] and image restoration [125]. While, unlike CNNs, a vanilla MLP lacks a good inductive bias for imaging, in particular translation equivariance, vision transformers and MLP-mixers restore it by processing patches instead of entire images [126]. However, these strategies require large datasets and networks to achieve performance comparable with CNNs. In our work, we propose a differentiable local processing network for CT imaging, demonstrating that even a small MLP can achieve performance on par with or even surpassing that of popular image-to-image CNNs.

Uncalibrated CT imaging. In CT imaging, the acquisition operator is often known but an insufficient number of measurements is obtained. This may occur when a reduced

number of projections is used to minimize radiation exposure or shorten acquisition time (sparse view) or when only a limited cone of projection angles may be used (limited view). In certain situations, the acquisition operator is only partially or approximately known. Neglecting this uncertainty can result in a significant drop in the quality of the reconstructions [118]. To tackle this challenge, total least squares approaches have been developed, involving the perturbation of an assumed forward operator [127–129]. Recently, Gupta et al. [119] used auto-differentiation and gradient descent to estimate the uncalibrated forward operator in a self-consistent manner.

3.2 Computed tomography

CT imaging [130] plays an important role in many applications including medical diagnosis [1], industrial testing [131], and security [132]. We consider 2D computed tomography where the image of interest $f(\mathbf{x})$ with size $D \times D$ is reconstructed from measurements of (X-ray) attenuation. The forward model is the Radon transform Rf which computes integrals of $f(\mathbf{x})$ along lines L,

$$Rf(L) = \int_L f\big(\mathbf{x}\big)|d\mathbf{x}|. \tag{3.1}$$

We parameterize a line L by its distance from the origin t and its normal vector's angle with the x-axis α. We can then reformulate (3.1) as

$$Rf(\alpha, t) = \int_{-\infty}^{\infty} f\big(x(z), y(z)\big)dz, \tag{3.2}$$

where,

$$x(z) = z\cos(\alpha) - t\sin(\alpha), \tag{3.3}$$
$$y(z) = z\sin(\alpha) + t\cos(\alpha). \tag{3.4}$$

The image of interest is observed from a finite set of r different viewing directions $\{\alpha_m\}_{m=1}^r$, each having N parallel, equispaced rays. The measurements of the attenuation are then represented as a transform-domain "image" $\mathbf{s} \in \mathbb{R}^{N\times r}$ called a sinogram.

Standard methods for CT image recovery discretize the image of interest $f(\mathbf{x})$ into a discrete image $\mathbf{f} \in \mathbb{R}^{N\times N}$ supported on an $N \times N$ grid. After discretization, the forward model can be written as

$$\mathbf{s} = \mathbf{A}\mathbf{f} + \mathbf{n}. \tag{3.5}$$

where $\mathbf{A}$ is the matrix of the discretized Radon transform and we model the measurement noise by $\mathbf{n}$. The most commonly used analytical inversion method is the filtered backprojection (FBP),

$$\mathbf{f}_{x,y}^{\text{FBP}} = \sum_{m=1}^{r} \tilde{\mathbf{s}}[y\cos(\alpha_m) - x\sin(\alpha_m), m], \tag{3.6}$$

where $\mathbf{f}^{\text{FBP}} \in \mathbb{R}^{N\times N}$ is the FBP reconstruction, $\tilde{\mathbf{s}}[\cdot, m] = \mathbf{s}[\cdot, m] \star \mathbf{h}$, $\mathbf{h}$ is a certain high-pass filter, $\star$ denotes the convolution and linear interpolation is used in (3.6) for evaluating $\tilde{\mathbf{s}}[x, \cdot]$ when x is not an integer. As shown in Proposition 3.5.1 in Appendix 3.5.3, while the Ram–Lak filter is the optimal choice for $\mathbf{h}$ in the case of noise-free complete measurements, it can amplify the noise in real-world noisy measurements, leading to poor reconstruction.

With measurement noise and an incomplete collection of projections, tomographic image reconstruction from a sinogram becomes an ill-posed inverse problem that requires an image prior. In the following section, we introduce our proposed method, GLIMPSE, designed so that it respects the geometry of CT imaging.

3.2.1 GLIMPSE: Local imaging with MLPs

To recover the image $\mathbf{f}(x, y)$ at location $\mathbf{x} = (x, y)$, we identify the elements in the sinogram $\mathbf{s}$ influenced by this pixel. As illustrated in Figure 3.1, the corresponding measurements for the pixel (x, y) are supported along a sinusoidal curve in the sinogram; we denote them $\text{SIN}_{x,y} \in \mathbb{R}^r$, with elements being given as

$$\text{SIN}_{x,y}[m] = \mathbf{s}[y\cos(\alpha_m) - x\sin(\alpha_m), m]. \tag{3.7}$$

Similar to (3.6), we can use interpolation to evaluate $\mathbf{s}[x, \cdot]$ for non-integer x. This localization is formally captured by the following proposition.

Proposition 3.2.1 (Impulse response of Radon transform)**.** *Let $f(u, v) = \delta(u - x, v - y)$ be the Dirac delta distribution in $\mathbb{R}^2$ at location (x, y). Its Radon transform (in the sense of distributions) is*

$$Rf(\alpha, t) = \begin{cases} 1, & \text{if } t = r\cos(\alpha + \varphi) \\ 0, & \text{otherwise}, \end{cases}$$

where $r = \sqrt{x^2 + y^2}$, $\varphi = \operatorname{atan2}(y, x)$, and $\operatorname{atan2}(\cdot, \cdot)$ the four-quadrant arctangent.

The proof is standard and outlined for completeness in Appendix 3.5.4.

The sinusoidal portion of the sinogram $\text{SIN}_{x,y}$ should have enough information to recover the pixel intensity (x, y) as it contains all the measurements associated with this pixel. Note however that the pixel at (x, y) influences the integral over any line passing through it and thus also the parts of the sinogram corresponding to pixels on those lines. This can be loosely thought of as a consequence of non-orthogonality of the Radon transform. The above statement is thus more precisely a statement about the *filtered* sinogram since information is "relocalized" by the high-pass filtering step in the FBP.

This is related to the celebrated support theorems of Sigurdur Helgason, Jan Boman, and others [133–136]. These theorems state that under appropriate conditions a compactly-supported image may be recovered from a compactly-supported subset of its Radon data. These results do not involve filtering explicitly, but its influence is implicit. They apply to idealized sampling and SNR conditions.

Indeed, the high-pass filtering in the FBP is derived for noiseless data and a continuum of observed angles. In reality the projections are corrupted with noise and come from a sparse subset of projection angles. We address this by 1) incorporating "contextual information" about the target pixel and 2) letting the filter be learnable to adapt it to the specifics of discretization and noise.

As shown in Figure 3.3, we exploit the spatial regularity of medical images (encoded in training data) by using the measurements that provide *local* information around (x, y). This ensures that the model does not overfit large-scale features in the training data while maintaining low computational complexity. We thus additionally extract from the sinogram the regions associated with the neighboring pixels around (x, y) and store this information in vector $\mathbf{p}_{x,y}$,

$$\mathbf{p}_{x,y} = \{\text{SIN}_{x+dn,y+dn'} | n, n' = -\lfloor C/2 \rfloor, \cdots, \lfloor C/2 \rfloor\}, \tag{3.8}$$

where $K = C^2$ determines the number of neighboring pixels around (x, y) for an odd number $C \geq 1$ and d denotes the scale of the window which adjusts the receptive field. In order to recover the image at pixel (x, y) from $\mathbf{p}_{x,y}$, we use a multi-layer perception $\text{MLP}_\theta : \mathbb{R}^{r \times K} \to \mathbb{R}$ parameterized by θ,

$$\hat{\mathbf{f}}(x, y) = \text{MLP}_\theta\big[\mathbf{p}_{x,y}\big], \tag{3.9}$$

which estimates the pixel intensity $\hat{\mathbf{f}}_{x,y}$ from the local features around (x, y). We call the proposed model GLIMPSE, standing for generalized[1] local imaging with MLPs. In the following section, we describe how our implementation of GLIMPSE allows to adapt to noisy measurements. We then propose a training strategy with resolution-agnostic memory usage in Section 3.2.3. In Appendix 3.5.2, we show how GLIMPSE compensates for calibration errors. Further details for network architecture and training can be found in Appendix 3.5.1.

3.2.2 Adaptive filtering for noisy measurements

The Ram–Lak high-pass filter is the optimal filter $\mathbf{h}$ for the FBP reconstruction in the case of complete noise-free measurements; see Appendix 3.5.3 for a standard demonstration. In real applications, however, we always encounter noisy projections from a subset of

[1]The word "generalized" emphasizes that locality is also encoded in the transform domain, not just in real space as in some of earlier work.

angles. The Ram–Lak filter is then suboptimal and typically degrades the reconstruction quality as it amplifies high-frequency noise. Alternative filters with lower amplitudes in high frequencies like Shepp–Logan, cosine, and Hamming have been used to mitigate the noisy measurements, but they are all ad hoc choices. It is advantageous to adapt $\mathbf{h}$ to the specifics of noise and sampling strategy in the target application. To design this task-specific filter, we let MLP_θ take as input the filtered sinogram $\tilde{\mathbf{s}}[\cdot, m] = \mathbf{s}[\cdot, m] \star \mathbf{h}$ and consider the filter $\mathbf{h}$ (in Fourier space) as trainable parameters to be optimized during training. This allows us to automatically learn a noise-adaptive filter from data, again with almost no additional computational cost.

3.2.3 Resolution-agnostic memory usage in training

To simplify notation, we denote the entire GLIMPSE pipeline described above by $\hat{\mathbf{f}}(\mathbf{x}) = \text{GLIMPSE}_\phi(\mathbf{x}, \mathbf{s})$. The inputs are the target pixel coordinates $\mathbf{x} = (x, y)$ and the sinogram $\mathbf{s}$; the output is an estimate of $\mathbf{f}(x, y)$. The parameters ϕ denote the trainable parameters of GLIMPSE including the MLP weights θ, the projection angles $\{\alpha_m\}_{m=1}^r$ (see Appendix 3.5.2), the adaptive filter $\mathbf{h}$ and the window receptive field scale d. We consider a set of training data $\{(\mathbf{s}_i, \mathbf{f}_i)\}_{i=1}^L$ from the noisy sinograms and images. We optimize the GLIMPSE parameters ϕ using gradient-based optimization to minimize

$$\phi^\star = \underset{\phi}{\text{argmin}} \sum_{i=1}^{N^2} \sum_{j=1}^{L} |\text{GLIMPSE}_\phi(\mathbf{x}_i, \mathbf{s}_j) - \mathbf{f}_j(\mathbf{x}_i)|^2. \tag{3.10}$$

At inference time, we simply evaluate the image intensity at any pixel as $\hat{\mathbf{f}}_{\text{test}}(\mathbf{x}) = \text{GLIMPSE}_{\phi^\star}(\mathbf{x}, \mathbf{s}_{\text{test}})$. One major advantage of GLIMPSE compared to CNNs like U-Net and LPD is its memory and compute complexity. CNN-based models exhibit memory requirements that scale directly with image resolution, making them prohibitively expensive for realistic image resolutions. As shown in (3.10), GLIMPSE can be trained using stochastic gradient-based optimizers with the flexibility to select mini-batches from both the objects and pixels. This adaptability in mini-batch pixel selection grants a memory footprint agnostic to resolution making GLIMPSE suitable for training on realistic image resolutions like 1024×1024 and higher.

3.3 Experiments

We simulate parallel-beam X-ray CT with $r = 30$ projections uniformly distributed around the object with additive Gaussian noise to reach a signal-to-noise ratio (SNR) of 30 dB. The reconstruction quality is quantified using the peak signal-to-noise ratio (PSNR) and Structural Similarity Index (SSIM) [93]. We compare the performance of GLIMPSE with successful CNN-based models: U-Net [25], learned gradient scheme (LGS) [107]

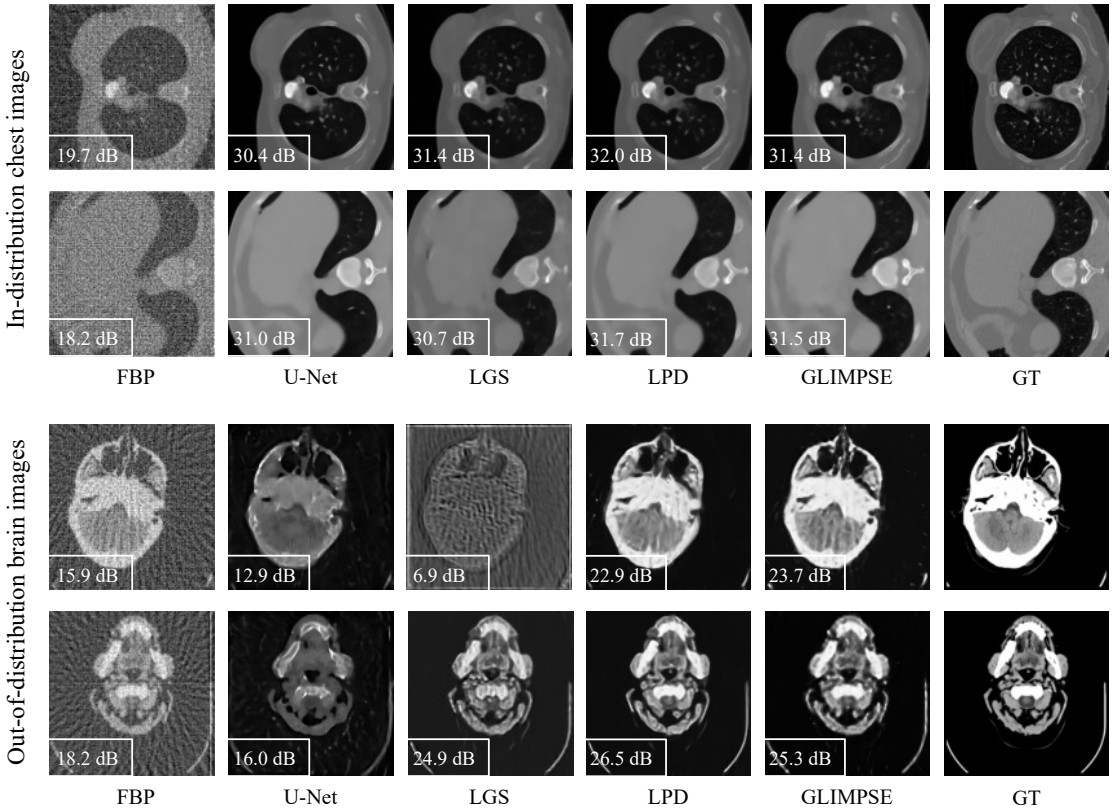


Figure 3.4: Performance of different models trained on training data of chest images evaluated on in-distribution and OOD samples for sparse view CT image reconstruction. GLIMPSE has excellent performance on OOD data due to its localized MLP, significantly better than U-Net [25] and LGS [107] and comparable with LPD [108].

and learned primal-dual (LPD) [108] for sparse view CT image reconstruction. We use 35820 training samples of chest images from the LoDoPaB-CT dataset [137] in resolution 128×128. Model performance is assessed on 64 in-distribution test samples of chest images, while 16 OOD brain images [138] are included to evaluate the generalization capability of the models. For further information regarding the network architectures and training details please refer to Section 3.5.1.

In Section 3.3.1, we compare GLIMPSE to CNN-based models for sparse view CT image reconstruction on both in-distribution and OOD data. In Section 3.3.2, we analyze the computational efficiency of the aforementioned models. We analyze the learned filters $\mathbf{h}$ across different measurement noise levels in Section 3.3.3. In Appendix 3.5.2 we consider the uncalibrated and blind scenarios.

Table 3.1: Comparison of different models for sparse view CT image reconstruction

(a) The reconstruction quality averaged on 64 test samples

	In-distribution (chest)		**Out-of-distribution (brain)**	
	PSNR	**SSIM**	**PSNR**	**SSIM**
FBP [114]	17.0	0.17	17.1	0.22
U-Net [25]	30.1	0.84	15.1	0.28
LGS [107]	30.9	0.84	20.5	0.54
LPD [108]	**31.6**	**0.86**	**25.5**	0.76
GLIMPSE	30.9	0.84	25.1	**0.79**

(b) Memory usage and training time (batch size 64)

	GLIMPSE	U-Net [25]	LGS [107]	LPD [108]
Num params	900k	7800k	19k	400k
128×128	4GB / 114s	6GB / 34s	4GB / 384s	13GB / 963s
256×256	4GB / 123s	16GB / 117s	13GB / 575s	41GB / 1517s
512×512	4GB / 185s	53GB / 460s	45GB / 1682s	$>$ 80GB
1024×1024	5GB / 419s	$>$ 80GB	$>$ 80GB	$>$ 80GB

3.3.1 Sparse view CT image reconstruction

The upper row of Figure 3.4 and Table 3.1a show the performance of different models on in-distribution test samples of chest images. This experiment shows that GLIMPSE, by leveraging only a single MLP network, can outperform successful CNNs like U-Net and achieve comparable performance with LGS and LPD methods.

The lower row of Figure 3.4 and Table 3.1a shows a comparison of the performance of various models trained on chest images when applied to OOD brain images. This experiment demonstrates that while U-Net excels on in-distribution samples, its performance significantly deteriorates on OOD data.

On the contrary, GLIMPSE shows strong performance on OOD data. Although LPD's performance on OOD data is sometimes comparable or slightly better than that of GLIMPSE, it comes at a very high memory and compute cost due to the repeated application of the forward operator and its adjoint in the network architecture; we analyze this in the next section.

3.3.2 Computational efficiency

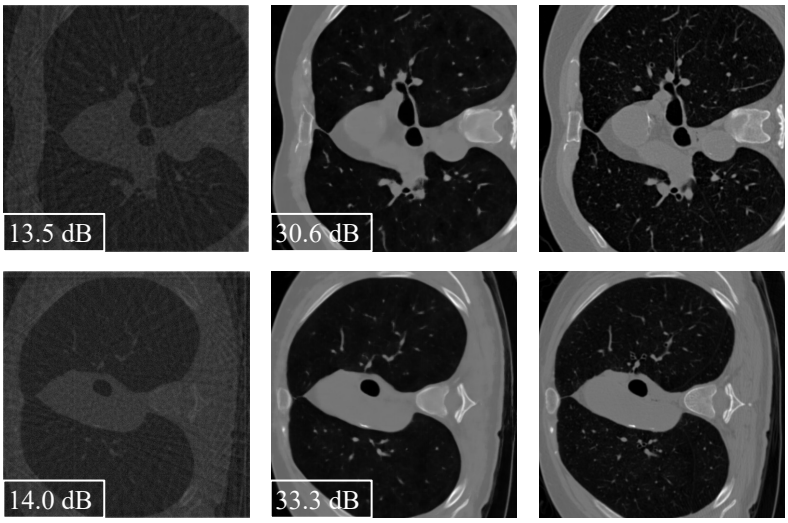
The fact that LPD far outperforms U-Net on OOD data is a testament to the benefits of incorporating the forward operator in the architecture. On the other hand, as evident from Table 3.1b, it comes at the cost of unfavorable training time and memory footprint which rapidly worsens with resolution. Table 3.1b shows that CNN-based models may become impractical already at resolutions like 512×512, even on GPUs with 80GB memory.

On the other hand, GLIMPSE is computationally efficient; the memory usage remains almost unaffected by image resolution. Remarkably, GLIMPSE can be trained with only 5GB memory in less than a day, even when dealing with resolutions of 1024×1024 and higher. Figure 3.5 shows the performance of GLIMPSE on in-distribution and OOD samples in resolution 512×512 where we considered 40dB measurement noise. This experiment demonstrates that a relatively small MLP, with almost 10 times fewer parameters than a standard U-Net, can achieve strong performance in realistic high-resolutions while maintaining a rather modest memory footprint.

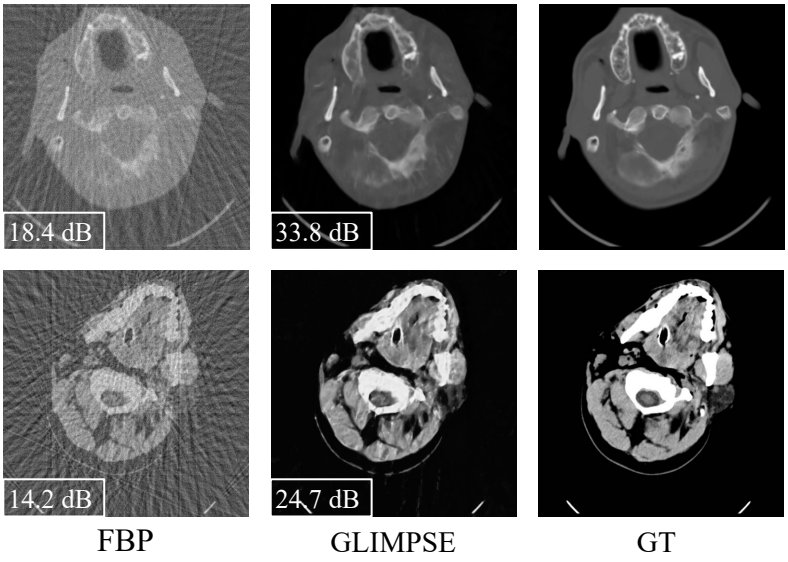
3.3.3 Learned filter

In this section, we analyze the behavior of the learned filter obtained through training of GLIMPSE introduced in Section 3.2.2 across datasets with different measurement noise levels. This analysis provides useful signal processing insights into how the properties of the learned filter are influenced by varying noise levels.

In Figure 3.6 we show the frequency response of the learned filters, alongside with



(a) In-distribution chest samples



(b) OOD brain samples

Figure 3.5: GLIMPSE's performance in resolution 512×512 trained on chest training data with $r = 30$ projections and 40dB noise; GLIMPSE requires only 4GB memory and can be trained in less than 10 hours on a single GPU.

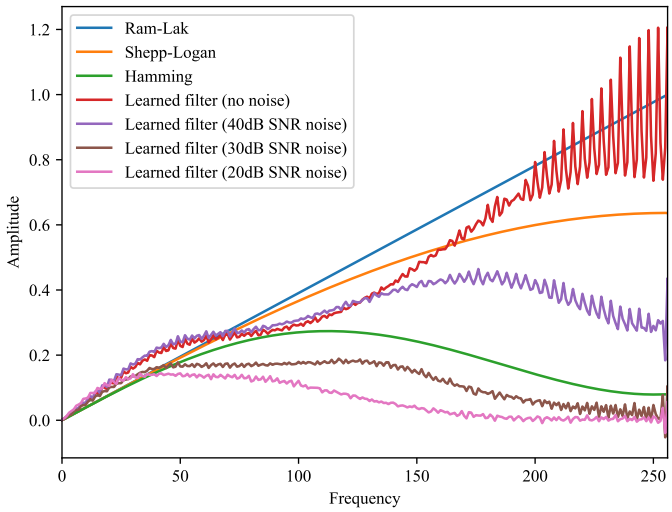


Figure 3.6: The learned filter for datasets with different noise levels, all the filtered are initialized by Ram-Lak filter in GLIMPSE architecture. By increasing the noise level, the filter assigns smaller amplitudes for high-frequencies to suppress the noise and aligns with the optimality of the Ram-Lak filter for noise-free complete measurements shown in Section 3.5.3.

standard hand-crafted filters such as Ram-Lak, Shepp-Logan, and Hamming filters. These learned filters are derived from GLIMPSE training on datasets characterized by different levels of noise.

As expected from the discussion in Appendix 3.5.3, the learned filter for noise-free measurements is similar to the Ram–Lak filter, with a relatively high amplitude in high frequencies. As the noise level increases (by decreasing the noise SNR), the filter progressively takes smaller values in high frequencies to suppress the noise. This confirms that GLIMPSE can autonomously adapt the characteristics of the filter according to the noise level observed in the training data.

3.4 Summary

We used a natural notion of locality for CT which is adapted to sinogram geometry. This is different from CNNs which reconstruct the image as a whole, and where the notion of locality (at small scales) is in the sense of real image space. Our approach adopts a coordinate-based strategy, focusing on processing the sinusoidally-shaped regions of the sinogram associated with pixels using a small MLP. Our results demonstrate that this localized processing framework does significantly improve the robustness to OOD

data while maintaining nearly constant memory requirements across different resolutions, being computationally efficient even at realistic high-resolutions.

While the memory required by GLIMPSE varies little with image resolution, a drawback of the current scheme is that memory and computing costs increase as the number of projections r. A possible alternative to the standard MLP architecture which is the culprit for this is to use mixture-of-experts layers [139–141], which selectively employ smaller MLPs for processing inputs. This mixture-of-experts approach is an effective drop-in replacement for standard MLP layers of language transformers [142] and vision transformers [122].

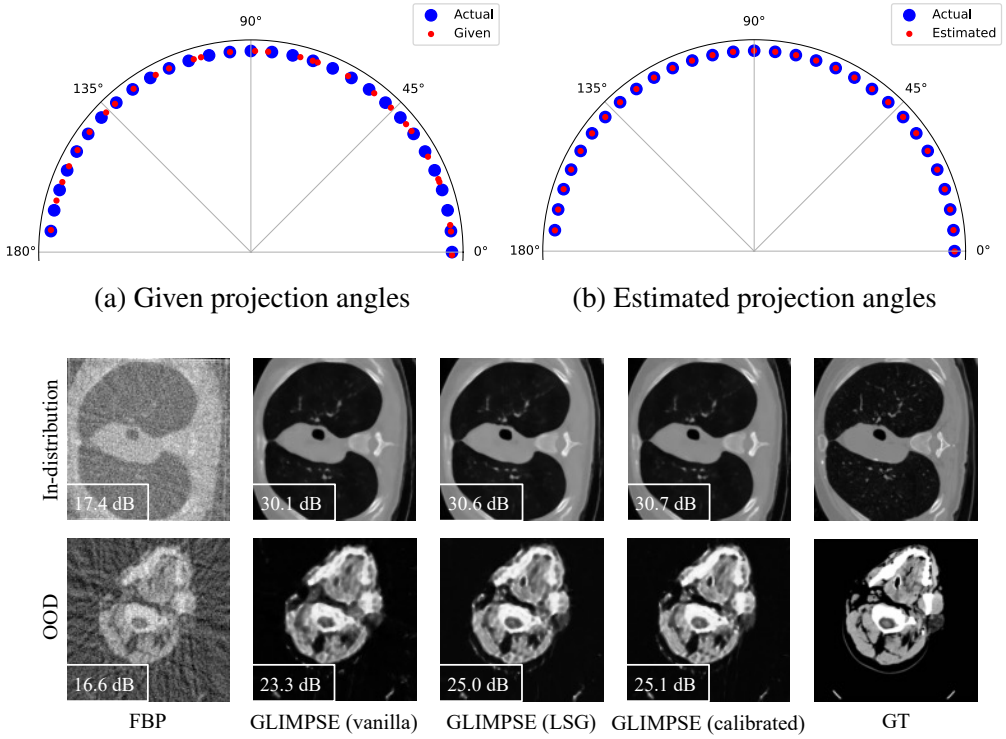
GLIMPSE could be integrated with various imaging problems involving line integrals in forward operators such as photoacoustic [143, 144] and cryo-electron tomography (cryoET) [145, 146]. Its future full-3D adaptation may yield efficient architectures that resolve fundamental memory issues with applications of deep learning in 3D medical imaging. This extension is particularly interesting given the ability of GLIMPSE to operate locally and its near-fixed memory requirement across resolution, which makes it an ideal choice for large 3D objects.

3.5 Appendix

3.5.1 Network architecture and training details

For GLIMPSE architecture, we use an MLP network comprising 9 hidden layers, each with dimensions [256, 256, 256, 256, 128, 128, 128, 64, 64] with ReLu activations. The input to the MLP network consists of sinusoidal curves sampled from $K = 9^2$ neighboring pixels. To prevent edge artifact of the circular convolution, we apply zero-padding with a size of 512 to the sinogram before applying the filter $\mathbf{h}$. Linear interpolation is used in (3.7). For the experiment in resolution 512×512 in Section 3.3.2, we use a larger network with hidden layer dimensions [1024, 1024, 1024, 512, 512, 512, 256, 256] to enhance the quality of reconstructions.

We implement our model in PyTorch [96] on a machine equipped with a Nvidia A100 GPU with 80GB of memory to train the different architectures. We report the maximum capacity of the graphics card during training and the time needed to complete the training. All models in Section 3.3 were trained for 200 epochs with MSE loss using the Adam optimizer [97]. A learning rate 10^{-4} was used for GLIMPSE and U-Net, while LGS and LPD were trained with a learning rate 10^{-3}. All models were trained with batch size 64. In the case of GLIMPSE, for each mini-batch of random objects, we performed optimization on a random mini-batch of 512 pixels 3 times.



(a) Given projection angles

(b) Estimated projection angles

(c) Reconstructions

Figure 3.7: Estimated sensor geometry by GLIMPSE (LSG) and reconstructions for an uncalibrated system with a random sensor shift; as expected, the learnable sensor geometry can effectively learn the projection angles and exhibits excellent robustness with no degradation under such a big model mismatch and measurement noise (30dB).

3.5.2 Learned sensor geometry

CT imaging algorithms such as FBP [114], SART [117], LGS [107], LPD [108] assume that the projection angles $\{\alpha_m\}_{m=1}^r$ are known. In an uncalibrated system where sensor geometry is different from what the algorithms assume, the quality of reconstruction deteriorates [118, 147]. GLIMPSE allows directly optimizing the projection angles during training. We thus jointly optimize $\{\alpha_m\}_{m=1}^r$ with other trainable parameters in (3.10). This additional angle estimation incurs a very modest computational cost.

In the absence of calibration, we cannot expect to have paired ground truth images. In the following experiments, we only want to showcase the possibility to differentiably optimize over angles in GLIMPSE so we assume having access to paired data (while simulating the uncalibrated forward operator). In practice, we could use a self-supervised loss, for example, based on equivariance [148].

We assess the performance of GLIMPSE in situations with mismatched projection orientations. In the following experiments, we place $r = 30$ sensors uniformly around the

object at angles $\alpha = 0°, 6°, ..., 174°$. We conduct a comparative analysis of three models: 1) GLIMPSE (vanilla), with no learnable sensor geometry, 2) GLIMPSE (LSG), incorporating the proposed learned sensor geometry, and 3) GLIMPSE (calibrated), operating under ideal conditions with no model mismatch (informed with correct projection angles). We let the GLIMPSE (LSG) learn the projection angles from the training data where the optimized values $\{\alpha_m\}_{i=1}^r$ obtained through training can provide a reliable estimate of the actual projection angles.

Uncalibrated system with random sensor shifts: As shown in Figure 3.7a, we randomly perturb projection angles by a normally distributed error so that $\alpha_i^{\text{given}} = \mathcal{N}(\alpha_i, \sigma^2)$; we set $\sigma = 2°$. We initialize the projection angles $\{\alpha_m\}_{i=1}^r$ in the GLIMPSE (LSG) architecture with α_i^{given}. Figure 3.7b shows the estimated projection angles obtained through training—GLIMPSE (LSG) accurately recovers the angles even in the presence of 30 dB measurement noise. As shown in Figure 3.7c, this accurate estimation of projection angles results in high-quality reconstructions by GLIMPSE (LSG) comparable with the network trained in an ideal calibrated system.

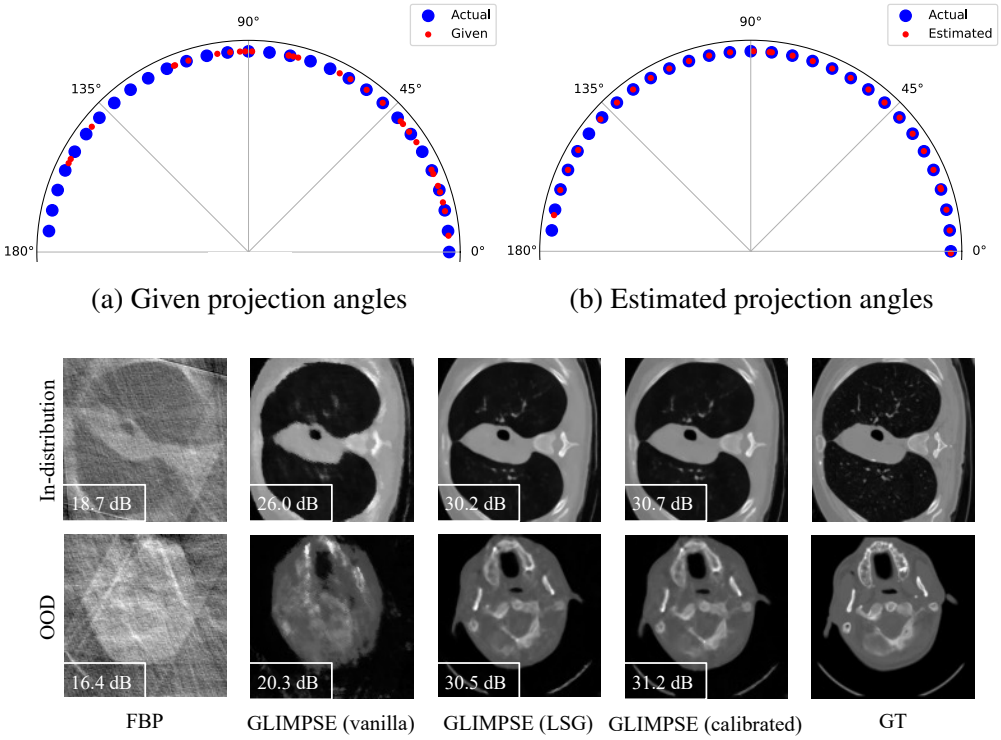
Blind inversion with no information from projection angles: We consider the blind scenario where the model operates without any prior knowledge of the sensor geometry making inversion challenging. As shown in Figure 3.8a, we initialize the projection angles $\{\alpha_m\}_{i=1}^r$ in the GLIMPSE (LSG) architecture with random values. The estimated projection angles are shown in Figure 3.8b, highlighting GLIMPSE (LSG)'s ability for data-driven sensor geometry estimation. Figure 3.8c presents the reconstructions achieved by GLIMPSE in both its vanilla and LSG versions. As expected, FBP and the GLIMPSE (vanilla) show poor reconstructions due to the missing sensor geometry information. On the other hand, GLIMPSE (LSG) could accurately reconstruct both in-distribution and OOD samples. Remarkably, these results are comparable to those achieved by the calibrated GLIMPSE with informed projection angles.

3.5.3 Optimal filter for FBP reconstruction

Proposition 3.5.1 (Reconstruction for continuous Radon transform)**.** *We have the following identity*

$$f(x, y) = \int_0^{\pi} Rf(\theta, \cdot) \star \psi d\theta,$$

where ψ is the filter that has for Fourier transform $|\cdot|$.



(a) Given projection angles

(b) Estimated projection angles

(c) High-quality reconstructions by GLIMPSE (LSG) despite having no information from sensor geometry.

Figure 3.8: Estimated sensor geometry by GLIMPSE (LSG) and reconstructions for blind inversion; GLIMPSE (LSG) was initialized with random projection angles $\{\alpha_m\}_{i=1}^{r}$ (a) could reliably estimate the projection angles purely from data (b) resulting in high-quality reconstructions (c).

Proof. Let $\mathbf{p} = (x, y)$, $\xi = (\xi_1, \xi_2)$. We have

$$\begin{aligned} f(x,y) &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mathcal{F}_{2D}(f)(\xi_1,\xi_2)\exp(2i\pi\langle\xi,\mathbf{p}\rangle)d\xi \\ &= \int_{0}^{+\infty}\int_{0}^{2\pi} \mathcal{F}_{2D}(f)(r\cos(\theta), r\sin(\theta))\exp(2i\pi r\langle\mathbf{k},\mathbf{p}\rangle)rdrd\theta, \end{aligned}$$

by doing a change of variable in polar coordinates, where $\mathbf{k} = (\cos(\theta), \sin(\theta))$. Observe that $\mathcal{F}_{2D}(f)(r\cos(\theta), r\sin(\theta))$ is the Fourier Transform of f along the line of direction $\mathbf{k}$. By the Fourier slice theorem [130], we have

$$\mathcal{F}_{2D}(f)(r\cos(\theta), r\sin(\theta)) = \mathcal{F}_{1D}(Rf(\theta,\cdot))(r)$$

By symmetry of the Radon transform, we have $Rf(\theta, r) = Rf(\theta + \pi, -r)$. Finally,

$$\begin{aligned} f(x,y) &= \int_{-\infty}^{+\infty}\int_{0}^{\pi} \mathcal{F}_{1D}(Rf(\theta,\cdot))(r)\exp(2i\pi r\langle\mathbf{k},\mathbf{p}\rangle) \\ &\qquad |r|drd\theta = \int_{0}^{\pi} \mathcal{F}_{1D}^{-1}\left(\mathcal{F}_{1D}(Rf(\theta,\cdot)) \odot |\cdot|\right) d\theta. \end{aligned}$$

This shows that

$$f(x,y) = \int_{0}^{\pi} \left(Rf(\theta,\cdot) \star \psi\right)(\langle\mathbf{k},\mathbf{p}\rangle)d\theta,$$

where ψ is the filter that has for Fourier transform $|\cdot|$.

□

3.5.4 Proof of proposition 3.2.1

Proof. Using the definition of the radon transform in (3.2), we have

$$\begin{aligned} Rf(\alpha,t) = \int_{-\infty}^{+\infty} &\delta(z\cos(\alpha) - t\sin(\alpha) - x, \\ &z\sin(\alpha) + t\cos(\alpha) - y)dz. \end{aligned}$$

Solving $z\cos(\alpha) - t\sin(\alpha) - x = 0$ for z leads to

$$z = \frac{t\sin(\alpha) + x}{\cos(\alpha)}.$$

Then, solving $z\sin(\alpha) + t\cos(\alpha) - y = 0$ for t, using the previous expression for z leads to

$$t = y\cos(\alpha) - x\sin(\alpha).$$

□

Chapter 4

FunkNN: Neural Interpolation for Functional Generation

In Chapters 2 and 3, we proposed a coordinate-based image reconstruction pipeline for solving various imaging problems. This chapter explores another significant application of our proposed coordinate-based network: continuous image generators. By integrating our continuous image synthesis pipeline with off-the-shelf generative models, we develop a functional generator capable of producing images at any arbitrary coordinate. We leverage this functional generator as a strong prior for solving PDE-based inverse problems.

Deep generative models are effective image priors in applications from ill-posed inverse problems [66, 149] to uncertainty quantification [23] and variational inference [150]. Since they approximate distributions of images sampled on discrete grids they can only produce images at the resolution seen during training. But natural, medical, and scientific images are inherently continuous. Generating continuous images would enable a single trained model to drive downstream applications that operate at arbitrary resolutions. If this model could also produce exact spatial derivatives, it would open the door to generative regularization of many challenging inverse problems for partial differential equations (PDEs).

There has recently been considerable interest in learning grid-free image representations. Implicit neural representations [28, 151–153] have been used for mesh-free image representations in various inverse problems [28, 31, 32, 35, 154, 155]. An implicit network $f_\theta(\mathbf{x})$, often a multi-layered perceptron (MLP), directly approximates the image intensity at spatial coordinate $\mathbf{x} \in \mathbb{R}^D$. While $f_\theta(\mathbf{x})$ only represents a single image, different works incorporate a latent code $\mathbf{z}$ in $f_\theta(\mathbf{x}, \mathbf{z})$ to model *distributions* of continuous images.

These approaches perform well on simple datasets but their performance on complex data like human faces is far inferior to that of conventional grid-based generative models based on convolutional neural networks (CNNs) [32, 38, 154]. This is in fact true even when evaluated at the resolution they were trained on. One reason for their

limited performance is that these implicit models use MLPs which are not well-suited for modelling image data. In addition, unlike their grid-based counterparts, these implicit generative models suffer from a significant overhead in the form of a separate encoding hyper-network [32, 35, 38] or parameter training [154] to obtain latent codes $\mathbf{z}$.

In this chapter, we alleviate the above challenges with a new mesh-free convolutional image generator that can faithfully learn the distribution of continuous image functions. The key feature of the proposed framework is our local continuous super-resolution network—FunkNN— which takes a discrete image at any resolution and super-resolves it to generate image intensities at arbitrary spatial coordinates. As shown in Figure 4.1), our approach combines a traditional discrete image generator with FunkNN, resulting in a deep generative model that can produce images at arbitrary coordinates or resolution. FunkNN can be combined with *any* off-the-shelf pre-trained image generator (or trained jointly with one). This includes the highly successful GAN architectures [156, 157], normalizing flows [21, 70] or diffusion models [71, 79]. It naturally enables us to learn complex image distributions.

Unlike prior works [32, 35, 38, 154], FunkNN neither requires a large encoder to generate latent codes nor does it use any MLPs. This is possible thanks to FunkNN's unique way of integrating the coordinate $\mathbf{x}$ with image features. The key idea is that resolving image intensity at a coordinate $\mathbf{x}$ should only depend on its neighborhood. Therefore, instead of generating a code for the entire image and then combining it with $\mathbf{x}$ using an MLP, FunkNN simply crops a patch around $\mathbf{x}$ in the low-resolution image obtained from a traditional generator. This window is then provided to a small convolutional neural network that generates the image intensity at $\mathbf{x}$. The window cropping is performed in a differentiable manner with a spatial transformer network [158].

We experimentally show that FunkNN reliably learns to resolve images to resolutions much higher than those seen during training. In fact, it performs comparably to state-of-the-art continuous super-resolution networks [35] despite having only a fraction of the latter's trainable parameters. Unlike traditional learning-based methods, our approach can also super-resolve images that belong to image distributions different from those seen while training. This is a benefit of patch-based processing which reduces the chance of overfitting on global image features. In addition, we show that our overall generative model framework can produce high quality image samples at any resolution. With the continuous, differentiable map between spatial coordinates and image intensities, we can access spatial image derivatives at arbitrary coordinates and use them to solve inverse problems.

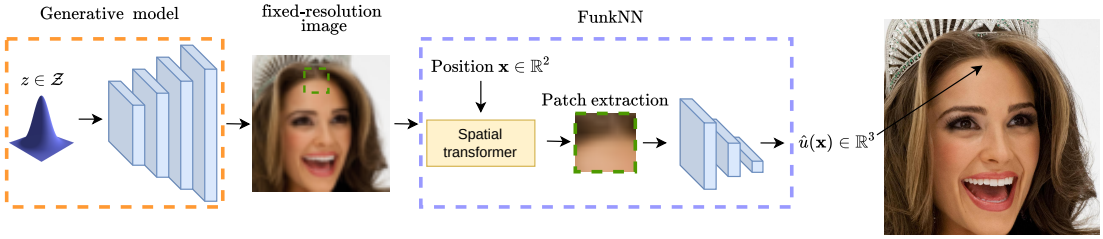


Figure 4.1: The proposed architecture. The generative model (orange) produces a fixed-resolution image that is differentiably used by FunkNN to produce the image intensity at any location (blue).

4.1 Implicit neural representations for continuous image representation

Let $u \in L^2(\mathbb{R}^D)^c$ denote a continuous signal of interest with $c \geq 1$ channels and $\mathbf{u}$ be its discretized version supported along a fixed grid with $n \times n \in \mathbb{N}^*$ elements. For simplicity, we consider the case with $D = 2$. Our discussion naturally extends to higher dimensions.

An implicit neural representation approximates u by $f_\theta : \mathbb{R}^2 \to \mathbb{R}$ parameterized by weights $\theta \in \mathbb{R}^K$. In standard approaches, the weights are obtained by solving

$$\theta^\star(u) = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n^2} \| f_\theta(\mathbf{x}_i) - \mathbf{u}(\mathbf{x}_i) \|_2^2, \tag{4.1}$$

where $\mathbf{u}(\mathbf{x}_i)$ is the image target value sampled at spatial coordinate $\mathbf{x}_i$. This way, implicit neural representations can provide a continuous interpolation for $\mathbf{u}$. Notice, however, that the properties of this interpolation scheme depend on the choice of the network architecture and do not exploit the statistics of given images.

From (4.1), we can observe that the obtained weights $\theta^\star(u)$ apply only to a single image $\mathbf{u}$, thereby making $f_\theta(\mathbf{x})$ incapable of providing interpolating representations for a class of images. Some prior works address this by incorporating a low dimensional latent code $\mathbf{z}$ in the implicit network. The new representation then is given by the map $\mathbf{x} \mapsto f_\theta(\mathbf{x}, \mathbf{z})$, where each $\mathbf{z}$ represents a different image. Note that this framework often entails an additional overhead to relate the images with the latent codes. For example, [32] and [38] jointly train f_θ with an encoder network to generate the latent codes for images. [154], on the other hand, optimize the codes as trainable parameters. Figure 4.2 shows a general illustration of implicit generative networks.

The prior approaches to model continuous image distributions often exhibit poor performance in comparison to their grid-based generative counterparts [32, 38]. This is due to their reliance on MLP networks which a priori do not possess the right inductive biases to model images.

4.2 Our approach

To address the challenges discussed in Section 4.1 we propose a convolutional image generator that produces continuous images. As shown in Figure 4.1, the proposed framework relies on FunkNN, our new continuous super-resolution network which interpolated images to arbitrary resolution. Our approach first samples discrete fixed-resolution images from an expressive *discrete* convolutional generator. This is followed by continuous interpolation with FunkNN. We combine FunkNN with any state-of-the-art pre-trained image to learn complex image distributions.

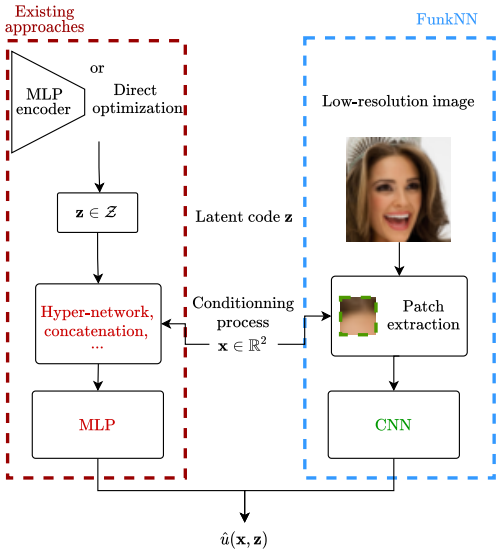


Figure 4.2: Conceptual difference between FunkNN and existing implicit neural representations.

The key advantage of FunkNN is its architectural simplicity and interpretability from a signal processing perspective. The main idea is that super-resolving a sharp feature from a low-resolution image at a spatial coordinate $\mathbf{x}$ should only use information from the neighborhood of $\mathbf{x}$. This is because high frequency structures like edges are spatially localized and have small correlations with pixels at far-off spatial locations in the low-resolution image. FunkNN, therefore, selects a small patch around coordinate $\mathbf{x}$ and passes it to a small CNN to regress image intensity at $\mathbf{x}$. Figure 4.2 illustrates the differences between FunkNN and prior approaches. In the following sections, we provide a formal description of FunkNN and our continuous image generative framework.

4.2.1 FunkNN: continuous super-resolution network

Given a low-resolution image $\mathbf{u}_{\text{lr}} \in \mathbb{R}^{d \times d \times c}$, FunkNN estimates the image intensity at a spatial coordinate $\mathbf{x} \in \mathbb{R}^2$ by a two-step process: i) it first extracts a $p \times p$ patch $\mathbf{w} \in \mathbb{R}^{p \times p \times c}$ from $\mathbf{u}$ around the coordinate $\mathbf{x}$, and then ii) a CNN maps the patch to the estimated intensity. More formally, if we denote $\text{ST} : (\mathbf{x}, \mathbf{u}) \mapsto \mathbf{w}$ to be a patch extraction module and $\text{CNN}_\theta : \mathbb{R}^{p \times p \times c} \to \mathbb{R}^c$ to be a CNN block that estimates pixel intensity at $\mathbf{x}$ from $\mathbf{w}$, then FunkNN can be characterized as

$$\text{FunkNN}_\theta \stackrel{\text{def.}}{=} \text{CNN}_\theta \circ \text{ST}.$$

While we could extract a patch in the ST module by trivially cropping it from the image, doing so would not give access to derivatives of image intensities with

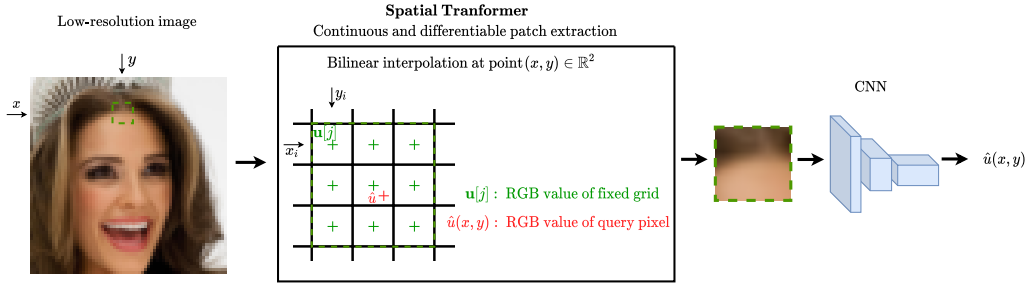


Figure 4.3: FunkNN architecture. Given a location (x, y), the Spatial Transformer extracts a patch in the discrete image and a CNN produces the intensity at the query position.

respect to spatial coordinates, a property of importance for PDE-based problems. We address this by modelling ST with a spatial transformer network [158]. As illustrated in Figure 4.3, spatial transformers map the low-resolution image and input coordinate to the cropped patch in a continuous manner, thereby preserving differentiability. These derivatives can, in fact, be exactly computed using auto-differentiation in PyTorch [96] and TensorFlow [159]. We evaluate the accuracy of these derivatives in Figure 4.13 in the Appendix 4.7.6. This experiment clearly shows the effectivity of FunkNN for approximating the first-order and second-order derivatives, crucially important for solving PDE-based inverse problems. Another advantage of the ST module over trivial cropping is that it allows us to learn the effective receptive field of the patches during training, thereby providing better performance. For further details, please refer to the Appendix 4.7.2.

Training FunkNN

Let $(\mathbf{u}_m)_{1\leq m\leq M}$ denote a dataset of M discrete images with the maximum resolution $n \times n$. We generate low-resolution images $(\mathbf{u}_{\text{lr},m})_{1\leq m\leq M}$ from this dataset where $\mathbf{u}_{\text{lr},m} \in \mathbb{R}^{d\times d\times c}$ and $d < n$. FunkNN is then trained to super-resolve $(\mathbf{u}_{\text{lr},m})$ with $(\mathbf{u}_m)$ as targets using a patch-based framework. In particular, from each low-resolution image, small $p \times p$ sized patches centered at randomly selected coordinates are extracted and provided to FunkNN which then regresses the corresponding intensity at the coordinates in the high-resolution image. We optimize the weights of the CNN as follows.

$$\underset{\theta\in\mathbb{R}^K}{\operatorname{argmin}} \sum_{m=1}^{M} \sum_{i=1}^{n^2} \left|\text{FunkNN}_\theta(\mathbf{x}_i, \mathbf{u}_{\text{lr},m}) - u_m(\mathbf{x}_i)\right|^2 . \tag{4.2}$$

Note that since FunkNN takes patches as inputs, it can be used to simultaneously train on images with different sizes. In addition, it allows us to train with mini-batches on both images and sampling locations, thereby enabling memory-efficient training.

We consider three strategies to train our model, namely (i) *single*, (ii) *continuous* and (iii) *factor*. In *single* mode, FunkNN is trained with input images of fixed resolution, eg. $d = \frac{n}{2}$. The *continuous* mode uses low-resolution input images at different scales, i.e. $d = \frac{n}{s}$, where the scale s can vary between $[s_{\min}, s_{\max}]$. In *factor* mode, the super-resolution factor, s, between the low and high-resolution images is always fixed and the low-resolution size d is varied in the range $d_{\min}$ and $d_{\max}$. The high-resolution target size is then correspondingly chosen as $ds \times ds$. At test time, we can then use this model to super-resolve images by a factor s in a hierarchical manner. For instance, with input image $\mathbf{u}_{hr}^{(0)} = \mathbf{u}$ of size $d \times d$, it can iteratively generate high-resolution images $(\mathbf{u}_{hr}^{(i)})_{i \geq 1}$ of sizes $(d_i)_{i \geq 1}$ satisfying $\mathbf{u}_{hr}^{(i)} = \mathrm{FunkNN}_\theta(\mathbf{u}_{hr}^{(i-1)})$ and $d_i = s^i d$.

4.2.2 Generative network

To generate continuous samples, we can combine FunkNN with *any* fixed-resolution convolutional generator. This flexibility allows our overall model to learn complex continuous image distributions, and to produce images and their spatial derivatives at arbitrary resolutions. More formally, let $G : \mathbb{R}^L \to \mathbb{R}^{d \times d \times c}$ be a generative prior on low-resolution images of size $d \times d$ that takes as input a latent code of size L. Then, G can be combined with FunkNN_θ to yield a continuous image generator $\mathrm{FunkNN}_\theta \circ G$.

The choice of fixed-sized generator we use prior to FunkNN depends on the downstream application. Generative adversarial networks (GANs) [156] and normalizing flows [70] are popular high-quality image generators. However, the former exhibits challenges when used as generative prior for inverse problems [21, 66] and the latter have very large memory requirements and are expensive to train [21, 67, 160]. Injective encoders [21] were recently proposed to alleviate these challenges by using a small network and latent space. However, since we do not necessarily require injectivity in our applications of focus, we use a simpler architecture inspired from [21] for faster training. In particular, we use a conventional auto-encoder to obtain a latent space for our training images and use a low-dimensional normalizing flow to learn this latent space distribution. The standard practice of using mean squared loss as suggested in [21, 161] results in a loss of high frequency components in our reconstructions. We remedy this by using a perceptual loss [162]. Further details on network architecture and training are given in Appendix 4.7.1. It is worth noting that if generative modelling is not aimed, we can train FunkNN independently of any generator as shown in Section 4.4.1.

4.3 Continuous generative models and solving inverse problems

Continuous generative models built using FunkNN can be used for image reconstruction at resolutions much higher than those seen during training. This is particularly useful when solving ill-posed inverse problems commonly found in experimental sciences like seismic [163], medical [164] and 3-D molecular imaging [165]. Due to difficulty and high cost of measurement acquisition, obtaining training data is challenging in these applications. Therefore, new models can not readily be trained when faced with measurements acquired at resolution different from the ones previously seen during training.

We consider three inverse problems: reconstructing an image given its (i) first derivatives, (ii) when a fraction of derivatives are known, and (iii) limited-view computed tomography (CT). In the following, we assume that we have low-resolution generative prior on $d \times d$-pixel images, $G : \mathbb{R}^L \to \mathbb{R}^{d \times d \times c}$, which takes as input a latent code $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$.

4.3.1 Inverting derivatives

We first apply FunkNN to reconstruct continuous image u from its spatial derivatives $\nabla_{\mathbf{x}} u(\mathbf{x}_j)$ in coordinates $\{\mathbf{x}_j\}_{j=1}^{n^2}$. Related ill-posed reconstructions appear in different domains such as computer vision [154] or obstacle scattering [31]. We use the exact spatial derivatives provided by FunkNN and a pre-trained generative model to obtain the latent code $\mathbf{z}^\star$ that gives us output aligned with the given derivatives,

$$\mathbf{z}^\star = \operatorname*{argmin}_{\mathbf{z}} \sum_{j=1}^{n^2} \|\nabla_{\mathbf{x}} \mathrm{FunkNN}_\theta(\mathbf{x}_j, G(\mathbf{z})) - \nabla_{\mathbf{x}} u(\mathbf{x}_j)\|_2^2 + \lambda \|\mathbf{z}\|_2^2. \tag{4.3}$$

We also found that updating the weights θ of the generative model further helps in producing better reconstruction as suggested in [166]. The exact procedure is detailed in Appendix 4.7.4. The estimated reconstruction is given by sampling the trained FunkNN network on the high-resolution grid: $\hat{u}(\mathbf{x}_j) = \mathrm{FunkNN}_\theta(\mathbf{x}_j, G(\mathbf{z}^\star))$.

4.3.2 Sparse derivatives

We now make the previous problem even more ill-posed by observing not the spatial derivatives on a dense grid, but only 20% of them that have the highest intensity. It amounts to only sample the gradient on the $n_s < n^2$ locations that gives the largest value $\|\nabla_{\mathbf{x}} u(\mathbf{x}_j)\|$. In order to account for the missing low-amplitude gradient, we add an extra

total-variation regularization term. Thus, we aim at solving

$$\mathbf{z}^\star = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{j=1}^{n_s} \|\nabla_{\mathbf{x}} \mathrm{FunkNN}_\theta(\mathbf{x}_j, G(\mathbf{z})) - \nabla_{\mathbf{x}} u(\mathbf{x}_j)\|_2^2 + \lambda \|\mathbf{z}\|_2^2 + \lambda_2 \|\nabla G(\mathbf{z})\|_2, \tag{4.4}$$

using the same process as in the previous setting (see Appendix 4.7.4).

4.3.3 Limited-view CT

In CT, we observe projections of a 2-dimensional image $u \in L^2(\mathbb{R}^2)$ through the (discrete angle) Radon operator $\mathcal{A}(\alpha) : L^2(\mathbb{R}^2) \to L^2(\mathbb{R})^I$ where $\alpha = \{\alpha_1, \ldots, \alpha_I\}$ denotes the I viewing directions with $\alpha_i \in [0, 2\pi)$. Finally, the discrete observation is given by

$$\mathbf{v}_i = (\mathcal{A}(\alpha_i)u)(\mathbf{x}) + \eta_i, \tag{4.5}$$

where $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_{n^2}]$ is the $n \times n$ sampling grid and $\eta_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \mathbf{I})$ is a random perturbation. If the view directions correspond to a fine sampling of the interval $[0, 2\pi)$, then the adjoint operator $\mathcal{A}^\star$, namely the filtered back-projection (FBP) operator, allows to recover the original signal even in the presence of noise. For application-specific reasons it is not always possible to observe the full viewing direction set, leading to limited-view CT. Here we sample (α_i) uniformly between -70 degrees and $+70$ degrees. This incomplete set of observations makes the estimation of the signal u from (4.5) strongly ill-posed. We address limited-view CT using the FunkNN framework:

$$\mathbf{z}^\star = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{I} \|\mathcal{A}(\alpha_i)\mathrm{FunkNN}_\theta(\mathbf{x}, G(\mathbf{z})) - \mathbf{v}_i\|_2^2 + \lambda \|\mathbf{z}\|_2^2. \tag{4.6}$$

4.4 Experiments

4.4.1 Super-resolution

We first evaluate the continuous, arbitrary-scale super-resolution performance of FunkNN. We work on the CelebA-HQ [91] dataset with resolution 128×128, and we train the model using three training strategies explained in Section 4.2.1, *single*, *continuous* and *factor*. More details about network architecture and training details are provided in Appendix 4.7.2.

The model *never sees training data at resolutions 256 × 256 or higher*. Figure 4.4 shows the performance of FunkNN trained using *factor* strategy, which indicates that

Figure 4.4: FunkNN model is only trained over CelebA-HQ images of maximum resolution 128×128, but it can reliably reconstruct high-quality images in higher resolutions.

FunkNN yields high-quality image reconstructions at much higher resolutions than it was trained on. This experiment shows that FunkNN learns an accurate super-resolution operator and generalizes to resolutions not seen during training. Table 4.1 shows that our model demonstrates comparable performance to the state-of-the-art baseline LIIF-EDSR [35] but with a considerably smaller network; our model only has 140K trainable parameters while LIIF uses 1.5M parameters. Additionally, we provide a comparison with LIIF by pruning its trainable parameters to reach 140k. The results suggest that while FunkNN obtains comparable results to the original LIIF architecture, reducing its parameters to be closer to FunkNN deteriorates the results. This experiment emphasizes the capacity of FunkNN to perform similarly to state-of-the-art methods while being conceptually simpler. It is worth mentioning that on the same Nvidia A100 GPU, the average time to process 100 images of size 128×128 is 1.97 seconds for LIIF, 0.81 seconds for LIIF with 140k parameters and 0.7 seconds for FunkNN.

4.4.2 Robustness to out-of-distribution data

One exciting aspect of the proposed model is its strong generalization to out-of-distribution (OOD) data–a boon obtained from FunkNN's patch-based operation. We showcase it by evaluating FunkNN originally trained on CelebA-HQ [91] at resolution 128×128 (Section 4.4.1), to super-resolve images from the LSUN-bedroom [167] dataset which are structurally very different from the training data. Moreover, we super-resolve images from size 128 to 256, which were not seen at training time. Figure 4.5 indicates that the proposed model can faithfully super-resolve OOD images at new resolutions. We compare our approach with U-Net [25] which has shown promising results for super-resolution; however, it can only be trained to map between two fixed resolutions. We

Figure 4.5: Image super-resolution on 256×256 LSUN images with the model trained on 128×128 images from CelebA-HQ data.

trained a U-Net to superresolve from 128×128 to 256×256; it was thus trained on larger images than other methods. Nevertheless, Table 4.1 shows that while the U-Net performs well on inlier samples, it gives rather bad results for OOD data. We also report the shift-invariant SNRs [168] in Table 4.2 to compensate for the fact that U-Net is unable to recover the correct amplitudes for OOD samples.

Table 4.1: Performance of different methods over super-resolution in SNR (dB); the SNRs are averaged over 100 test samples of CelebA-HQ (inliers) or LSUN-bedroom (outliers).

	$128 \to 256$	$128 \to 512$	$128 \to 1024$	$128 \to 256$ (OOD)	Trainable params
Bilinear Interpolation	27.5	24.9	22.4	22.9	-
U-Net [25]	31.3	-	-	19.9	8800K
LIIF-EDSR [35]	31.6	28.0	23.9	**28.0**	1500K
LIIF-EDSR [35] (140K param.)	31.6	27.5	23.8	26.7	140K
FunkNN (single)	31.2	26.3	22.5	27.2	140K
FunkNN (continuous)	30.7	27.7	23.5	26.4	140K
FunkNN (factor)	**32.6**	**28.2**	**24.2**	26.6	140K

4.4.3 Inverse problems

We solve equation 4.3 using the generative framework described in Section 4.3 trained over CelebA-HQ samples to reconstruct an image from its spatial derivatives. Experimental settings are provided in Appendix 4.7.4. Figure 4.6 illustrates FunkNN's performance, which provides high-quality spatial reconstructions from the observation of the spatial derivatives. Notice that Problem (4.3) is ill-posed and it is not possible to accurately estimate mean intensity of the original image only based on the derivatives. Despite the different color scales, the unknown image is accurately estimated by FunkNN.

We solve equation 4.4 with the same procedure but over LoDoPaB-CT samples [137]. We compare FunkNN with the continuous GAN proposed by [38]. We solve the problem described in Equation (4.4) but replacing our model by their GAN. The training procedure is described in Appendix 4.7.3. Figure 4.7 shows that FunkNN significantly outperforms the continuous GAN; while we observe an accurate estimation of the image gradients by

Figure 4.6: Reconstructing an image from its spatial derivatives using the generative prior; FunkNN could faithfully reconstruct the image details.

GAN (first row, second column), its reconstruction (second row, second column) lacks a lot of details that were well retrieved by our method. The poor reconstruction of GANs as a prior for solving inverse problems has already been observed in [21, 160] and is confirmed by this experiment. Shift-invariant SNRs are reported in the bottom-left corner of each reconstruction. This example of highly under-determined system shows that the proposed FunkNN architecture can leverage existing prior in addition to super-resolving both an image and its derivative.

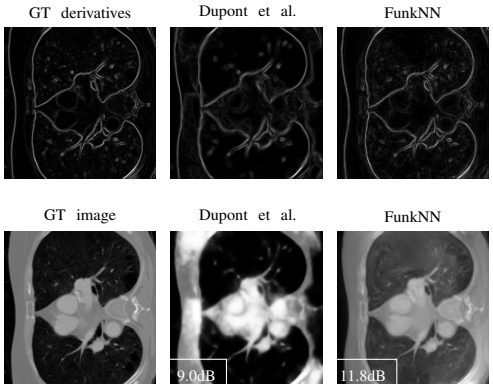


Figure 4.7: Solving PDE-based inverse problem with sparse derivatives.

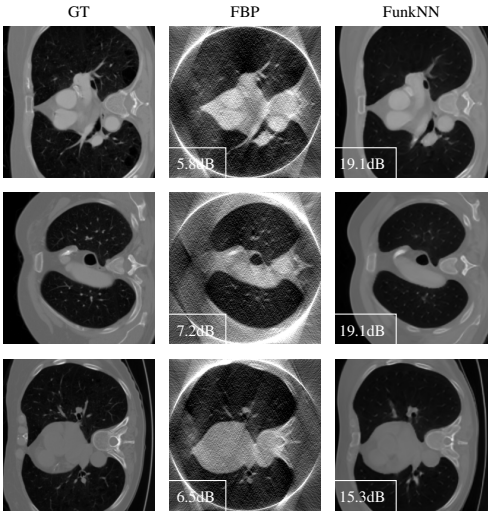


Figure 4.8: Limited-view CT reconstruction.

We repeat the same procedure and solve equation 4.6 to reconstruct an image from its limited-view sinogram. The precise experimental setting is relegated to Appendix 4.7.5. Figure 4.8 shows FunkNN's reconstructions compared to FBPs which confirms the power of the learned low-resolution prior in solving Problem (4.5).

4.5 Related works

4.5.1 Super-resolution

Various learning based approaches have been proposed over the years for image super-resolution. Patch-based dictionary learning [169] and random forests [170] have been used to learn a mapping between fixed low and high-resolution image pairs. More recently, deep learning methods based on convolutional neural networks have provided state-of-the-art performance on super-resolution tasks [13, 171–175]. Inspired by the popular U-Net [25], [27] proposed a multi-scale convolutional sparse coding based approach that performed comparably to U-Net on various tasks including super-resolution. Furthermore, [176] learned the interpolation kernel to design an implicit mapping from low to high-resolution samples.

[66] and [21] have used generative models as priors for image super-resolution using an iterative process. In addition, conditional generative models have been used to address ill-posedness of super-resolution [23, 177–179]. While the prior works mentioned above exhibit excellent super-resolution performance, they all have a limitation—these models are restricted to be used only at the image resolution seen during training. On the contrary, we focus on super-resolving images to arbitrary resolution.

In an attempt towards building continuous super-resolution models, [180] propose a new upsampling module, MetaSR, that adjusts its upscale weights dynamically to generate high-resolution images. While this approach exhibits promising results on in-distribution training scales, it has limited performance of out of distribution factors. The closest to our work is the network LIIF [35] that also uses local neighbourhoods around spatial coordinates for super-resolution. However, unlike FunkNN that directly operates on small image patches, LIIF first uses a large convolutional encoder to generate feature maps for the image. Image intensity at a coordinate $\mathbf{x}$ is then obtained by passing this coordinate along with features in its neighbourhood to an MLP. Note that FunkNN's approach of combining image features with spatial coordinates is architecturally much simpler than that of LIIF. It also requires less parameters, is slightly faster to evaluate and performs equivalently well.

4.5.2 Generative models for continuous signals

Generative models based on MLPs have been proposed in recent years to learn continuous image distributions. [32] use implicit representations to learn 3D shape distributions but lack expressivity to represent complex image data. [38] propose adverserial training based on MLPs and Fourier features [151] to represent different data modalities like faces, climate data and 3D shapes. While their approach is quite versatile, it still performs poorly compared to convolutional image generators [156, 157, 181].

As a step towards building expressive continuous image generators, [182] and [183] used positional encodings to make style-GAN generator [156] shift and scale invariant, thereby resulting in arbitrary scale image generation. [184, 185] proposed continuous GANs for effectively learning representations of real-world images. However, these models do not provide access to derivatives with respect to spatial coordinates, making them unsuitable for solving inverse problems.

In a related line of work, [186] propose to combine a cascade of a diffusion and fixed-size super-resolution models to generate images at higher resolution. Unlike FunkNN, they use fixed-size super-resolution blocks so their approach does not generate continuous images.

4.6 Summary

Recent continuous generative models rely on fully connected layers and lack the right inductive biases to effectively model images. In this chapter, we addressed continuous generation in two steps. (i) Instead of generating continuous images directly, we first used a convolutional generator pre-trained to sample discrete images from the desired distribution. (ii) We then used FunkNN to continuously super-resolve the discrete images given by the generative model. Our experiments demonstrated performance comparable to state-of-the-art networks for continuous super-resolution despite using only a fraction of trainable parameters.

4.7 Appendix

4.7.1 Generative model architecture

The generative networks used to address inverse problems described in Section 4.3 are trained in two-steps: i) we train an auto-encoder (AE) neural network to map the training samples to a lower dimensional latent code, ii) we train a normalizing flow model to learn the distribution of the computed latent codes by using maximum likelihood (ML) loss function.

Auto-encoder architecture

The AE network is a succession of an encoder and a decoder neural networks. The encoder maps images to a low dimensional latent code, a L-dimensional vector. It is a succession of 6 Conv+ReLu blocks (one convolution layer followed by a ReLu activation unit). Between each Conv+ReLu blocks, there is a down-sampling operation. The

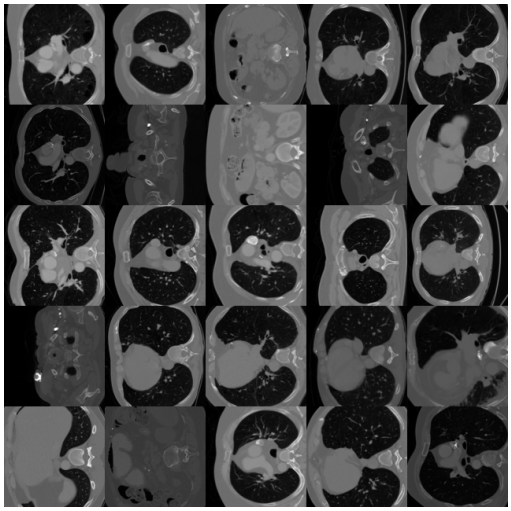
(a) Ground truth test samples (b) AE reconstructions

Figure 4.9: AE performance on CelebA-HQ in resolution 128×128.

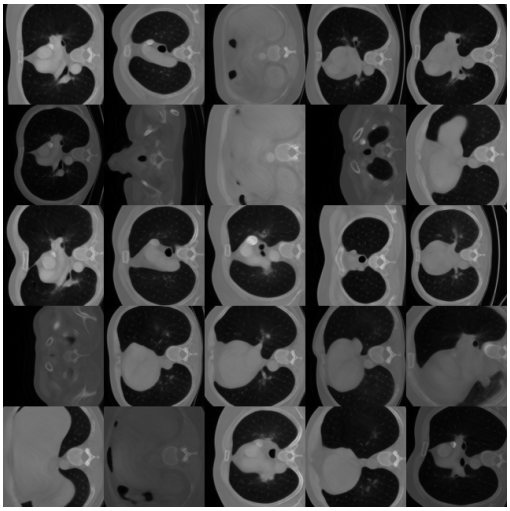
decoder network transforms the computed latent code to the image samples. The decoder has the same structure as the encoder network, except that the down-sampling operations are replaced by up-sampling layers. In order to increase the expressivity of the model, we equip our model with local skip connections in both encoder and decoder, as suggested in [51]. The latent code dimension is 512 for training data in resolution 128×128 for both RGB and gray-scale images. The AE model has around 40M and 11M trainable parameters for limited-view CT and CelebA-HQ experiments.

Training strategy While we can train the AE model using only the MSE loss [21,161], it often suppresses the high-frequency components of the reconstructed image. To alleviate this issue, we combine the perceptual loss proposed in [162] with regular MSE loss which significantly improves the quality of the reconstructions.

Experimental Result: We train AE model over 40000 images of the LoDoPaB-CT [137] and 30000 images of CelebA-HQ [91] datasets in resolution 128×128. We train the model over CT and CelebA-HQ images for 600 and 200 iterations. Figures 4.10 and 4.9 display the output image produce by the AE on the test dataset. Notice that very small details are lost by the AE. However, the final reconstruction contains the most important details. We show that this is enough to learn an informative prior and solve ill-posed inverse problems. We also show that it is possible to go beyond the quality of reconstruction given by the trained AE, see Section 4.3.3.



(a) Ground truth test samples



(b) AE reconstructions

Figure 4.10: AE performance on LoDoPaB-CT in resolution 128×128.

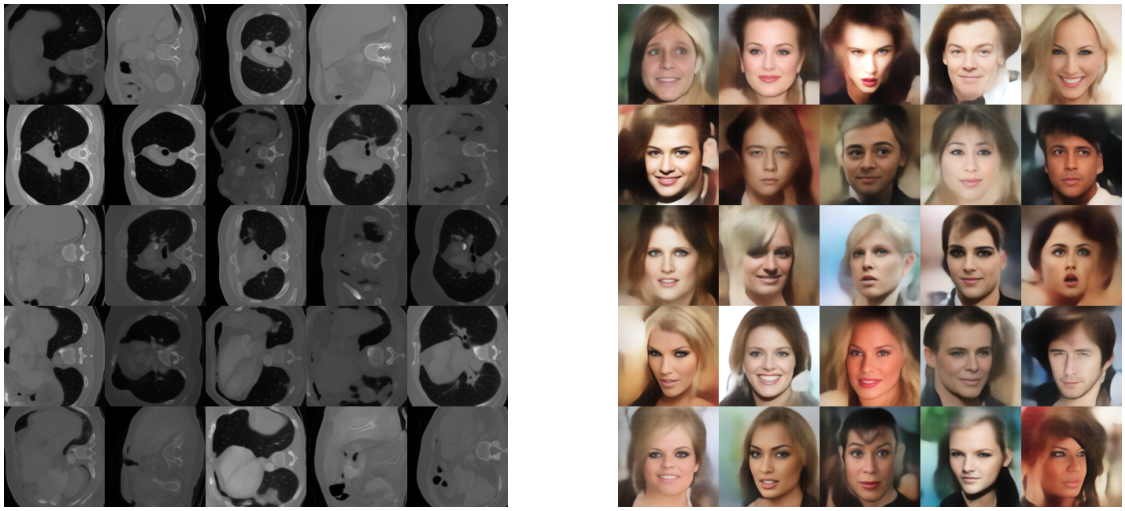
Normalizing flow

We use RealNVP [187] architecture with fully-connected layers for scale and bias sub-networks with two hidden layers of dimension 1024 and 512. The model comprises five masked affine coupling blocks and activation normalization. As mentioned earlier, we train the model using ML loss. The flow model has 10M trainable parameters.

Experimental Results: As soon as the flow model is trained, we take samples from the Gaussian distribution and feed to the flow model to generate new latent codes. Decoder then takes the generated latent codes and produce new images. Figure 4.11 demonstrates the generated samples by our generative model (flows + AE) for CT and CelebA-HQ dataset.

4.7.2 FunkNN architecture

We crop a patch with size $p = 9$ from the low-resolution image by using spatial transformer with *bicubic* interpolation kernel and *reflection* mode for the border pixels. We let two trainable parameters defined the size of the patch with respect to low-resolution image, i.e. the location of the 9 pixels around the query pixel are estimated during training. We use eight convolutional layers with filter size 2 and channel dimension 64 and ReLu activation functions. We use two maxpooling layers with size (2,2) inside to reduce the feature size. This downsampling layer if applied between 3 succesive applications of convolution+ReLu layers, so that the size of the input is devided by 4 in total in each direction. We use skip connections between each convolution layers. We then use four fully-connected layers with latent dimension 64 with skip connections between each



(a) CT (b) CelebA-HQ

Figure 4.11: Generative model performance on CT and CelebA-HQ samples in resolution 128×128.

Linear+Relu block. Finally, following inspiration from residual neural network, we add to the output the intensity of the pixel central to the input patch. We also use mini-batches with 512 pixels and 64 objects for each training iteration. All the models are trained using Adam optimizer [97] with learning rate 10^{-4}.

4.7.3 Dupont et al. architecture

We train the continuous generative model proposed in Dupont et al. [38] over the 40000 images of the LoDoPaB-CT dataset. We train their model over 100 epochs using the default parameters given in their public repository. We visually asses the quality of the generated sample in Figure 4.12.

4.7.4 Experimental details: PDE inverse problem

The training strategy of the networks is detailed in Appendix 4.7.1 for two datasets used in Section 4.4.3. The low-resolution images are composed of $d \times d = 128 \times 128$ pixels and the high-resolutions target images are of size $n \times n = 256 \times 256$. We use $\mathbf{z} = 0$ as the initialization as suggested by [160] and optimize for 2500 iterations using Adam optimizer over Problem (4.3) with $\lambda = 0$ and on Problem (4.4) with $\lambda = 0$ and $\lambda_2 = 10^{-2}$. Then, we run 1000 iterations of stochastic gradient descent to optimize the weights of the auto-encoder of the generative model as suggested in [166].

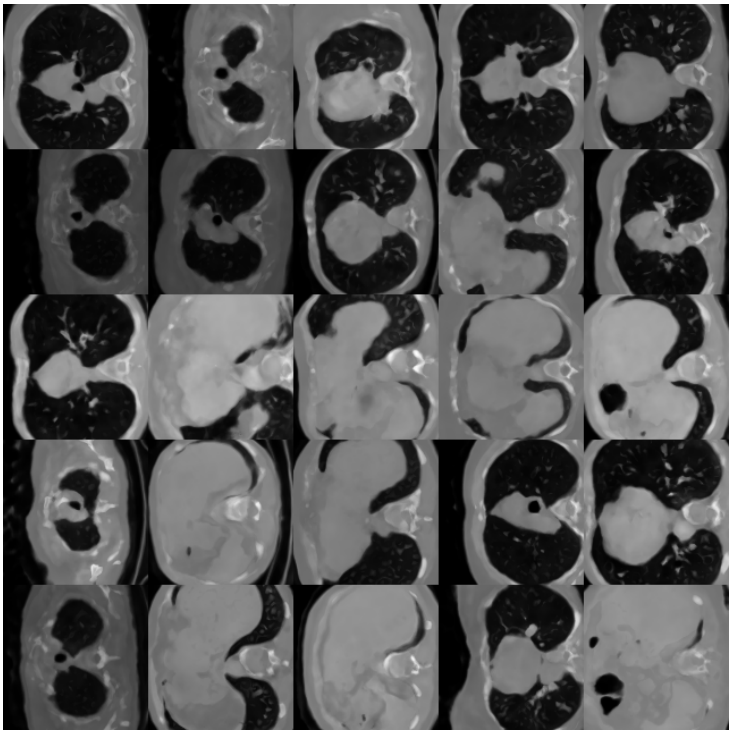


Figure 4.12: CT images generated by [38]

4.7.5 Experimental details: limited-view CT

The training strategy of the networks is detailed in Appendix 4.7.1. The low-resolution images are composed of $d \times d = 128 \times 128$ pixels and the high-resolutions target images are of size $n \times n = 256 \times 256$. We use $\mathbf{z} = 0$ as the initialization. The estimated solution is obtained by running 5000 iterations of stochastic gradient descent using Adam optimization algorithm on Problem (4.5) with $\lambda = 0$. The observations given by equation (4.5) are degraded by additive Gaussian noise such that the SNR of each projection is 30dB.

4.7.6 Experimental details: super-resolution

Accuracy of the derivatives provided by FunkNN

In order to quantify the quality of FunkNN derivatives, we train our network on images with intensity given by a Gaussian functional:

$$g(x, y) = \exp\left(-\frac{(x - x_0)^2 + (y - y_0)^2}{2\sigma^2}\right),$$

Table 4.2: Performance of different methods over super-resolution in scale-invariant SNR (dB)

	$128 \to 256$	$128 \to 512$	$128 \to 1024$	$128 \to 256$ (OOD)
Bilinear Interpolation	27.5	24.9	22.4	22.9
U-Net [25]	31.5	-	-	22.8
LIIF-EDSR [35]	31.6	28.0	24.0	**28.0**
LIIF-EDSR [35] (140k param.)	31.6	27.6	23.8	26.7
FunkNN (single)	31.4	26.4	22.6	27.2
FunkNN (continuous)	30.8	27.7	23.5	26.4
FunkNN (factor)	**32.6**	**28.2**	**24.2**	26.6

where the position of the center (x_0, y_0) is chosen at random in the field of view and the standard deviation σ is chosen uniformly at random in $[0.1, 0.4]$. We use cubic convolutional interpolation kernel [188] in the spatial transformer of the FunkNN trained using the training procedure described in 4.7.2. In the first row of Figure 4.13, from left to right, we display the test function g, the norm of its gradient and its Laplacian sampled on a discrete grid. in the second row, we display the same quantity estimated by the trained FunkNN as well as the SNR between the estimation and the ground truth.

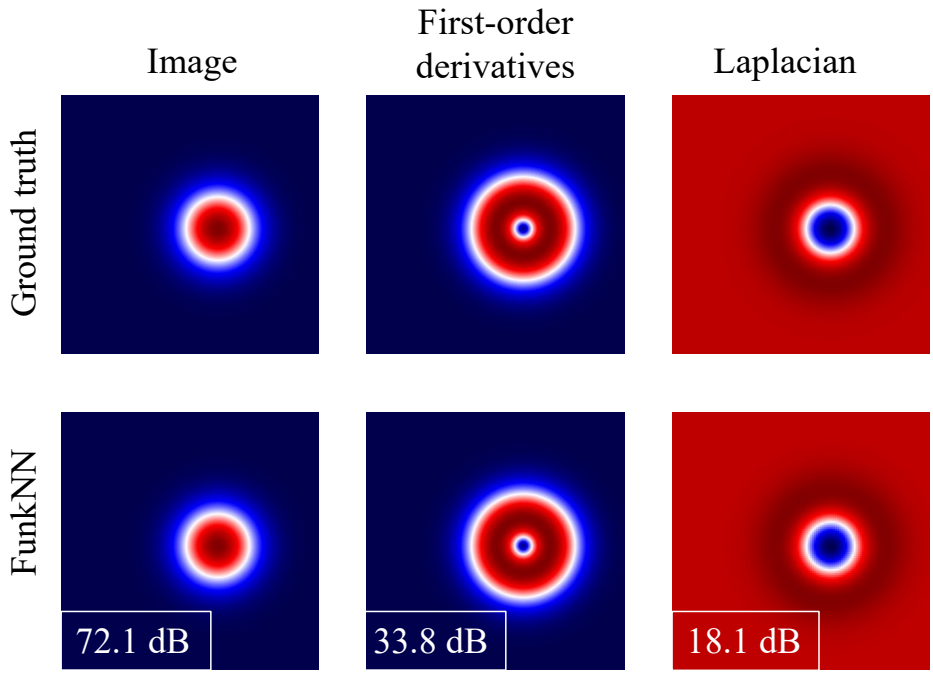


Figure 4.13: The derivatives produced by FunkNN on images of a Gaussian. The true derivatives are computed analytically.

Chapter 5

Trumpets: Injective Flows for Inference and Inverse Problems

In previous chapters, we focused on developing coordinate-based neural networks with strong generalization. Starting this chapter, we focus on Bayesian imaging; generating multiple solutions for the target image and uncertainty quantification to improve the reliability of our reconstruction and downstream interpretation. To this end, we develop injective neural networks specifically designed for ill-posed inverse problems.

Modeling a high-dimensional distribution from samples is a fundamental task in unsupervised learning. An ideal model would efficiently generate new samples and assign likelihoods to existing samples. Some deep generative models such as generative adversarial networks (GANs) [189] can produce samples of exceedingly high quality, but they do not give access to the underlying data distribution. Moreover, GANs are often hard to train, suffering from pathologies such as mode collapse [190, 191]. Since they are generally not invertible, or computing the inverse is slow, they are not well-suited for downstream inference tasks such as image reconstruction from compressive measurements or uncertainty quantification.

Normalizing flows alleviate many of the drawbacks of GANs: they approximate high-dimensional probability distributions as invertible transformations of a simple, tractable base distribution. They allow both efficient sampling and likelihood evaluation. They can be trained using maximum likelihood, and at inference time they provide direct access to likelihoods. These desirable features are a consequence of clever architectural components known as coupling layers [192].

Normalizing flows, however, are extremely compute-intensive. As a case in point, training a Glow model [70] for the 5-bit 256×256 CelebA dataset takes a week on 40 GPUs. This is in part because the dimension of the "latent" space in normalizing flows equals that of the generated images. Since signals of interest are often concentrated close to low-dimensional structures embedded in high-dimensional spaces, this is a waste of

resources. Beyond reducing computational cost, a low-dimensional latent space acts as a natural regularizer when solving ill-posed inverse problems [66].

In this chapter, we propose a new generative model termed TRUMPET—an injective flow based on convolutional layers that are *injective* by construction. Similarly to traditional coupling layers, our proposed layers have fast, simple inverses and tractable Jacobians; however, they map to a space of higher dimension. Since they are injective, they can be inverted on their range. Our design combines standard coupling layers with recent results on injective neural networks [193]. Further, our models can be trained via exact maximum likelihood by separating the training of the injective part from that of the bijective part [161].

TRUMPETs can be trained an order of magnitude faster than earlier injective models based on traditional normalizing flows [161] while producing samples of comparable (or better) quality. Moreover, thanks to their fast inverse, they can be used to design fast inference algorithms based on generative priors. We apply TRUMPETs to Bayesian inference problems in compressive sensing and limited-angle tomography. In particular, we devise an algorithm for efficient computation of a MAP estimator using a variant of projected gradient descent. The fast inverse yields a projection while thanks to injectivity we can compute the likelihoods. We then adapt recent work on uncertainty quantification for inverse problems with normalizing flows [194] to work with generative priors and a low-dimensional latent space of TRUMPETs. We anticipate that neural-network-based uncertainty quantification can be naturally integrated in a rigorous analysis in the context of inverse problems [195, 196].

Our **main contributions** can be summarized as follows:

- We propose *injective* coupling layers with fast inverses and tractable Jacobians.
- We use these layers to construct TRUMPETs—injective flow generative models. The proposed generative models train orders of magnitude faster than the usual flow models while producing samples of comparable or better quality and giving access to likelihoods.
- We apply the proposed models to Bayesian inference problems and uncertainty quantification, showing remarkable gains in efficiency and reconstruction quality over established methods. In particular, we show how the low-dimensional latent space of TRUMPETs leads to an efficient variational approximation of the posterior distribution.

In the following section we describe the construction of TRUMPETs; an overview of related work is given in Section 5.4.

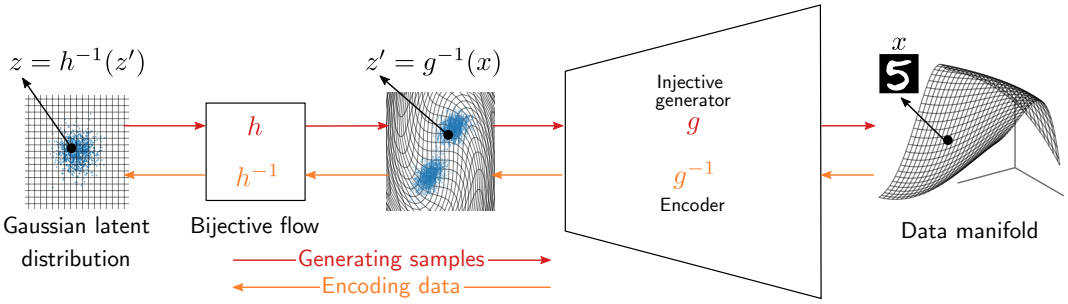


Figure 5.1: TRUMPET—A reversible injective flow-based generator

5.1 TRUMPETs: Injective flows

Flow-based generative models [187, 192] approximate the target distribution via a series of bijective transformations of a simple latent distribution. Unlike GANs [189] or VAEs [197] they allow for efficient *exact* likelihood evaluation. Crucial to the design of flow-based models are tractable inverses and Jacobians of all the constituent bijective transformations [70, 198], based on special coupling layers such as NICE [192] or RealNVP [187]. A generative model $f_\theta : \mathbb{R}^D \to \mathbb{R}^D$ parameterized by the weights θ maps latent variables Z to data X. Note that we use uppercase letters for random vectors and corresponding lowercase letters for their realizations. Log-likelihoods of the generated samples $x = f_\theta(z)$ can be evaluated as

$$\log p_X(x) = \log p_Z(f_\theta^{-1}(x)) - \log |\det J_{f_\theta}(f_\theta^{-1}(x))|. \tag{5.1}$$

Given an iid training dataset $\{\xi^{(i)}\}_{i=1}^n$ from some ground truth distribution[1] p_Ξ, training a normalizing flow entails maximizing the log-likelihood of the training data given as $\sum_{i=1}^N \log p_X(\xi^{(i)})$ over the weights θ in order to learn a generative model f_θ. Equivalently, it entails minimizing the KL divergence between p_X and p_Ξ. While invertibility ensures a non-singular J_{f_θ} at all points, defining likelihoods only requires injectivity of f_θ.[2]

5.1.1 Making flows injective

Machine learning for high-dimensional signals such as images relies on the fact that these signals concentrate around low-dimensional structures. We adopt the common assumption that p_Ξ is concentrated close to a d-dimensional manifold in $\mathbb{R}^D$, with $d \ll D$. We then aim to learn a generative model f_θ, now mapping from $\mathbb{R}^d$ to $\mathbb{R}^D$, such that the observed data lies in the range of f_θ. When f_θ is an injective map its Jacobian $J_{f_\theta} \in \mathbb{R}^{D\times d}$

[1] We use ξ to denote samples from the ground truth distribution p_Ξ to distinguish them from the samples x from p_X, the distribution induced by our network f_θ.

[2] With (L)ReLU activations, Jacobians are defined "only" almost everywhere; this rarely (if ever) causes issues in practice.

has full column rank for all input points. Thus one can still have access to likelihoods of samples generated by f_θ by modifying (5.1) as [199]

$$\log p_X(x) = \log p_Z(f_\theta^\dagger(x)) - \frac{1}{2}\log|\det[J_{f_\theta}(f_\theta^\dagger(x))^\mathsf{T} J_{f_\theta}(f_\theta^\dagger(x))]|, \tag{5.2}$$

which is valid for $x \in \mathrm{Range}(f_\theta)$. We use $f_\theta^\dagger$ to denote an inverse of f_θ on its range, that is $f_\theta^\dagger(f_\theta(z)) = z$. As described later, due to the way we construct $f_\theta^\dagger$, (5.2) corresponds to the likelihood of a projection of x on the range of f_θ for $x \notin \mathrm{Range}(f_\theta)$.

Building on the general change of variable formula (5.2), we propose TRUMPET—a network architecture that is injective by construction. The network architecture (Figure 5.1) consists of a "flat" invertible part which maps $\mathbb{R}^d$ to $\mathbb{R}^d$ and an expanding injective part which maps $\mathbb{R}^d$ to $\mathbb{R}^D$, resembling its namesake in shape. Crucially, expansion is enabled via injective revnet steps [200] generalizing the recently proposed Glow [70] layers.

We begin by reviewing the revnet step. A forward (F) revnet step has 3 operations, each having a simple inverse (I):

1. activation normalization,

$$\text{F: } y = \frac{x-\mu}{\sigma}, \qquad \text{I: } x = \sigma y + \mu,$$

2. 1×1 convolution with a kernel w,

$$\text{F: } y = \ell_w(x) = w * x, \qquad \text{I: } x = w^{-1} * y,$$

3. affine coupling layer

$$\begin{array}{lllll} \text{F:} & y_1 = x_1, & y_2 = s(x_1) \circ x_2 + b(x_2), \\ \text{I:} & x_1 = y_1, & x_2 = s(y_1)^{-1} \circ (y_2 - b(y_1)), \end{array} \tag{5.3}$$

where $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$, $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, and s and b are the scale and bias functions that are implemented by neural networks. The coupling layers have triangular Jacobians making their log determinants easy to compute.

We now generalize the second step to allow for an increase in dimension while retaining computational tractability.

Injective 1×1 convolutions. We consider generalizations of the 1×1 convolution layers (ℓ_w) that (1) are injective, (2) have fast (pseudo)inverse and (3) a fast Jacobian independent of x. These requirements yield two layer variants—linear and ReLU 1×1 convolutions:

$$
\begin{array}{lll}
 & \textsc{Linear} & \mathrm{ReLU} \\
\textsc{Forward} & y = w * x, & y = \mathrm{ReLU}\left(\begin{bmatrix} w \\ -w \end{bmatrix} * x\right); \\
\textsc{Inverse} & x := w^{\dagger} * y, & x := w^{\dagger} * \left(\begin{bmatrix} I & -I \end{bmatrix} y\right).
\end{array}
$$

Here $w^{\dagger}$ is the left pseudoinverse of w. Since w is a 1×1 convolution, we write it as a matrix of size $c_{\text{out}} \times c$, where c, c_{out} are the number of input and output channels respectively; taking the pseudoinverse of this matrix yields $w^{\dagger}$.

In Appendix 5.6.2, we show that for both types of layers,

$$
\log\det J_{\ell_w}^{\mathsf{T}} J_{\ell_w} = \sum_{i=1}^{c} s_i(w)^2,
$$

where the $s_i(w)$ are the singular values of w. We choose the size of w such that the number of output channels is kc (resp. $\lfloor \frac{k}{2} \rfloor c$) for the linear (resp. ReLU) layer. While $k \geq 1$ is enough for the linear variant to be injective, $k \geq 2$ is necessary and sufficient for the ReLU variant [193].

Injective revnet step. By generalizing the 1×1 convolutions to increase dimensions, we can still utilize the revnet step as in Glow by replacing the invertible 1×1 convolutions by their injective counterparts. Therefore, if the input tensor is of size $N \times N \times C$, the output after an injective revnet step is of size $N \times N \times kC$, where the expansion by a factor k occurs in the injective convolution (ℓ_w) step.

5.1.2 Architecture of TRUMPETs

Injective coupling layers introduced in the previous section allow us to build an architecture that trains at a fraction of the time and memory cost of regular flows. As shown in Figure 5.1, a TRUMPET model $f_\theta(z) = g_\gamma(h_\eta(z))$ with weights $\theta = (\gamma, \eta)$ has two components: an injective map $g_\gamma(z') = g_1 \circ g_2 \ldots \circ g_K(z')$ which maps from $\mathbb{R}^d$ to $\mathbb{R}^D$, and a bijective part h_η implemented as a flow $z' = h_\eta(z) = h_1 \circ h_2 \ldots \circ h_L(z)$ in the low-dimensional latent space. Unlike normalizing flows such an architecture allows us to progressively increase dimension and markedly reduce the number of parameters.

The role of the injective part g_γ is to match the shape of the manifold that supports the ground truth distribution p_Ξ, while the role of the low-dimensional flow is to match the density on the manifold. As recently proposed by [161] and as we elaborate in Section 5.1.3, this separation enables training even though likelihood is not defined for samples outside the range of f_θ.

To build the injective map g_γ we compose the proposed injective revnet layers, progressively increasing dimension from that of the latent space to that of the image space. To improve expressivity, at each resolution, we interleave a small number of bijective revnet layers. Each injective layer increases feature dimension by a factor of 2 in a single step in the forward direction (and decreases it by a factor of 2 in the reverse direction). Since our latent space is d-dimensional we need $m \approx \log_2 \frac{D}{d}$ injective layers interspersed with a few bijective layers. Following [187] we employ upsqueezing to increase resolution. Our network architecture results in significantly fewer parameters and faster training than the recently proposed variant of injective flows [161].

Finally, performance of revnets in generative modeling of images can be improved [187] by introducing multiscale implementations of the scale (s) and bias (b) functions. For these implementations, we propose to use U-Nets [25] in affine coupling layers as opposed to regular convolutional stacks used in previous normalizing flows [70, 187]. We find that integrating U-Nets greatly improves the performance of our network.

5.1.3 Training of TRUMPETs

An advantage of injective architectures such as TRUMPETs is that they can be trained using maximum likelihood. However, since the range of f_θ is a d-dimensional submanifold in $\mathbb{R}^D$, likelihoods of the samples not on this manifold are not defined. To circumvent this difficulty we adopt a strategy recently proposed by [161]. We split the training into two phases: (i) the mean squared error (MSE) phase where we only optimize the injective part g_γ, and (ii) the maximum likelihood (ML) training phase where we fit the parameters η of the bijective part h_η to maximize the likelihood of the preimage of training data through g_γ; this step aims to match the density of p_X to that of the ground truth p_Ξ.

The loss function that we minimize to find the parameters of g_γ is given as

$$\mathcal{L}_{\text{MSE}}(\gamma) = \frac{1}{N} \sum_{i=1}^{N} \|\xi^{(i)} - g_\gamma(g_\gamma^\dagger(\xi^{(i)}))\|_2^2, \tag{5.4}$$

where $\xi^{(i)}$-s are the training samples. We find that only a few epochs of training are sufficient to train g_γ. Note that $P_{g_\gamma}(x) := g_\gamma(g_\gamma^\dagger(x))$ is an idempotent projection operator on the range of g_γ. The low-dimensional range of g_γ acts as a regularizer in the context of inverse problems. Injectivity implies that the range of f_θ is a true manifold unlike in the case of GANs where it may be an arbitrary low-dimensional structure [193]. This allows us to define likelihoods as in (5.2).

After the MSE training phase, we have a manifold that near-interpolates the data samples. In the ML training phase, we match the density (or measure) on the manifold to p_Ξ by maximizing the likelihood of the preimages of the training samples $\{g_\gamma^\dagger(\xi^{(i)})\}$ over

η. This gives us the loss function for the ML training phase as

$$\mathcal{L}_{\text{ML}}(\eta) = \frac{1}{N}\sum_{i=1}^{N}\left(-\log p_Z(z^{(i)}) + \sum_{l=1}^{L}\log|\det J_{h_{\eta,l}}|\right), \tag{5.5}$$

where $z^{(i)} = h_\eta^{-1}(g_\gamma^\dagger(\xi^{(i)}))$ and $J_{h_{\eta,l}}$ are evaluated at appropriate intermediate inputs. Together with the gradually-expanding architecture of TRUMPETs this two-step procedure yields much faster training than previous work which concatenates standard invertible flows.

Stability of layerwise inversions. To minimize $\mathcal{L}_{\text{MSE}}$ (5.4), we need to calculate the left inverse $g_\gamma^\dagger$ for points that do not lie in the range of g_γ. This entails computing the pseudoinverses of injective convolutional layers ℓ_w. We study the stability of inversion for out-of-range points under the assumption that $y' = \ell_w(x) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I)$. In particular, we are interested in estimating the inverse error $E_{\text{Inv}}(y') = \|\ell_w^\dagger(y') - x\|_2^2$ and the re-projection error $E_{\text{Proj}}(y') = \|\ell_w(\ell_w^\dagger(y')) - y'\|_2^2$.

We show in Appendix 5.6.2 that for both linear and ReLU injective convolutions the average errors are

$$\mathbb{E}_\epsilon E_{\text{Inv}}(y) \propto \sigma_\epsilon^2 \sum_{i=1}^{c}\frac{1}{s_i(w)^2}, \qquad \mathbb{E}_\epsilon E_{\text{Proj}}(y) \propto \sigma_\epsilon^2,$$

where $s_i(w)$-s are the singular values of w and c is the number of input channels in the forward direction.

The reconstruction error thus behaves gracefully in σ_ϵ, but could blow up for poorly conditioned w. In order to stabilize inversions and training, we regularize the inverse via Tikhonov regularization. This changes the error terms from $\sum_{i=1}^{c} 1/s_i(w)^2$ to $\sum_{i=1}^{c}\frac{s_i(w)}{s_i(w)^2+\lambda}$ which is upper bounded by $\frac{c}{2\sqrt{\lambda}}$, thus effectively stabilizing training. Here, λ is the regularization parameter.

5.2 Inference and uncertainty quantification with TRUMPET

From this chapter, we change our notation in Chapter 1 to the standard notation for probabilistic modeling. We consider reconstructing an object $x \in \mathbb{R}^D$ from measurements $y \in \mathbb{R}^n$. We assume that x and y are realizations of jointly distributed random vectors X, Y, with the joint distribution $p_{X,Y}(x, y)$. In inference, we are mainly interested in characterizing the posterior $p_{X|Y}(x|y)$. We note that this setting generalizes point estimation of x given y common in inverse problems where the task is to recover x from

measurements $y = Ax + \epsilon$, where ϵ is additive noise and $A \in \mathbb{R}^{n \times D}$ is the forward operator. The maximum a posteriori (MAP) estimate $x_{\text{MAP}} = \operatorname{argmax}_x p_{X|Y}(x|y)$ can be obtained through Bayes theorem,

$$\begin{aligned} x_{\text{MAP}} &= \operatorname*{argmin}_x -\log p_{Y|X}(y|x) - \log p_X(x) \\ &= \operatorname*{argmin}_x \tfrac{1}{2}\|y - Ax\|_2^2 - \sigma_\epsilon^2 \log p_X(x), \end{aligned} \tag{5.6}$$

where we assume that $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I)$.

5.2.1 MAP estimation with TRUMPET prior

We now address two inference tasks where TRUMPETs are particularly effective. Recall that since g_γ is injective one can build a fast projector $P_{g_\gamma}(x) = g_\gamma(g_\gamma^\dagger(x))$ on the range of g_γ, i.e., the range of our generator.

Beyond simply projecting on the range, injectivity and Bayes theorem enable us to maximize the likelihood of the reconstruction under the posterior induced by the TRUMPET prior [201]. The injective flow (iFlow) algorithm described below in Algorithm 2 then alternates projections on the range with gradient steps on the data fidelity term and the prior density. We study two variants—iFlow and iFlow-L that correspond to running Algorithm 2 without and with the $-\log p_X$ term.

Algorithm 2 iFlow

Input: loss function L, y, A, g_γ
Parameter: step size η and $\lambda(\propto \sigma^2)$

$x^{[0]} = A^\dagger y$ **for** $i \leftarrow 0$ **to** $T - 1$ **do**
  $v \leftarrow P_g(x^{[i]})$ $x^{[i+1]} \leftarrow \texttt{GradientStep}(L(v))$
end
$x^{[T]} \leftarrow P_g(x^{[T]})$

One caveat with computing $-\log p_X(x)$ is that it requires $\log|\det[J_{f_\theta}^\mathsf{T} J_{f_\theta}](f_\theta^\dagger(x))|$ according to (5.2). While we have layer-wise tractable Jacobians, $\log|\det J_{f_\theta}^\mathsf{T} J_{f_\theta}|$ cannot be split into layerwise $\log\det$ terms due to the change of dimension. Fortunately, the literature is abundant with efficient stochastic estimators. We describe one in Section 5.2.3 that we use to compare and report likelihoods. In order to implement the iFlow-L, however, we propose a much faster scheme.

We define the following quantity,

$$R(x) = -\log p_Z(f^\dagger(x)) + \frac{1}{2}\sum_{k=1}^{K} \log|\det J_{g_{\gamma,k}}^\mathsf{T} J_{g_{\gamma,k}}| + \sum_{l=1}^{L} \log|\det J_{h_{\eta,l}}|, \tag{5.7}$$

where the layer Jacobians are evaluated at the appropriate intermediate layer outputs. Since all our layers including the injective layers have $\log\det$ Jacobians readily available we use (5.7) as a surrogate for intractable term $-\log p_X(x)$. This yields the proposed iFlow-L algorithm (Algorithm 2) for solving (5.6). The objective function is

$$L(x) := \tfrac{1}{2}\|y - Ax\|_2^2 + \sigma^2 R(x). \tag{5.8}$$

Note that when solving inverse problems we constrain the final solution x to be in the range of f, that is, $x = f_\theta(z)$ for some $z \in \mathbb{R}^d$.

5.2.2 Posterior modeling and uncertainty quantification

The second application enabled by TRUMPETs is efficient uncertainty quantification for inverse problems in imaging. We build on a method recently proposed by [194] which computes a variational approximation to the posterior $p_{X|Y}(x|y)$ corresponding to the measurement y and a "classical" regularizer. They train a normalizing flow which produces samples from the posterior, with the prior and the noise model given implicitly by the regularized misfit functional.

The injectivity of the TRUMPET generator f_θ and the assumption that the modeled data concentrates close to the range of f_θ allows us to write the posterior on X, $p_{X|Y}$, in terms of $p_{Z|Y}$, with $X = f_\theta(Z)$. That is,

$$p_{X|Y}(f_\theta(z)|y) = p_{Z|Y}(z|y) \cdot |\det J_{f_\theta}^T J_{f_\theta}|^{-1/2}. \tag{5.9}$$

We can thus derive a computationally efficient version of the algorithm proposed by [194] by only training a low-dimensional flow.

Instead of using TRUMPETs to simply reduce computational complexity, we showcase another interesting possibility: approximating the posterior with respect to the learned prior given by the TRUMPET. To do this we train another network u_υ which is a low-dimensional flow, so that the distribution of $f_\theta \circ u_\upsilon(T)$ approximates the posterior $p_{X|Y}$ when T is an iid Gaussian vector. The generative process for (approximate) samples from $p_{X|Y}$ is then

$$T \xrightarrow{u_\upsilon} Z \underbrace{\xrightarrow{h_\eta} Z' \xrightarrow{g_\gamma}}_{f_\theta} X.$$

We thus require that $u_\upsilon(T) \sim p_{Z|Y}$ with $T \sim \mathcal{N}(0, I)$ and $X = f_\theta(Z)$. Letting q_υ be the distribution of $u_\upsilon(T)$, the parameters υ are adjusted by minimizing the KL divergence between q_υ and $p_{Z|Y}$,

$$\begin{aligned}\upsilon^\star &= \operatorname*{argmin}_\upsilon \; \mathrm{D}_{\mathrm{KL}}\left(q_\upsilon \,\|\, p_{Z|Y}\right) = \operatorname*{argmin}_\upsilon \; \mathbb{E}_{Z\sim q_\upsilon}[-\log p_{Y|Z}(y|Z) - \log p_Z(Z) + \log q_\upsilon(Z)] \\ &= \operatorname*{argmin}_\upsilon \mathbb{E}_{T\sim\mathcal{N}(0,I)}[-\log p_{Y|Z}(y|u_\upsilon(T)) - \log p_Z(u_\upsilon(T)) + \log p_T(T) \\ &\quad - \log|\det J_{u_\upsilon}(T)|].\end{aligned} \tag{5.10}$$

We revisit the inverse problem associated with $y = Ax + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. In this setting we have

$$\begin{aligned} \upsilon^\star = \operatorname*{argmin}_{\upsilon} \mathbb{E}_{T\sim\mathcal{N}(0,I)} \left[\tfrac{1}{2}\|y - Af_\theta(u_\upsilon(T))\|_2^2 \right. \\ \left. - \sigma^2 \log p_Z(u_\upsilon(T)) - \sigma^2 \log|\det J_{u_\upsilon}(T)|\right]. \end{aligned} \tag{5.11}$$

We evaluate (5.11) by drawing k iid samples $\{t_i\}_{i=1}^k$ from the base Gaussian, yielding the following loss to train u_υ,

$$\begin{aligned} \mathcal{L}(\upsilon) := \frac{1}{k}\sum_{i=1}^{k}(\|y - Af_\theta(u_\upsilon(t_k))\|_2^2 \\ - \sigma^2 \log p_Z(u_\upsilon(t_k)) - \beta\sigma^2 \log|\det J_{u_\upsilon}(t_k)|), \end{aligned} \tag{5.12}$$

where we added β as a hyper-parameter to control the diversity of samples we generate from the posterior [194].

5.2.3 Estimating log-likelihoods

The training of TRUMPETs only requires the log det of the Jacobian of h_η. Some applications call for the log det of the Jacobian of the full network, typically evaluated a small number of times. Here, we provide a stochastic estimate via the truncation of a Neumann series.

As $J_{f_\theta}^\mathsf{T} J_{f_\theta}$ is a square matrix, we find that

$$\begin{aligned} \log|\det J_{f_\theta}^\mathsf{T} J_{f_\theta}| &= \operatorname{Tr}(\log J_{f_\theta}^\mathsf{T} J_{f_\theta}) \\ &= \operatorname{Tr}\left(\log\frac{1}{\alpha}(I - (I - \alpha J_{f_\theta}^\mathsf{T} J_{f_\theta}))\right) \\ &= -\operatorname{Tr}\left(\sum_{k=1}^{\infty}\frac{(I - \alpha J_{f_\theta}^\mathsf{T} J_{f_\theta})^k}{k}\right) - d\log\alpha \\ &\approx -\mathbb{E}_v \sum_{k=1}^{n}\frac{1}{k} v^\mathsf{T}(I - \alpha J_f^\mathsf{T} J_f)^k v - d\log\alpha \end{aligned}$$

where we choose α such that the maximal singular value of $I - \alpha J_{f_\theta}^\mathsf{T} J_{f_\theta}$ is about 0.1. This ensures that the series converges fast and we can truncate the expansion to about 10 terms. We estimate the largest singular value of $J_{f_\theta}^\mathsf{T} J_{f_\theta}$ using power iteration. In the last step we use the Hutchinson trace estimator [202] to evaluate the trace; the v-s are sampled iid from $\mathcal{N}(0, I)$. The terms of the power series can be efficiently implemented by vector-Jacobian and Jacobian-vector products using automatic differentiation as described in Algorithm 3 [203].

Algorithm 3 Stochastic $\log\det$ Jacobian estimator

Input: f, n
Output: $\log|\det J_f^\mathsf{T} J_f|$

$\log\det = 0$
$\beta = 0.9\,(\texttt{MaxSingularValue}(J_f))^{-1}$
Draw v from $\mathcal{N}(0, I)$
$w^\mathsf{T} = v^\mathsf{T}$
for *k=1* **to** n **do**
  $u_1^\mathsf{T} = \texttt{jvp}(w)$
  $u_2^\mathsf{T} = \texttt{vjp}(u_1)$
  $w = w - \beta u_2$
  $\log\det \mathrel{-}= \dfrac{w^\mathsf{T} v}{k}$
end
$\log\det \mathrel{-}= d\log\beta$

5.3 Experiments

We begin by evaluating the generative performance of TRUMPETs. Next, we test TRUMPETs on two inference tasks in imaging: maximum a posteriori estimation and uncertainty quantification.

5.3.1 Generative modeling

We train TRUMPETs on the MNIST [204], CIFAR10 [205], CelebA [206] and Chest X-ray [207] datasets with image sizes $32 \times 32 \times 1$, $32 \times 32 \times 3$, $64 \times 64 \times 3$ and $128 \times 128 \times 1$ respectively.

We find that our networks train much faster than invertible flows and their recent injective generalizations. As a point of comparison, training the models of [161] takes over 10 days on the CelebA dataset. The corresponding TRUMPET trains in 38 hours while yielding better samples in terms of the Fréchet inception distance (FID) [208] (see Table 5.1).[3]

Since the range of a TRUMPET is a manifold, a relevant metric is the reconstruction error, $\frac{\|\xi - f_\theta(f_\theta^\dagger(\xi))\|}{\|\xi\|}$, which we report for ξ-s on the test set in Table 5.2. We show generated samples and reconstructions on test sets from trained TRUMPETs in Figures 5.6b, 5.7b, 5.8 and 5.9 in Appendix 5.6.3.

[3]Our FID scores are reported at sampling temperature $T = 1$, that is, we use the same prior distribution statistics for training and sampling. We show the variation of the FID metric with the temperature in Figure 5.5 in Appendix 5.6.3

Table 5.1: FID scores on 8-bit 64×64 celebA dataset.

Model	FID
[209]	40.23
[161]	37.4
TRUMPET (Ours)	**34.3**

Table 5.2: Training times in hours for TRUMPET: all models were trained on a single V100 GPU

	Training time (hours)	$\frac{\|x - f_\theta(f_\theta^\dagger(x)\|}{\|x\|}$	Trainable params
MNIST	11	0.04	9M
CIFAR10	11	0.22	9M
CelebA	38	0.15	16M
Chest X-ray	25	0.13	11M

We note that the variants with the linear and ReLU 1×1 convolutions perform similarly (see Figures 5.6a,5.6b, 5.7a, 5.7b); hence, for the subsequent datasets and experiments we only report results with the linear variant.

The negative log-likelihood values estimated for trained TRUMPET models using Algorithm 3 on the $[-1, 1]$ normalized MNIST and CelebA dataset are 114.82 ± 5.8 and 294 ± 7.4 nats respectively. Note that these represent likelihoods over measures supported in a d-dimensional latent space whereas the previous literature [70, 187] reports D-dimensional likelihoods. This issue is unfortunately not resolved by simply dividing by dimension. We thus caution the reader that such a comparison may be misleading.

5.3.2 MAP estimation

We test TRUMPETs on image reconstruction from compressive measurements. We work with four different forward operators / corruption models: (i) **RandGauss (m)**: we sample an entrywise iid Gaussian matrix $A \in \mathbb{R}^{n \times D}$, where $n = 250$ and D is the dimension of the vectorized image; (ii) **RandMask (p)**: we mask pixels (that is, replace a pixel with zero) with probability $p = 0.15$; (iii) **Super-resolution (x4)**: we downsample the image by a factor of 4 along each dimension; and (iv) **Mask (s)**: we mask (replace with zero) an $s \times s$-size portion of the image.

Since TRUMPETs have a readily available inverse we focus on the benefits this brings in imaging. Specifically, we use Algorithm 2 to compute an estimate using a trained TRUMPET prior. We test the algorithm on the MNIST and CelebA datasets and use the same TRUMPET prior for all problems. We compare our approach to two deep learning baselines—compressed sensing with generative models (CSGM) [66] and deep

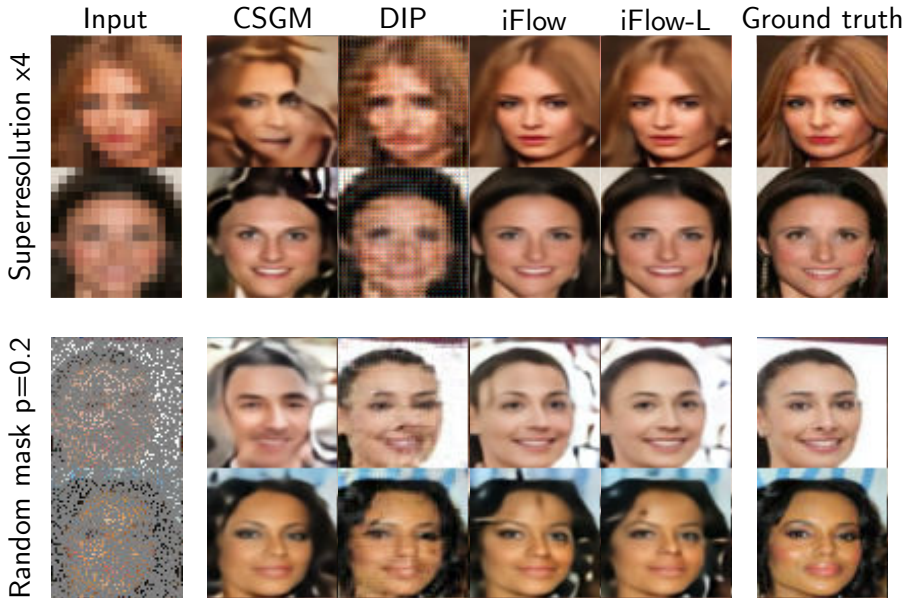


Figure 5.2: Comparison of various reconstruction schemes. The iFlow-L and iFlow methods refer to Algorithm 2 respectively with and without the likelihood term.

image prior (DIP) [210]. CSGM solves $\hat{x} = f(\text{argmin}_z \|y - Af(z)\|_2^2)$ while DIP solves $\hat{x} = f_\theta(\text{argmin}_\theta \|y - Af_\theta(z)\|_2^2)$ given a randomly chosen fixed z and regularized by early stopping. Figure 5.2 compares all methods for the superresolution and random masking problems on the CelebA dataset while Table 5.3 gives a comprehensive evaluation for all inverse problems.

We also perform an ablation study to assess the influence of including the prior likelihood as opposed to simply doing a gradient descent with manifold projections [211]. The latter corresponds to setting $\lambda = 0$ in Algorithm 2. Table 5.3 clearly shows that accounting for the prior density and not only support—that is, computing the MAP estimate—performs better in almost all settings.

We mention that we attempted to compare with a method involving projections proposed by [149] but found it to be $50 - 100\times$ slower than iFlow. It was thus infeasible to finalize this comparison. On average we found that DIP converged the fastest followed by our method which was about $2\times$ slower. Finally, while each iteration of CSGM was as fast as each of DIP, CSGM requires several restarts which made the method about $4\times$ slower than ours. We report the best results from CSGM with 10 restarts.

Note that the baselines [66,149,210] were developed without injectivity as a constraint. As a result, they typically use off-the-shelf GAN architectures inspired by [181], but they are by design agnostic to architectural details. To keep the comparisons fair we use the same generative model f_θ for all methods. This allows us to test the importance of tractable inverses and likelihoods for the design of image reconstruction algorithms based on generative priors.

Table 5.3: Performance on inverse problems measured in reconstruction SNR (dB)

	Dataset	CSGM	DIP	iFlow	iFlow-L
RandGauss ($m = 250$)	MNIST	11.32	12.72	21.34	**21.81**
	CelebA	8.98	**11.25**	8.90	8.91
RandMask ($p = 0.15$)	MNIST	3.85	4.94	4.76	**10.10**
	CelebA	12.63	**17.26**	13.89	14.43
Super-resolution ($\times 4$)	MNIST	5.943	1.0	9.851	**12.75**
	CelebA	11.08	14.12	17.36	**20.07**
Mask ($s = 15$ px)	MNIST	3.34	4.38	3.90	**9.54**
	CelebA	13.42	21.31	21.74	**21.79**
Limited-view CT	Chest	11.58	13.76	20.93	**21.23**

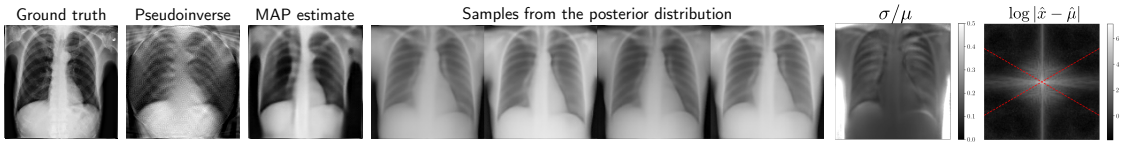


Figure 5.3: Uncertainty quantification for limited view CT.

5.3.3 Posterior modeling and uncertainty quantification

Next, we use TRUMPET priors for uncertainty quantification in computed tomography. We work with a chest X-ray dataset and use the limited-angle CT operator as the forward operator, A. We choose a sparse set of $n_{\text{angles}} = 30$ view angles from $30°$ to $150°$, with a $60°$ missing cone.[4] We add 30dB noise to the measurements. The resulting inverse problem is severely ill-posed and solving it requires regularization. (Note that Table 5.3 includes the performance of Algorithm 2 on this problem.)

Here we provide a pixel-wise uncertainty estimate of the form $\mathbb{E}_{X\sim p_{X|Y=y}}|X - \langle X\rangle|^p$, with $p = 1, 2$, $|\cdot|$ the pixel-wise absolute value, and $\langle X\rangle$ the posterior mean. In Figure 5.3, we show the MAP estimate obtained from the iFlow-L algorithm (Algorithm 2). We also show the Fourier spectrum of the mean absolute deviation calculated in the Fourier domain where the mean was calculated over the Fourier transform of all samples from the posterior. We observe a cone of increased uncertainty in the Fourier spectrum that corresponds to the missing angles in the limited-view CT operator. Furthermore, we observe a thick vertical bright line that corresponds to uncertainty in predicting the

[4]We emphasize that the purpose of this numerical experiment is to illustrate the UQ algorithm rather than provide a realistic, competitive method. Indeed, in real CT the projections would be taken in planes perpendicular to the spine.

location of the ribs (which have a strong horizontal periodic component) as shown in the middle plot of Figure 5.3.

Reassuringly, both the spatial- and the frequency-domain representations of uncertainty correlate well with our intuitive expectations for this problem. Positions of the ribs in space and the missing cone in the spectrum exhibit higher uncertainty.

5.4 Related works

Normalizing flows have been introduced in [192]. The key to their success are invertible coupling layers with triangular Jacobians. Different variants of the coupling layer along with multiscale architectures [70, 187, 198] have considerably improved performance of normalizing flows. Glow [70] uses invertible 1×1 convolutions to improve expressivity, producing better samples than NICE and Real-NVP. Alas, training a Glow model is extremely compute intensive—1 week on 40 GPUs for the 5-bit 256×256 CelebA dataset. A crucial drawback of the mentioned models is that they are bijective so the dimension of the latent and data spaces coincide. This results in a large number of parameters and slow training: since the ground data lies close to low-dimensional subset of $\mathbb{R}^D$, training should encourage the model to become "almost non-invertible" which makes the optimization more difficult.

The authors of [209] propose approximate injective flows by using spectral regularization in auto-encoders. However they lack access to likelihoods. Further, their training strategy is only a proxy for injectivity. The authors of [161] proposed injective flows to learn a data distribution on a manifold very similar to our work, including a two-stage training scheme we use. However, they use regular normalizing flow architectures with zero padding in the latent space which results in architectures that are very expensive to train. [212] build injective flows by adding noise to the range; this requires stochastic inversion whereas ours is deterministic. More recently, the authors of [213] proposed a canonical manifold learning flow to improve the efficiency and compactness of the latent representation.

In a parallel development, autoregressive flows were shown to have favorable expressivity compared to normalizing flows. We refer to [214–216] and the references therein for a more extensive account.

5.5 Summary

We proposed TRUMPETs—a flow-based generative model that is injective by construction. TRUMPETs alleviate the main drawback of invertible normalizing flows which is that they are very expensive to train. We showed that TRUMPETs are competitive in terms

of generative modeling performance and that the fast inverse on the range markedly improves reconstructions in ill-posed inverse problems. We also showed how to use TRUMPETs to model posteriors and perform uncertainty quantification directly in the low-dimensional latent space. Currently our reconstructions on data lack high frequency features; this is common in normalizing flow models [187]. Strategies such as adding the adversarial loss in the MSE phase of training may help alleviate this drawback. Furthermore, using a richer class of coupling layers may help— [217] show that flows based on rational quadratic splines are more expressive. Integrating such layers also holds promise for improving the expressivity of TRUMPETs.

Our work combines several basic ideas in an intuitive way that yields gains in computational efficiency and modeling quality. It is worth noting that recent results on universality of globally injective neural networks [193] and universality of flows [218] suggest that TRUMPETs are universal approximators of probability measures concentrated on Lipschitz manifolds; a rigorous proof is left to future work.

5.6 Appendix

5.6.1 Network architecture and training details

We describe the injective portion of our network architecture that was used to train a CelebA dataset in Figure 5.4. The bijective revnet block has 3 bijective revnet steps in each block while the injective revnet block has just one injective revnet step which is explained in details in Section 5.1.1. The bijective part of our network is not shown in Figure 5.4 but it has 32 bijective revenet steps.

For the scale and bias terms of the coupling layer we used the U-Net architecture with 2 downsampling blocks and 2 corresponding upsampling blocks. Each resolution change is preceded by 2 convolution layers with 32 and 64 output channels. We choose the latent space dimension as 64 for MNIST, 256 for Chest X-ray dataset and 192 for all other datasets. We normalize the data to lie in $[-1, 1]$.

The number of training samples for CelebA, Chest X-ray, MNIST and CIFAR10 are 80000, 80000, 60000, and 50000 respectively. We trained all models for about 300 epochs with a batch size of 64.

All models are trained with Adam optimizer [97] with learning rate 10^{-4}. $\gamma = 10^{-6}$ was used as the Tikhonov regularizer parameter for computing pseudoinverse of injective convolutional layers.

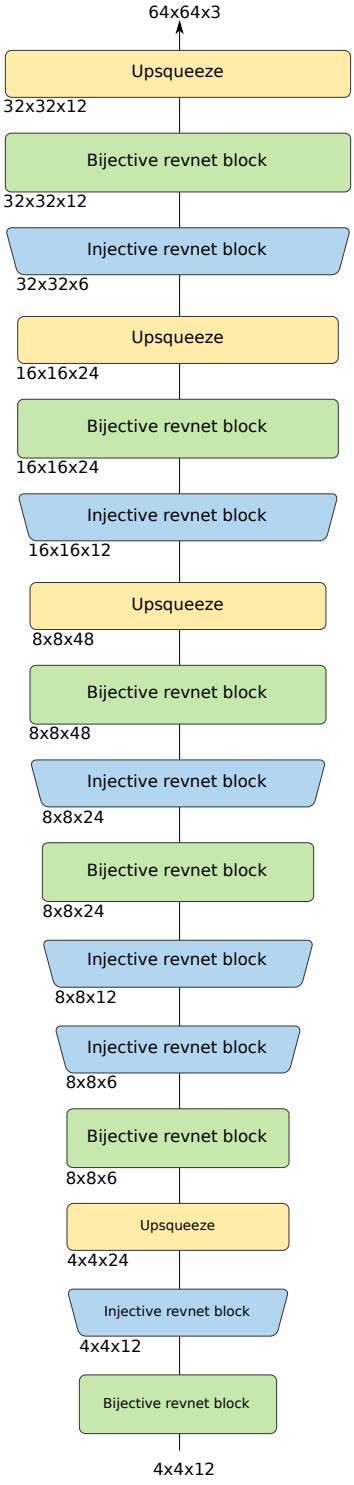


Figure 5.4: CelebA architecture for the injective portion g of TRUMPET. The input size to each layer is written below it.

5.6.2 Derivations of error

Measuring error due to deviations from range

Claim 1. Consider $y' = y + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I)$, $y = \ell_w(x)$ and let $E_{\text{Inv}}(y') := \|\ell_w^\dagger(y') - x\|_2^2$ and the re-projection error $E_{\text{Proj}}(y') := \|\ell_w(\ell_w^\dagger(y')) - y'\|_2^2$. Then for both ReLU and linear variants of ℓ_w we have

$$\mathbb{E}_\epsilon E_{\text{Inv}}(y') \propto \sigma_\epsilon^2 \sum_{i=1}^{c} \frac{1}{s_i(w)^2}, \quad \mathbb{E}_\epsilon E_{\text{Proj}}(y') \propto \sigma_\epsilon^2, \tag{5.13}$$

where $s_i(w)$'s are the singular values of w and c is the number of input channels in the forward direction.

Proof. Consider $y' = y + \epsilon$, where $y = \ell_w(x)$ and $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I_{2n})$. We consider a vectorized x and write the 1×1 convolution as a matrix-vector product, Wx say. For a ReLU injective convolution one could write the inverse as

$$x' = W^\dagger \begin{bmatrix} I_n & -I_n \end{bmatrix} y'. \tag{5.14}$$

We calculate $\mathbb{E}_\epsilon \|x' - x\|_2^2$. Let $M := \begin{bmatrix} I_n & -I_n \end{bmatrix}$ and $B := W^\dagger$, then

$$\begin{aligned} x' &= BM(y + \epsilon) \\ x' - x &= BM\epsilon, \end{aligned}$$

whence

$$\begin{aligned} \|x' - x\|_2^2 &= (BM\epsilon)^\mathsf{T} BM\epsilon \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(BM\epsilon(BM\epsilon)^\mathsf{T}\right) \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(BM\epsilon\epsilon^\mathsf{T} M^\mathsf{T} B^\mathsf{T}\right) \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(M^\mathsf{T} B^\mathsf{T} BM\epsilon\epsilon^\mathsf{T}\right) \end{aligned}$$

so that

$$\begin{aligned} \mathbb{E}_\epsilon \|x' - x\|_2^2 &= \mathbb{E}_\epsilon \operatorname{Tr}\left(M^\mathsf{T} B^\mathsf{T} BM\epsilon\epsilon^\mathsf{T}\right) \\ \mathbb{E}_\epsilon \|x' - x\|_2^2 &= \operatorname{Tr}\left(M^\mathsf{T} B^\mathsf{T} BM\right) \sigma_\epsilon^2 \\ \mathbb{E}_\epsilon \|x' - x\|_2^2 &= 2 \operatorname{Tr}\left(B^\mathsf{T} B\right) \sigma_\epsilon^2 \\ \mathbb{E}_\epsilon \|x' - x\|_2^2 &= 2 \sum_{i=1}^{c} s_i(w)^{-2} \sigma_\epsilon^2. \end{aligned}$$

□

Similarly for a linear layer the inverse is given as $x' = By'$. Therefore,

$$\begin{aligned} x' &= B(y + \epsilon) \\ x' - x &= B\epsilon \end{aligned}$$

hence

$$\begin{aligned} \|x' - x\|_2^2 &= (B\epsilon)^\mathsf{T} B\epsilon \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(B\epsilon(B\epsilon)^\mathsf{T}\right) \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(B\epsilon\epsilon^\mathsf{T} B^\mathsf{T}\right) \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(B^\mathsf{T} B\epsilon\epsilon^\mathsf{T}\right) \end{aligned}$$

so that

$$\mathbb{E}_\epsilon \|x' - x\|_2^2 = \sum_{i=1}^{c} s_i(w)^{-2} \sigma_\epsilon^2. \qquad \square$$

The re-projection error for a ReLU layer is given as

$$\begin{aligned} \sqrt{E_{\text{Proj}}(y')} &= \left\| \mathrm{ReLU}\left(\begin{bmatrix} W \\ -W \end{bmatrix} x' \right) - y \right\| \\ &= \left\| \mathrm{ReLU}\left(\begin{bmatrix} W \\ -W \end{bmatrix} x' \right) - \mathrm{ReLU}\left(\begin{bmatrix} W \\ -W \end{bmatrix} x \right) - \epsilon \right\| \\ &\leq \left\| \begin{bmatrix} W \\ -W \end{bmatrix} x' - \begin{bmatrix} W \\ -W \end{bmatrix} x \right\| + \|\epsilon\| \\ &= \left\| \begin{bmatrix} W \\ -W \end{bmatrix} (x + BM\epsilon) - \begin{bmatrix} W \\ -W \end{bmatrix} x \right\| + \|\epsilon\| \\ &= \left\| \begin{bmatrix} W \\ -W \end{bmatrix} BM\epsilon \right\| + \|\epsilon\| \\ &\leq (2\|WW^\dagger\|_F + 1)\|\epsilon\| \end{aligned}$$

Squaring both sides, we get

$$E_{\text{Proj}}(y') = (2\sqrt{c} + 1)^2 \|\epsilon\|^2. \qquad \square$$

Similarly, for a linear layer we have

$$\begin{aligned} E_{\text{Proj}}(y') &= \|Wx' - Wx - \epsilon\|^2 \\ &= \left\|WW^\dagger\epsilon - \epsilon\right\|^2 \\ &= (c+1)\left\|\epsilon\right\|^2. \end{aligned}$$

$\square$

log-determinants of Jacobians for ReLU injective convolutions

We vectorize x and, again, write the 1×1 convolution as a matrix-vector product Wx. Then, for a ReLU 1×1 convolution, we have

$$y = \mathrm{ReLU}\left(\begin{bmatrix} W \\ -W \end{bmatrix}\right) x.$$

This could be trivially rewritten as $y = W'x$, where the rows of W' are $w'_i = w_i$ if $\langle w_i, x \rangle > 0$ and $w'_i = -w_i$ otherwise. We note that changing the row signs does not change $|\det W|$. Hence, for such a ReLU injective convolutional layer, $\ell_w \log |\det J_{\ell_w}^T J_{\ell_w}| = \sum_{i=1}^{c} s_i^2(w)$, where $s_i(w)$'s are the singular values of w, where w is the 1×1 kernel corresponding to the convolution matrix W.

5.6.3 Generated samples

In Figures 5.6a, 5.6b and Figures 5.7a, 5.7b we compare the performance of TRUMPETs trained with ReLU and linear injective convolutions on the MNIST and 64×64 CelebA datsets. Both variants offer similar performance hence we choose to use linear convolutions for the rest of our results regarding inverse problems and uncertainty quantification. In Figures 5.9 and 5.8 we show generated samples from TRUMPET and a few reconstructions of original samples, x given as $f(f^{\dagger}(x))$ on the CIFAR10 and Chest X-ray datasets respectively. For the CIFAR10 dataset, we do see a low frequency bias in the generated samples. For other datasets the low-frequency bias seems to be less of a problem. In fact, on these datasets TRUMPETs outperform previous injective variants of flows [161, 209].

The temperature of sampling has a significant effect on the FID scores as shown in Figure 5.5. While samples in Figures 5.7a, 5.7b are for $T = 1$ we share some samples in Figure 5.10 for $T = 0.85$.

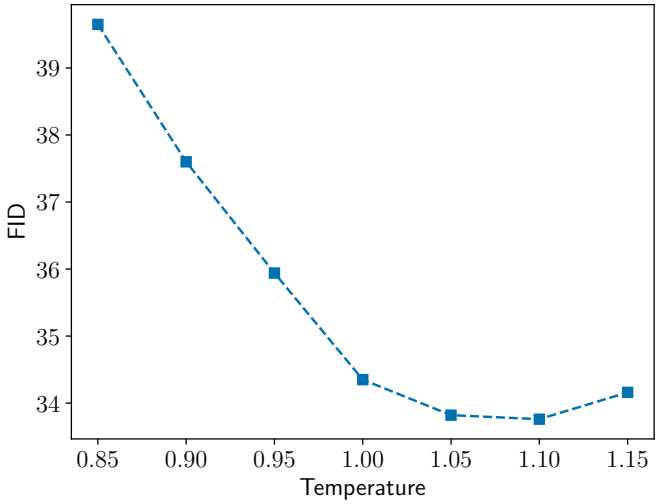
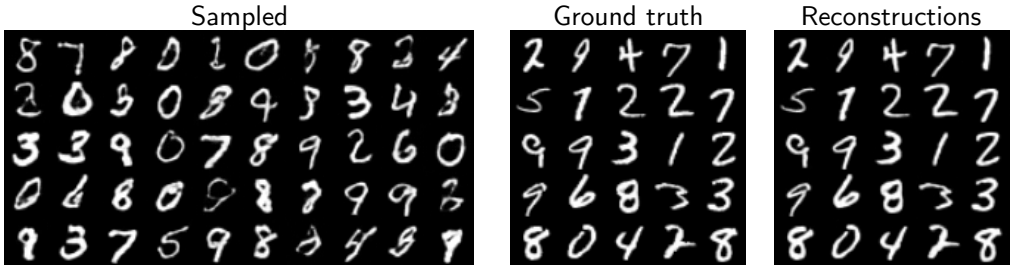
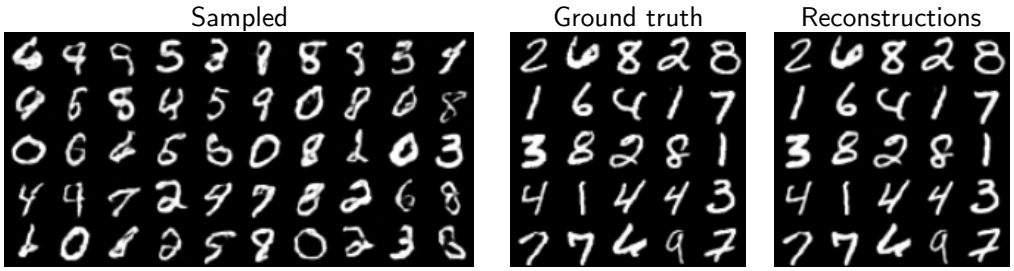


Figure 5.5: FID score of TRUMPET with sampling temperature.



(a) ReLU 1×1 convolutions



(b) Linear 1×1 convolutions

Figure 5.6: TRUMPETs trained with (a) ReLU and (b) linear 1×1 convolutions give similar sample quality.

(a) ReLU 1×1 convolutions

(b) Linear 1×1 convolutions

Figure 5.7: TRUMPETs trained with (a) ReLU and (b) linear 1×1 convolutions give similar sample quality. On the right, we showcase the reconstruction performance—the left column is ground truth and the right is our reconstruction (see Table 5.2 for quantitative assessment)

Figure 5.8: Generated samples on the Chest X-ray. On the right, we showcase the reconstruction performance—the left column is ground truth and the right is our reconstruction (see Table 5.2 for quantitative assessment)

Figure 5.9: Generated samples and reconstructions of original data on the CIFAR-10 dataset.

Figure 5.10: Generated samples on the celeba dataset with linear 1×1 convolution and $T = 0.85$.

Chapter 6

Deep Injective Prior for Inverse Scattering

In Chapter 5, we developed injective flows to solve linear inverse problems, demonstrating that Trumpets can achieve high-quality MAP estimates, posterior samples, and uncertainty estimates. In this chapter, we leverage Trumpets for a challenging non-linear inverse problem: electromagnetic inverse scattering. We show that Trumpets can generate high-quality posterior samples and provide a physically meaningful uncertainty quantification.

Electromagnetic inverse scattering is the problem of determining the electromagnetic properties of unknown objects from how they scatter incident fields. This non-destructive technique finds applications in various fields, such as early detection of breast cancer [219], mineral prospecting [220], detecting defects and cracks inside objects [221], imaging through the walls [222] and remote sensing [223].

While inverse scattering is well-posed and Lipschitz stable in theory, when full-aperture continuous measurements are available [224], it becomes a severely ill-posed inverse problem for a finite number of measurements. This means that even a small perturbation in the scattered fields can result in a significant error in the reconstructed permittivity pattern [9]. Additionally, the nonlinearity of the forward operator, caused by multiple scattering and amplified by higher permittivity contrasts [9], further complicates the inversion process. All these together make inverse scattering a challenging problem, especially for strong scatterers (objects with large permittivity) and noisy measurements. To address these challenges, an effective regularization technique is necessary to constrain the search space and achieve accurate recovery.

Several optimization-based methods have been proposed to tackle the nonlinearity and ill-posedness of the inverse scattering problem. These include the Born iterative method [225], distorted Born iterative method (DBIM) [226], contrast source inversion (CSI) [227], and subspace-based optimization (SOM) [228]. While these methods have demonstrated effectiveness in reconstructing objects with small permittivity variations,

they often fall short in accurately reconstructing objects with large permittivity contrasts. These methods typically rely on iterative optimization of a regularized objective, incorporating manually designed regularization terms [9].

Deep learning has achieved remarkable success in inverse scattering. Most deep learning models employed for inverse scattering adopt a *supervised* learning approach, which trains a deep neural network to regress the permittivity pattern. Some studies [23, 229, 230] have utilized scattered fields as the input of the neural network. Despite the satisfactory reconstructions [23], these methods are sensitive to changes in the experimental configuration, such as frequency, the number of transmitters and receivers or other real-world factors. Even slight variations in the distribution of scattered fields in test time can lead to a significant degradation in reconstruction quality, requiring costly acquisition of new training data. Back-projections can be used as input to tackle some of these issues [16, 231, 232]. While this approach yields good reconstructions for objects with small and moderate permittivity, due to the non-linearity the quality of back-projections significantly drops in large permittivity leading to a drop in the reconstruction quality [23]. Moreover, supervised learning methods are vulnerable to adversarial attacks [233], which is problematic in medical applications [234]. Importantly, incorporating the well-established physics of the scattering problem (i.e., the forward operator) to improve the generalization capability is not straightforward in such deep learning models [235–240].

To tackle these issues, we propose a deep learning approach to inverse scattering using Trumpet. The proposed method adopts an unsupervised learning framework—the training phase uses only the target permittivity patterns, and the physics of scattering is fully incorporated into the solution. Deep generative models such as generative adversarial networks (GANs) [181, 189], variational autoencoders (VAEs) [197], normalizing flows [70, 187, 192] and diffusion models [71] belong to a class of unsupervised learning methods and train a deep neural network to transform the samples of a simple (Gaussian) distribution into samples that resemble the target data distribution. Recently, deep generative models (DGM) have been used as a prior for solving inverse problems [11, 20, 21, 31, 66, 68]. By leveraging a trained generator on a dataset of target images (the solutions of a given inverse problem), one can explore the latent space of the generator to find a latent code yielding a solution that aligns with the given measurements.

The choice of generative model is of paramount importance to provide an effective regularization for solving *ill-posed* inverse problems. While GANs have been used as generative priors for inverse problems [66,156,166,241], they are unstable in training [190, 191] and result in local minima in iterative approaches [66]. Normalizing flows resolve some of these issues [67, 69, 201], however, they are computationally expensive to train and often do not provide sufficient regularization for highly ill-posed inverse problems. Injective normalizing flows [21, 161, 242], specifically designed for solving ill-posed inverse problems, alleviated these issues; they benefit from a low-dimensional latent space which serves as an effective regularizer for ill-posed inverse problems. In a related

work, Guo et al. [243] employed VAEs as generative priors for inverse scattering.

In this chapter, we use injective flows introduced in Chapter 5 as generative priors for full-wave inverse scattering. The proposed approach has a significant advantage: it only requires training on the target permittivity patterns and does not require any training data from scattered fields. Once the generator is trained, it can be used to solve inverse scattering problems in arbitrary configurations. This property endows the model with robustness against distribution shifts in the measurements as well as to adversarial attacks. In contrast to the work of Guo et al. [243], the invertibility of our generator allows us to perform optimization in both latent and data spaces, providing great flexibility in choosing the scattering solver. Additionally, while Guo et al. [243] require a data-driven initialization, our proposed method can leverage both back-projection and data-driven initializations (among others), making it adaptable to different scenarios and reducing dependence on the particularities of a specific starting point. We show that the proposed framework significantly outperforms traditional iterative solvers with reconstructions of comparable or better quality compared to highly successful supervised methods such as the U-Net [25].

All the aforementioned methods reconstruct a single point estimate from the permittivity pattern given the measurements. A point estimate, however, is often insufficient or misleading due to the ill-posedness of the inverse scattering problem. This limitation can be tackled by applying Bayesian frameworks based on deep learning networks [23, 244, 245] to generate multiple estimates of the permittivity and perform uncertainty quantification (UQ). However, these methods are supervised and suffer from the aforementioned issues. Our second contribution is to leverage our pre-trained injective generator to develop a Bayesian framework that produces multiple estimates of the permittivity pattern enabling the uncertainty quantification. Crucially, the proposed method does not rely on scattered fields during training. As we will discuss in Section 6.3, this framework requires injectivity and is thus not practicable with non-injective generators like GANs or VAEs.

This chapter is organized as follows. Section 6.1 provides a brief review of the forward and inverse scattering problem. Our proposed methods for MAP estimation and posterior modeling in inverse scattering are introduced in Sections 6.2 and 6.3. Computational experiments are presented in Section 6.4.

6.1 Forward and inverse scattering

We begin our discussion with equations governing the 2D forward and inverse scattering problem. We focus on the 2D transverse magnetic (TM_z) case, where the longitudinal direction is along $\hat{z}$. As depicted in Figure 6.1, we consider non-magnetic scatterers with permittivity ϵ_r situated in the investigation domain $\mathcal{D}_{\text{inv}}$, which is a $D \times D$ square. The scatterers are surrounded by a vacuum background with permittivity ϵ_0 and permeability

μ_0. The scatterers are illuminated by N_i plane waves with equispaced directions, and N_r receivers are uniformly positioned on a circle with radius R to measure the scattered fields. The forward scattering problem can be derived from the time-harmonic formulation of Maxwell's equations and can be expressed as follows [246],

$$\nabla \times (\nabla \times E^t(r)) - k_0^2 \epsilon_r(r) E^t(r) = i\omega\mu_0 J(r), \tag{6.1}$$

where E^t represents the total electric field which has only the E_z component in the TM_z case. In addition, $k_0 = \omega\sqrt{\mu_0\epsilon_0}$ denotes the wavenumber of the homogeneous background, and J corresponds to the contrast current density. The contrast current density, calculated using the equivalence theorem [247], is given by $J(r) = \chi(r)E^t(r)$, where $\chi(r) = \epsilon_r(r) - 1$ and is referred to as the contrast. Throughout this chapter, the time-dependence factor $\exp(i\omega t)$ with angular working frequency ω is assumed and will be suppressed for simplicity.

We discretize the investigation domain $\mathcal{D}_{\text{inv}}$ into $N \times N$ units. The state equation can be expressed as,

$$E^t = E^i + G_d \chi E^t, \tag{6.2}$$

where $G_d \in \mathbb{R}^{N^2 \times N^2}$ and E^t, E^i are the total and incident electric fields, respectively; χ is a diagonal matrix with elements $\chi(n,n) = \epsilon_r(n) - 1$ accounting for the contrast in the medium. On the other hand, the data equation is given by,

$$E^s = G_s \chi E^t + \delta, \tag{6.3}$$

where $G_s \in \mathbb{R}^{N_r \times N^2}$, E^s denotes the scattered electric fields, and δ is the additive noise in the measurements. It is worth mentioning that G_d and G_s have closed-form analytical expressions [9].

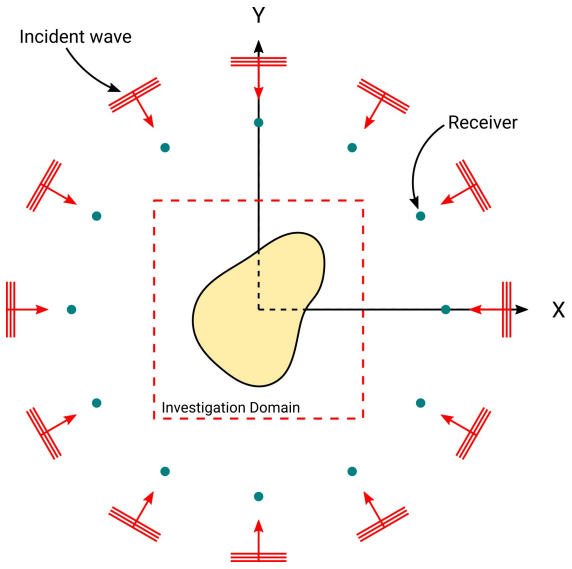


Figure 6.1: The setup for the inverse scattering problem, red arrows show the incident plane waves; the green circles are the receivers.

We combine (6.2) and (6.3) to obtain a unified expression for the forward model [9],

$$E^s = G_s \chi (I - G_d \chi)^{-1} E^i + \delta, \tag{6.4}$$

which represents a nonlinear mapping from χ to E^s. For convenience, we define a forward operator A that maps χ to E^s,

$$y = A(x) + \delta, \tag{6.5}$$

where $A(\cdot)$ corresponds to the nonlinear forward scattering operator,

$$A(\chi) = G_s \chi (I - G_d \chi)^{-1} E^i, \tag{6.6}$$

with $y = E^s$ and $x = \chi$. The objective of inverse scattering is to reconstruct the contrast χ from the scattered fields E^s, assuming that G_d, G_s, incident electric waves E^i, and hence the forward operator $A(\cdot)$ are known.

6.2 MAP inference with injective flows for inverse scattering

Inverse scattering with partial data is a severely ill-posed inverse problem, which means that a small perturbation in the measurements of scattered fields can result in a significant error in the recovered contrast [9]. As discussed in Section 6.1, inverse scattering is a nonlinear inverse problem, with the degree of nonlinearity being strongly influenced by the maximum contrast value. Particularly for objects with large contrasts, the problem becomes highly nonlinear, further increasing the difficulty of the inversion. In such cases, the presence of a robust regularizer that effectively constrains the search space becomes crucially important.

We model the contrast $\chi = x \in \mathcal{X}$ and the scattered fields $E^s = y \in \mathcal{Y}$ as random vectors. For simplicity, we assume that the additive noise δ in (6.5) is a random vector with Gaussian distribution $\delta \sim \mathcal{N}(0, \sigma^2 I)$ although our framework admits other distributions. With this assumption, the likelihood $p_{Y|X}$ can be expressed as,

$$p_{Y|X} = \mathcal{N}(A(X), \sigma^2 I). \tag{6.7}$$

An effective approach for solving ill-posed inverse problems is to compute the maximum a posteriori (MAP) estimate, where we seek the solution x that has the highest posterior likelihood given a measurement y,

$$x_{\text{MAP}} = \underset{x}{\operatorname{argmax}} \ \log p_{X|Y}(x|y), \tag{6.8}$$

where $p_{X|Y}(x|y)$ denotes the posterior distribution, representing the conditional distribution of the image of interest given the measurements y. The posterior distribution $p_{X|Y}$ can be computed using Bayes theorem as,

$$p_{X|Y}(x|y) = \frac{p_{Y|X}(y|x) p_X(x)}{\int_x p_{X,Y}(x, y) dx}, \tag{6.9}$$

which leads to the following expression for the MAP estimate,

$$x_{\text{MAP}} = \underset{x}{\operatorname{argmin}} \ -\log p_{Y|X}(y|x) - \log p_X(x). \tag{6.10}$$

From (6.7) we get

$$x_{\text{MAP}} = \underset{x}{\operatorname{argmin}} \ \frac{1}{2}\|y - A(x)\|_2^2 - \lambda \log p_X(x), \tag{6.11}$$

where the first term represents the data-consistency loss while $p_X(x)$ denotes the prior distribution of the contrast and yields a regularization term. We additionally insert λ as a hyperparameter to adjust the weight of the regularization term as its value depends on the unknown noise power. In general, estimating the prior distribution p_X is challenging, and a commonly used approximation is a Gaussian distribution with zero mean, leading to Tikhonov regularization. However, a Gaussian distribution often deviates significantly from the true prior, resulting in poor reconstructions.

This chapter explores a data-driven regularization in inverse scattering based on deep generative models. We leverage a training set of contrast patterns $\{x^{(i)}\}_{i=1}^N$ and train a deep generative model $x = f(z)$ to produce samples from (approximately) the same distribution as that of the training set. By sampling from a Gaussian distribution in the latent space $z \in \mathcal{Z}$, we expect the trained generator f to produce plausible contrast samples. This property of deep generative models makes them an effective regularizer for solving inverse problems [66, 243].

In this chapter, we employ injective flows, Trumpet introduced in Chapter 5, as a generative prior due to their suitability for addressing ill-posed inverse problems [21]. We perform optimization in the latent space to find the latent code that produces a permittivity pattern compatible with the measurements y. The optimization problem can be formulated as follows,

$$z_{\text{MAP}} = \underset{z}{\operatorname{argmin}} \ \frac{1}{2}\|y - A(f(z))\|_2^2 - \lambda \log p_X(f(z)), \tag{6.12}$$

where the regularization term $\log p_X$ is approximated via (5.1). The reconstructed contrast is then obtained as $x_{\text{MAP}} = f(z_{\text{MAP}})$. We call this method latent space optimization (LSO). We note that (6.12) has been previously proposed by [67, 201] for solving compressed sensing inverse problems using regular normalizing flows.

Unlike the supervised learning methods for inverse scattering [16, 230–232], which rely on paired training sets of contrast and scattered fields $\{(x^{(i)}, y^{(i)})\}_{i=1}^N$, our framework is unsupervised, without the need for scattered fields during training. This eliminates the need to retrain the model when the distribution of scattered fields changes due to variations in the experimental configuration. Once the injective generator is trained on the contrast samples, we can directly optimize (6.12) for new measurements to reconstruct the corresponding contrast. In addition, our proposed method fully leverages the underlying physics of the scattering problem by optimizing over the complex-valued scattered fields in (6.12). Kothari et al. [248] have demonstrated that incorporating wave physics into the neural network architecture can significantly enhance the quality of reconstructions, particularly for out-of-distribution data.

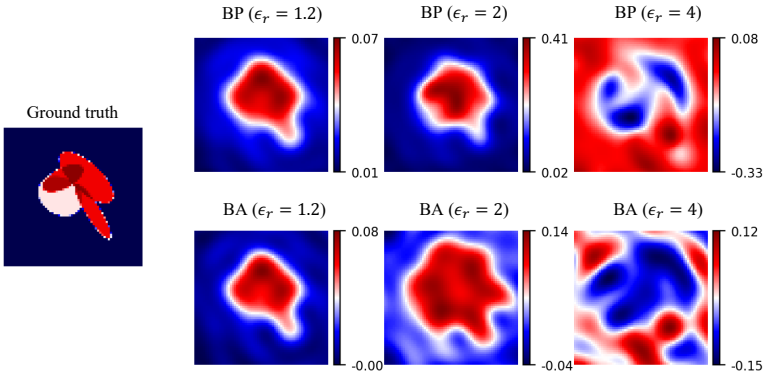


Figure 6.2: Performance analysis of the Back-Propagation (BP) and Born Approximation (BA) methods across objects with different maximum ϵ_r values. While both BP and BA reconstructions are visually meaningful for small ϵ_r, their performance significantly deteriorates for objects with larger ϵ_r.

Invertibility of the injective generator allows us to use an alternative method for (6.12) explained in Section 5.2.1 for linear inverse problems. This method performs the optimization directly in the data space. We call this method data space optimization (DSO) and formulate it as follows,

$$x_{\text{MAP}} = \underset{x}{\operatorname{argmin}} \ \frac{1}{2}\|y - A(g(g^{\dagger}(x)))\|_2^2 - \lambda \log p_X(x) \tag{6.13}$$

where $g(g^{\dagger}(x))$ represents the projection operator described in Section 5.1.3. Similar to LSO, the second term $\log p_X$ can be approximated using (5.1) and acts as an additional regularizer. In LSO the reconstructed point $x = f(z)$ always lies on the learned manifold; this is not the case for the DSO method, where the reconstructed image may deviate from the manifold. On the other hand, as we discuss next, DSO offers more flexibility in the choice of the initial guess.

The choice of initial guess is important for inverse scattering solvers. A poor initialization may result in convergence to poor local minima due to nonlinearity. A good initial guess facilitates efficient convergence to good minima. The authors of [226] used Born approximation as the initialization for the distorted Born iterative method (DBIM). A back-propagation (BP) solution was also used in [227, 249] as an initial guess of the contrast source inversion (CSI) method. Figure 6.2 shows the ground truth, back-propagation (BP), and Born approximations (BA) for an object with different maximum ϵ_r values. While BP and BA may yield satisfactory results for objects with small permittivity, their performance sharply drops for large ϵ_r (especially numerically) which makes them a poor initialization for strong scatterers.

In order to circumvent this issue, we adopt a data-driven initialization suggested in [67]; mean of the Gaussian distribution (MOG) in the latent space which is set to 0. The MOG initialization $z = 0$ provides a fixed initialization with respect to the

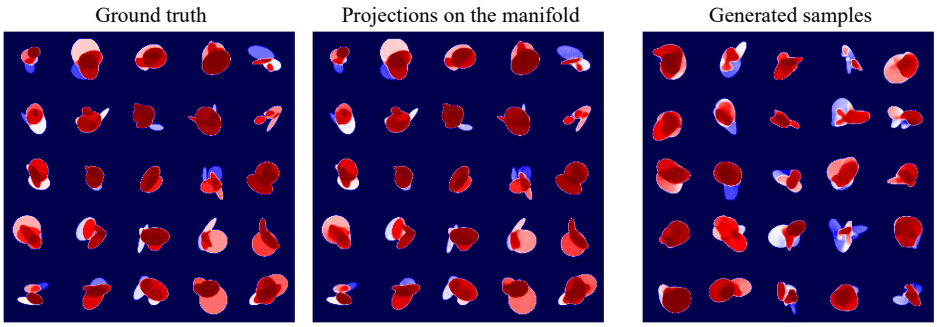


Figure 6.4: Performance evaluation of the trained injective flow on ellipses dataset; ground truth contrasts, their projections on the learned manifold and generated samples.

measurements (scattered fields); thereby being independent of the maximum contrast value and the problem configuration. This property leads to more robust convergence in both (6.12) and (6.13) even for objects with large permittivity. While the DSO method can be initialized with both BP and MOG, the LSO should exclusively be initialized with MOG. This is due to the possibility of BP being significantly distant from the range of the injective network, making inversion to the latent space infeasible. In Section 6.4, we will show that the MOG significantly improves the quality of the reconstructions compared to BP, especially for strong scatterers.

6.3 Posterior modeling and uncertainty quantification

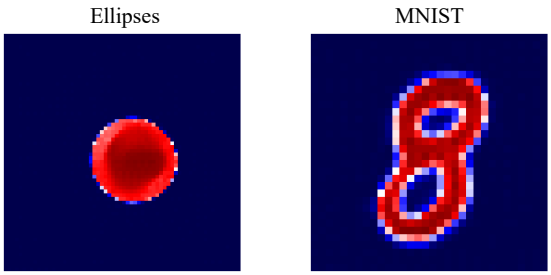


Figure 6.3: Illustration of the MOG initialization in the data space $f_\theta(z = 0)$ for ellipses and MNIST datasets.

Due to ill-posedness, there are an infinite number of contrasts that are consistent with the measurements within the noise level. These diverse solutions can lead to different scientific interpretations, highlighting the need to characterize their distribution. Relying on a single estimate, such as the MAP estimate obtained in the previous section, fails to reflect the inevitable uncertainty and pinpoint features recovered only with low confidence. To address this drawback of point estimates, we adopt a Bayesian perspective. Rather than solely computing the MAP estimate, we approximate the *full* posterior distribution $p_{X|Y}$ introduced in (6.9). By doing so, we can generate many posterior samples that explore plausible permittivity patterns.

The computation of the posterior distribution, as stated in (6.9), involves the integral $\int_x p_{X,Y}(x,y)dx$ which is intractable for high-dimensional imaging problems. Variational inference [250,251] is a promising framework that approximates the posterior distribution $p_{X|Y}(x|y)$ by defining a class of distributions $q_X(x;\psi)$ parameterized by ψ. The goal is to find the optimal ψ that ensures the "closeness" between $q_X(x;\psi)$ and $p_{X|Y}(x|y)$ for a given y. Examples of such approximators include Gaussian mixture models and distributions induced by deep generative models.

In variational inference, a commonly used measure of fit is the Kullback–Leibler (KL) distance,

$$\begin{aligned}\mathrm{KL}(q\|p) &= \int_{\mathcal{X}} q(x)\log\left(\frac{q(x)}{p(x)}\right)dx \\ &= \mathbb{E}_{x\sim q}[\log q(x) - \log p(x)].\end{aligned}$$

We optimize ψ to minimize the KL distance between $q_X(x;\psi)$ and $p_{X|Y}(x|y)$ for a given y,

$$\psi^\star = \operatorname*{argmin}_{\psi} \mathrm{KL}(q_X(x;\psi)\|p_{X|Y}(x|y)). \tag{6.14}$$

Sun et al. [194] parameterized $q_X(x;\psi)$ using an untrained normalizing flow through (5.1) and directly performed the optimization over the network's weights.

We propose to leverage our pre-trained injective flow f_θ as a prior to approximate the posterior distribution. Our approach relies on the following principle: when we apply an injective mapping to the distributions Q and P, resulting in new distributions Q' and P', respectively, the KL distance between Q' and P' remains the same as the KL distance between Q and P (refer to Section 6.6.2 in the appendix for further information). This property of injective mappings motivates us to approximate the posterior distribution in the *latent* space instead of the data space. Consequently, we minimize the KL distance between $q_Z(z,\psi)$ and $p_{Z|Y}(z|y)$ as follows,

$$\begin{aligned}\psi^\star &= \operatorname*{argmin}_{\psi} \mathrm{KL}(q_Z(z,\psi)\|p_{Z|Y}(z|y)) \\ &= \operatorname*{argmax}_{\psi} \mathbb{E}_{z\sim q_Z}\left[\log p_{Y|Z}(y|z)\right] - \mathrm{KL}(q_Z\|p_Z) \\ &= \operatorname*{argmin}_{\psi} \mathbb{E}_{z\sim q_Z}\left[\|y - A(f(z))\|_2^2\right] + \beta\big(\mathrm{KL}(q_Z\|p_Z)\big),\end{aligned} \tag{6.15}$$

where $p_Z = \mathcal{N}(0,I)$ represents the prior distribution introduced in (5.1). We consider β as a hyperparameter to control the diversity of the posterior samples as its value depends on the unknown noise power.

Now we must select our posterior approximator $q_Z(z,\psi)$. While previous works [21,252] used an additional normalizing flow to model $q_Z(z,\psi)$, we use a Gaussian distribution for simplicity and computational efficiency. Specifically, we define $q_Z(z,\psi) =$

$\mathcal{N}(z; \mu_q, \text{diag}(\sigma_q))$, where $\psi = (\mu_q, \sigma_q)$ represents our variational parameters. This Gaussian parameterization of $q_Z(z, \psi)$ simplifies the KL term in (6.15) since there exists a closed-form expression for the KL distance between two Gaussian distributions,

$$\text{KL}(q_Z \| p_Z) = \frac{1}{2} \sum_{i=1}^{d} \sigma_q(i)^2 + \mu_q(i)^2 - 1 - 2 \log \sigma_q(i), \tag{6.16}$$

where $\mu_q(i)$ and $\sigma_q(i)$ denote the ith element of $\mu_q \in \mathbb{R}^d$ and $\sigma_q \in \mathbb{R}^d$, respectively. Furthermore, since we have already obtained the MAP estimate in the latent space through (6.12), we set $\mu_q = z_{\text{MAP}}$ and only optimize σ_q.

We cannot directly optimize (6.15) using gradient-based methods since optimization variables are inside the expectation. We thus use the reparameterization trick [197, 253], letting $z = z_{\text{MAP}} + \sigma_q \odot t$, where $t \sim \mathcal{N}(0, I)$ and $\odot$ denotes the element-wise multiplication. By substituting (6.16) into (6.15) and incorporating the above reparameterization,

$$\begin{aligned} \sigma_q^\star = \underset{\sigma_q}{\text{argmin}}\, \mathbb{E}_{t \sim \mathcal{N}(0,I)} \Big[\| y - A(f(z_{\text{MAP}} + \sigma_q \odot t)) \|_2^2 \Big] \\ + \beta \sum_{i=1}^{d} \Big(\sigma_q(i)^2 - 2 \log \sigma_q(i) \Big). \end{aligned} \tag{6.17}$$

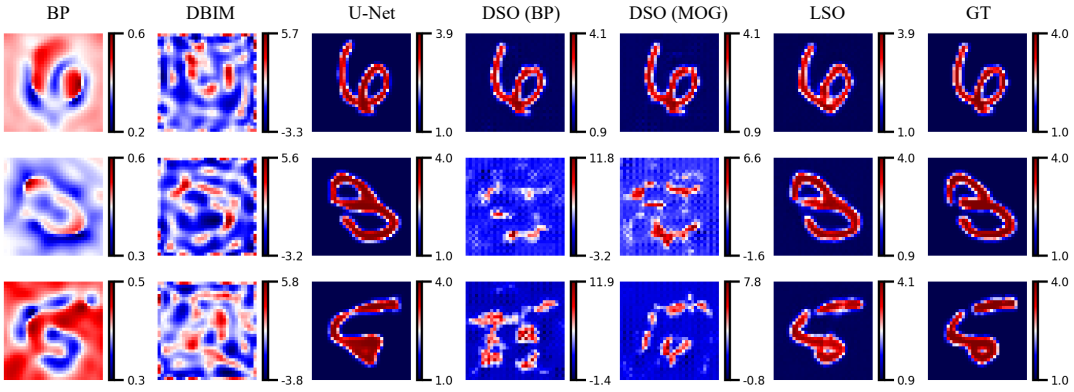
To evaluate the expectation, we compute the average over K iid samples drawn from the standard normal distribution,

$$\begin{aligned} \sigma_q^\star \approx \underset{\sigma_q}{\text{argmin}} \sum_{k=1}^{K} \Big(\| y - A(f(z_{\text{MAP}} + \sigma_q \odot t_k)) \|_2^2 \Big) \\ + \beta \sum_{i=1}^{d} \Big(\sigma_q(i)^2 - 2 \log \sigma_q(i) \Big). \end{aligned} \tag{6.18}$$

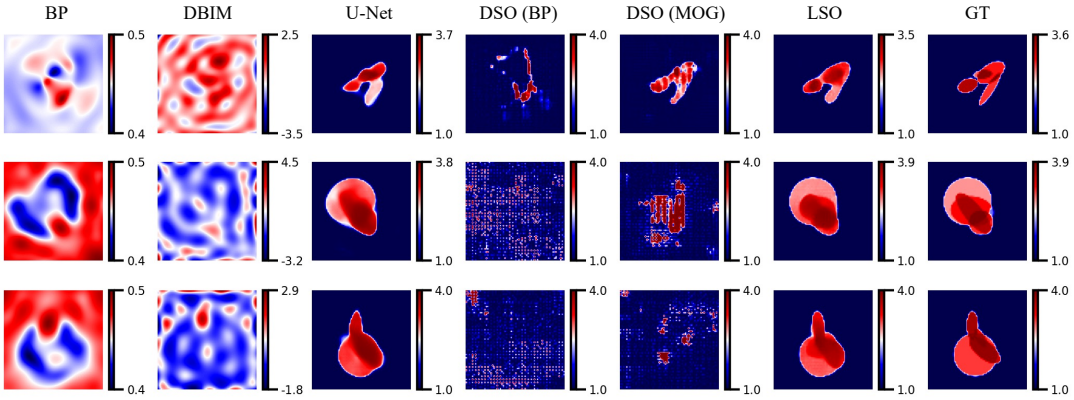
Once we obtain the optimal $\sigma_q^\star$, we can generate posterior samples $x_{\text{post}} = f(z_{\text{MAP}} + \sigma_q^\star \odot t)$ where $t \sim \mathcal{N}(0, I)$. Additionally, we can evaluate the empirical minimum mean-squared error (MMSE) estimate and the associated uncertainty by calculating the pixel-wise average and standard deviation over multiple posterior samples.

6.4 Experiments

We assess the performance of the proposed methods for MAP estimation and posterior modeling on synthetic and experimental data. We train the model on two synthetic large-scale datasets: 1) MNIST [204] with 60000 training samples in the resolution $N = 32$,



(a) MNIST in resolution 32×32



(b) Ellipses in resolution 64×64

Figure 6.5: Performance comparison of different methods for objects with maximum $\epsilon_r = 4$.

and 2) a more challenging dataset we generated comprising 60000 training samples with resolution $N = 64$ of overlapping ellipses used in [23]. Figure 6.4 shows example test contrasts, their projections on the learned manifold, and the samples generated by the injective network, verifying the ability of the model to produce outputs of good quality. For additional details about the network architecture and training, please refer to Section 6.6.1 in the appendix.

6.4.1 Synthetic data

In experiments with synthetic data, the task is to reconstruct the test samples from MNIST and ellipses datasets that have not been "seen" by the injective network during training. We use $N_i = 12$ incident plane waves and $N_r = 12$ receivers, uniformly distributed on a circle with radius $R = 20$ cm around the object with maximum permittivity ϵ_r and dimension $D = 20$ cm. The working frequency is 3 GHz and we added 30 dB noise to the measurements of the scattered fields.

MAP estimation We conduct a comprehensive evaluation of the DSO and LSO methods. We consider the MOG and BP initializations for DSO while only using the MOG initialization for LSO. We compare the performance of our proposed methods with a traditional iterative method, DBIM [226]. While our approach is unsupervised so that the scattered fields are not used during training, we also compare its performance with a supervised learning method, the U-Net [25], which has enjoyed tremendous empirical success in a variety of imaging inverse problems including inverse scattering [16]. The U-Net takes the BP image as input and regresses the corresponding permittivity.

We have fully implemented the forward operator in Tensorflow [159], enabling efficient GPU utilization for parallel reconstruction of multiple samples. Moreover, it allows us to use a variety of optimizers provided in Tensorflow including Adam [97] and L-BFGS [254]. In these experiments, we optimize (6.12) and (6.13) using the Adam optimizer with a learning rate of 0.05 for 300 iterations as it leads to more accurate reconstructions compared to L-BFGS. We set $\lambda = 0.01$ for BP and $\lambda = 0$ for MOG. For the MOG initialization, we begin from high-likelihood regions (mean of the Gaussian), viewed as a hidden regularizer and we thus set $\lambda = 0$. Figure 6.3 illustrates the MOG initializations for ellipses and MNIST datasets.

Figure 6.5 shows the performance of various methods for $\epsilon_r = 4$ using 5 test samples from MNIST and ellipses datasets. While DBIM falls short in this challenging task with a high contrast and 30 dB noise, DSO and LSO exhibit much better reconstructions. Moreover, the MOG initialization, as expected, yields superior reconstructions compared to BP. Notably, LSO outperforms DSO, demonstrating the advantages of running optimization in the latent space as discussed in Section 6.2. Despite not utilizing scattered fields during the training phase, LSO produces reconstructions of comparable or even superior quality

Table 6.1: Performance of different methods for solving inverse scattering ($\epsilon_r = 4$) averaged over 5 test samples.

	PSNR		**SSIM**	
	MNIST	**Ellipses**	**MNIST**	**Ellipses**
BP	7.75	7.00	0.01	0.01
DBIM [226]	5.77	4.67	0.01	0.01
U-Net [25]	24.26	**21.94**	**0.90**	0.82
DSO (BP)	8.73	7.89	0.16	0.16
DSO (MOG)	17.47	14.56	0.61	0.44
LSO (MOG)	**25.22**	20.50	0.89	**0.85**

to the supervised method U-Net. Table 6.1 lists the numerical results in PSNR and SSIM averaged over 5 test samples.

As discussed in Section 6.2, the maximum ϵ_r of the object plays a significant role in the performance of inverse scattering solvers. Figure 6.6 shows the performance of various methods across different maximum ϵ_r values on MNIST. This analysis shows that LSO, combined with the MOG initialization, remains effective even for objects with high ϵ_r, which highlights the significance of data-driven initialization and optimization in the latent space.

Regarding the computational efficiency, we used a single Tesla V100 GPU for training and solving the inverse scattering problem where each iteration of LSO (or DSO) takes 0.08 seconds at the resolution of $N = 32$ and 0.25 seconds at the resolution of $N = 64$. Although good estimates can be obtained with much fewer iterations, we empirically determined that 300 iterations ensure good convergence.

Posterior Sampling and UQ As explained in Section 6.3, we approximate the posterior distribution of contrast as a pushforward of a Gaussian around the MAP estimate in the latent space; the covariance is chosen to obtain the best variational approximation of the posterior in the sense of the KL divergence. We use the MAP estimate obtained from the LSO method in the previous section and optimize (6.18) using the Adam optimizer with a learning rate of 0.01. The initial value for σ_q is set as an all-one vector, and we use $K = 25$ random samples drawn from the standard Gaussian in each iteration. To compute the MMSE estimate and UQ, we calculate the pixel-wise average and standard deviation over 25 posterior samples. Figure 6.7 showcases 4 posterior samples along with UQ and MMSE estimates for $\beta = 0.01$ and $\beta = 0.05$. As expected, larger β values lead to more diverse posterior samples. The UQ map identifies regions with higher uncertainty visually represented in red. This information is highly valuable for conducting a more

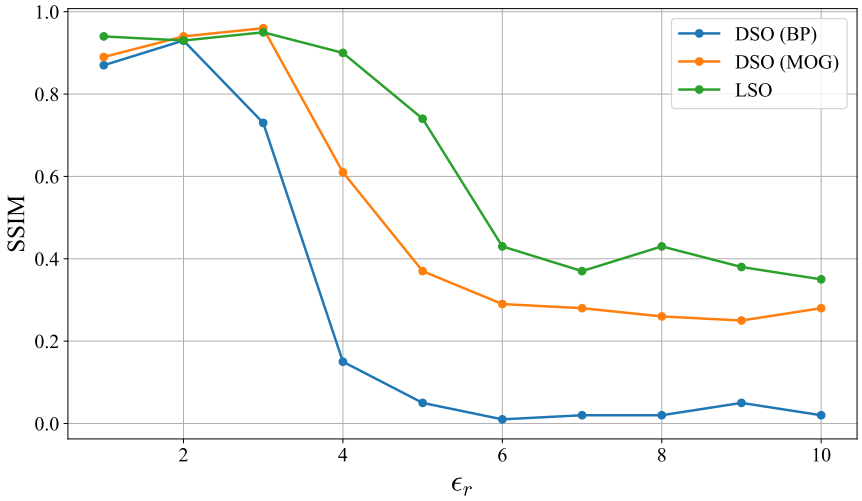


Figure 6.6: Performance of various methods across objects with different maximum ϵ_r values on the MNIST dataset.

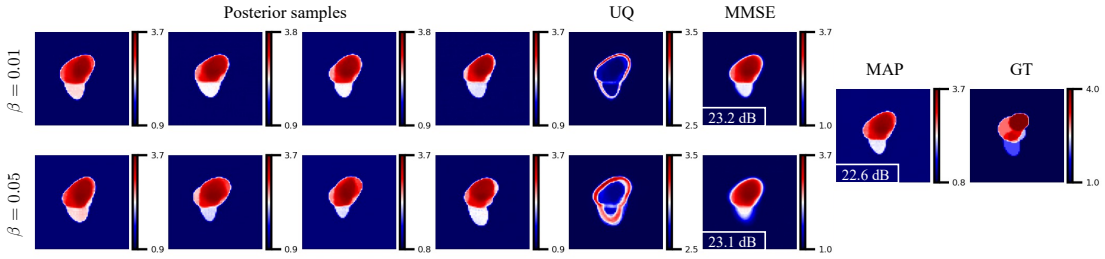


Figure 6.7: Posterior samples, UQ, MMSE, and MAP estimates for an object with $\epsilon_r = 4$ for $\beta = 0.01$ and $\beta = 0.05$; as expected, larger β values lead to more diverse posterior samples.

thorough and informed analysis. Finally, the MAP estimate is sharper than the MMSE as expected.

Generalization In this section, we evaluate the generalization performance of the proposed method under out-of-distribution changes in the permittivity patterns. We train injective flows exclusively on MNIST digits 0-5 and use the remaining digits for testing. The LSO solver is configured with the same setup as in the previous section. Figure 6.8 shows the posterior samples, UQ, MMSE, and MAP estimates for two test samples of digits 6 and 8 with $\beta = 0.05$. This experiment clearly shows the effectiveness of the proposed method in handling out-of-distribution data. We should point out that there exists a trade-off between regularization power and generalization performance, governed by the dimension of the latent space. Larger latent space dimensions yield

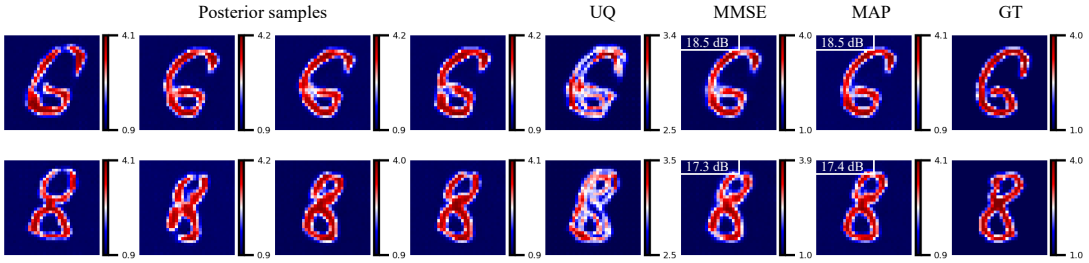


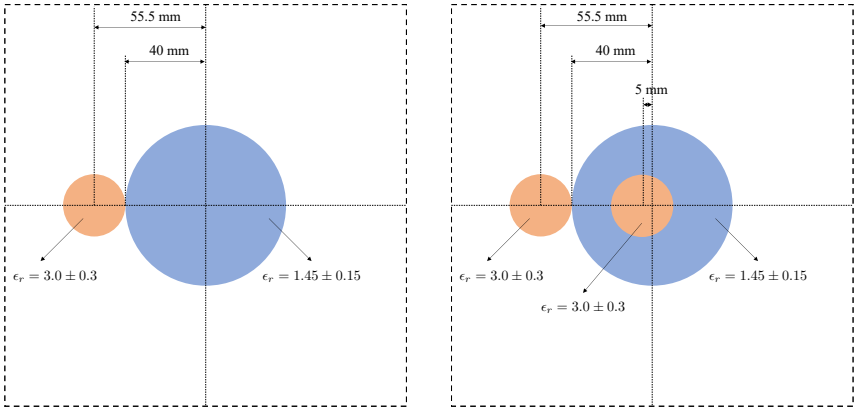
Figure 6.8: Reconstructions and UQ for out-of-distribution samples with $\epsilon_r = 4$. Despite being trained solely on MNIST digits 0-5, the proposed method exhibits excellent generalization by accurately reconstructing digits 6-9.

better generalization but less effective regularization. This has also been observed in regular normalizing flows, where matching dimensions in the latent and data space result in excellent generalization over out-of-distribution data but less effective regularization [67, 69].

6.4.2 Experimental data

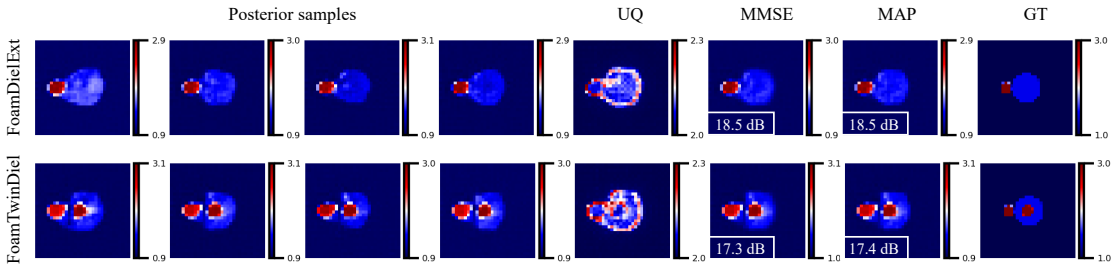
We finally evaluate our proposed model on FoamDielExt and FoamTwinDiel: real experimental data for two phantoms provided by the Institute Fresnel in Marseille, France [255]. In these experiments, there are $N_i = 8$ transmitters and 241 receivers located on a circle with radius $R = 1.67$ m. Out of those, we only use $N_r = 20$ receivers to make the inversion more challenging. Additional details about the setup are discussed in [255]. As shown in Figure 6.9, FoamDielExt and FoamTwinDiel consist of dielectric cylinders in a vacuum background. We use the measurements at the working frequency of 3 GHz, and the side length of the investigation domain is $D = 20$ cm.

We use two pre-trained injective flows on the ellipses dataset for resolutions $N = 32$ and $N = 64$. The inverse scattering problem is solved using (6.12) for MAP estimation and (6.18) for posterior modeling. We added the total-variation (TV) regularization term to (6.12) and (6.18) to further improve the quality of the reconstruction. The TV-norm multiplier is 0.1 and 0.08 for resolutions $N = 32$ and $N = 64$, respectively. Figure 6.10 shows posterior samples, UQ, MMSE, and MAP estimates. Despite the idealized forward operator and the substantial dissimilarity between the ground truth (two or three circles) and the training data (combinations of four ellipses with random positions and contrasts), the proposed framework produces satisfactory reconstructions. This experiment illustrates the robustness of the proposed method to noise and variations in experimental configuration. It also showcases the importance of posterior modeling: while the MAP and MMSE estimates in Figure 6.10a wrongly reconstruct the larger circle as compared to the ground truth, the uncertainty maps clearly signal that this part of the recovered contrast is not reliable.

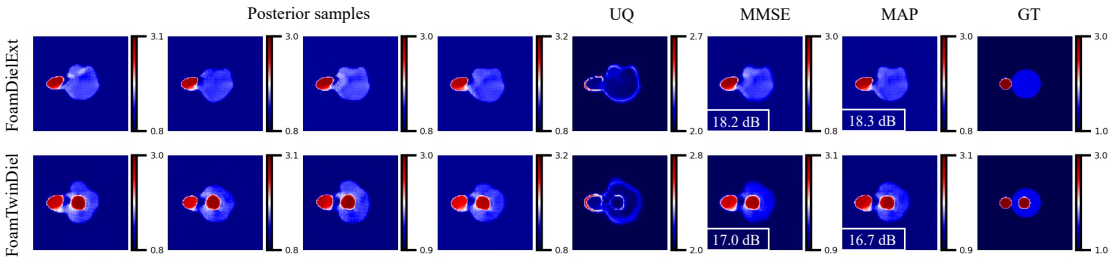


(a) FoamDielExt (b) FoamTwinDiel

Figure 6.9: Experimental Fresnel data [255]



(a) Resolution 32×32



(b) Resolution 64×64

Figure 6.10: Posterior samples, UQ, MMSE, and MAP estimates for experimental Fresnel data. The uncertainty maps clearly signify the importance of posterior modeling by assigning higher uncertainty to wrongly reconstructed areas (red regions).

6.5 Summary

We proposed a data-driven framework for inverse scattering using an injective prior. The proposed method fully exploits the physics of wave scattering while benefiting from a data-driven initialization resulting in a powerful solver even for objects with a large contrast. The invertible generator admits optimization in both latent and data space and uses either a data-driven initialization or a back-projection. We showed that optimization in the latent space and with the latent Gaussian center as the initial guess significantly outperforms traditional iterative methods and even gives reconstructions comparable to a strong supervised method, the U-Net.

The proposed framework has several key limitations. It requires running an iterative method at test time, which is slow and impractical for real-time applications. Moreover, iterative methods can converge to local minima even with clever initialization. To speed up convergence, one may consider a more accurate initial guess by exploiting physics in the data-driven initialization via a combination of traditional back-projection (like BP) and data-driven initializations (like MOG). Furthermore, while the L-BFGS optimizer didn't improve the convergence rate in our experiments, other Newton's family optimizers may improve the convergence rate as shown in [243]. Additionally, forcing the reconstruction to be within the range of an injective flow can introduce undesired bias and artifacts in certain applications. Recently, Hussein et al. [166] optimized the generator weights with a small rate after finding the optimal latent code in (6.12) to further improve the reconstructions; this idea might be adapted to our framework. We leave addressing these limitations for future work.

6.6 Appendix

6.6.1 Network architecture and training details

The injective subnetwork g_γ is composed of 6 injective revnet blocks described in Section 5.6.1, each increasing the dimension by a factor of 2. To enhance the expressiveness of the model, we insert 36 bijective revnet blocks between them. We choose a latent space of dimension 64 which provides a compression rate of 98.5% for resolution $N = 64$ and 93.7% for resolution $N = 32$. The bijective subnetwork h_η is constructed using 20 bijective revnet blocks.

We normalize the training data between 0 and 1 before training the model. We then multiply the output of the trained network by the maximum contrast of the dataset before using it as the generative prior. We train the injective subnetwork g_γ for 150 epochs to ensure the training samples (contrast patterns) align with the generator's range. Following this, we train the bijective subnetwork h_η for 150 epochs to maximize the likelihood of

the training samples in the intermediate space.

6.6.2 The Invariance of KL distance under injective mappings

Here we show that the KL distance is invariant under injective maps. Assume that probability distributions q_Z and p_Z have the same support. Let $q_X = f_{\#}q_Z$ and $p_X = f_{\#}p_Z$ where $f_{\#}p$ denotes the pushforward of p via mapping f, i.e., if x is distributed according to p, $f(x)$ is distributed according to $f_{\#}p$ [1].

As discussed in 5.2, the change of variable for the injective mapping f yields [199],

$$\log p_X(x) = \log p_Z(z) - \frac{1}{2}\log\left|\det[J_f(z)^T J_f(z)]\right|, \tag{6.19}$$

where $z = f^{\dagger}(x)$ and is valid for $x \in \text{Range}(f)$. We can now compute as follows,

$$\begin{aligned}
\text{KL}(q_X \| p_X) &= \mathbb{E}_{x\sim q_X}[\log q_X(x) - \log p_X(x)] \\
&= \mathbb{E}_{x\sim q_X}[\log q_Z(z) - \frac{1}{2}\log|\det[J_f(z)^T J_f(z)]| \\
&\quad - \log p_Z(z) + \frac{1}{2}\log|\det[J_f(z)^T J_f(z)]|] \\
&= \mathbb{E}_{x\sim q_X}[\log q_Z(z) - \log p_Z(z)] \\
&= \mathbb{E}_{z\sim q_Z}[\log q_Z(z) - \log p_Z(z)] \\
&= \text{KL}(q_Z \| p_Z),
\end{aligned}$$

which is what we wanted to show.

[1] For simplicity we lightly abuse notation by identifying a probability measure and its density.

Chapter 7

Conditional Injective Flows for Bayesian Imaging

In Chapter 5, we developed injective flows for posterior sampling and uncertainty quantification of various imaging problems. The proposed image recovery algorithm involves an iterative process that is slow and can be computationally expensive particularly for inverse problems with a costly forward operator like inverse scattering. In this chapter, we build a Bayesian architecture based on injective flows in the context of *amortized inference* enabling rapid posterior sampling and uncertainty quantification.

In Bayesian modeling of computational imaging problems, we assume that the (unknown) object of interest x and the observed measurements y are realizations of random vectors $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ with a joint distribution $p_{X,Y}$. This general model includes the common setting of a deterministic forward operator and additive Gaussian noise,

$$y = A(x) + \epsilon, \tag{7.1}$$

where

$$Y|X \sim \mathcal{N}(A(X), \sigma^2 I) \tag{7.2}$$

as well as other relevant models like $Y = \text{Poisson}(A(X), \lambda)$ or even random or uncertain forward operators A.

Common machine-learning approaches to **ill-posed** inverse problems yield point estimates, that is, they output a single reconstruction. For example, training a deep neural network f_θ with the mean-squared error (MSE) loss $\mathbb{E}\,\|X - f_\theta(Y)\|^2$ approximates the minimum-mean-squared-error (MMSE) estimator[1] $\mathbb{E}[X|Y]$ (the posterior mean) [14].

In many situations, however, a single point estimate can be misleading or incomplete. For example, in radio interferometric imaging which aims to reconstruct astronomical images from radio telescope measurements, there can be multiple solutions that fit the

[1] Often called the regression function in machine learning and statistics.

observed measurements; a now-famous example is the imaging of a black hole [194]. This can happen for a variety of reasons, all stemming from the ill-posedness of the imaging problem. The posterior may be multimodal, in which case the MMSE estimator blends the modes together and maximum a posteriori estimate (MAP) (or posterior mode) $\operatorname{argmax}_x p_{X|Y}(x|y)$, returns only one of the many modes. Even when the posterior is unimodal, providing a point estimate when measurements have low signal-to-noise ratio does not convey the amount of uncertainty in the estimate, thus requiring cautious interpretation. As a remedy, uncertainty quantification (UQ) on top of a reconstructed image can greatly help medical professionals make more informed decisions or order additional measurements in uncertain regions [256].

In Figure 7.1 we use a toy problem to illustrate access to the posterior can help in the presence of multiple modes. Both standard point estimates look close to a "9", but many posterior samples look like "4"s. While innocuous on MNIST [204], such failure modes come with risks in medical imaging.

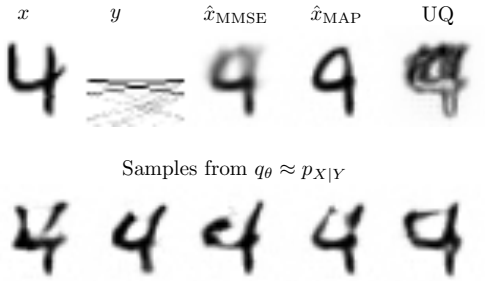


Figure 7.1: Travel-time tomography with boundary sensors; we measure the wave travel time between sensors in Figure 7.6c. The column x shows the ground truth, the column y the pseudo-inverse of the forward operator applied to the measurements. Conditional injective flows provide MMSE, surrogate MAP (Section 7.3.3) and physically meaningful uncertainty quantification (UQ). Given no sensors in the top regions of the image, the trained C-Trumpet assigns higher uncertainty to the top half of the domain. While both standard point estimates look close to "9", many posterior samples look like "4"s which indicates the significance of posterior sampling for ill-posed inverse problems.

Another approach is then to characterize the posterior $p_{X|Y}$ upon observing y. Given the joint distribution $p_{X,Y}$, computing the posterior $p_{X|Y}$ generally involves intractable high-dimensional integrations. A standard way to circumvent this is by sampling methods such as Markov chain Monte Carlo (MCMC) [257, 258]. These methods work well in low-dimensional problems but become computationally intensive when used in high-dimensional imaging tasks due to the need to compute the forward operator many times [259]. An alternative is to perform *variational inference*. This requires defining a tractable, parameterized family of distributions $\mathcal{Q}$ and choosing a $q \in \mathcal{Q}$ that is "close" to $p_{X|Y}$.

In this chapter, we define a new class of approximate posteriors $\mathcal{Q}$ using injective deep neural networks, suitable for imaging problems. We build on the Trumpet introduced in Chapter 5 and conditional coupling layers [260]. *Conditional* normalizing flows [260] inherit the favorable properties of their non-conditional counterparts—easy access to the likelihoods and inverses of generated samples. They were applied to inverse

problems [261, 262] where they enable posterior estimation, efficient sampling and even uncertainty quantification. In this work, we propose **conditional injective flows** termed C-Trumpets. While injective flows model image datasets supported on low-dimensional manifolds, the range of a C-Trumpet is a (potentially) different low-dimensional manifold of posterior samples *for each* measurement. As we show in Section 7.4, this makes them an effective model for data distributions supported on *fiber bundles* with the base space corresponding to the space of measurements $\mathcal{Y}$, and in particular effective models of posterior distributions in imaging inverse problems. Moreover, thanks to a low-dimensional latent space, they can be trained for high-dimensional data (256×256) using a single GPU while training conditional bijective flows at this resolution requires significantly more memory.

While generating approximate posterior samples is important, many applications still call for Bayesian point estimates such as the MAP or the MMSE estimator, and it is convenient if those can be computed with the same model. While it is clear (at least conceptually) how to do it for the MMSE estimator—generate many samples and average them—the MAP estimator is more elusive. There are not many deep learning approaches to imaging which compute (or approximate) the MAP estimator since the corresponding training loss (purely formally) is the highly irregular $\delta(x - x')$ as opposed to the "nice" $\|x - x'\|^2$ [263]. (A notable exception is amortized MAP for image super-resolution [264], although it is limited to noiseless linear low-rank projections.) We could in principle obtain a MAP estimate from C-Trumpets via iterative maximization; however, that is slow and it is not guaranteed to converge. We thus propose a modification of coupling layers which results in a fixed volume change with respect to the input. We use this newly designed layer to efficiently approximate MAP estimators without the need to run an iterative process or evaluate the forward operator.

Our main contributions can be summarized as follows:

- We define a class of deep learning models for amortized variational inference called C-Trumpets, with a smaller memory and compute footprint compared to bijective flows; C-Trumpets can be trained on 256×256 images on a single V100 GPU in a day.
- The new flows include bespoke architectural innovations: fixed-volume-change layers provide efficient MAP estimates without iterative optimization, while skip connections improve the quality of the generated samples and uncertainty quantification.
- We show that C-Trumpets outperform conditional bijective flows in solving computational imaging problems including nonlinear electromagnetic scattering, limited-view CT and seismic travel-time tomography; C-Trumpets produce much better posterior samples and uncertainty estimates that are consistent with the physics of the forward operators in various inverse problems.

- While standard injective flows parameterize manifolds and are thus a natural (regularizing) choice when we believe the manifold assumption holds, we show that conditional injective flows can parameterize fiber bundles [265].

7.1 Related works

There is by now a very large body of work on solving inverse imaging problems using deep neural networks. On the supervised regression end of the spectrum arguably the most important architecture is the U-Net [25]. It has been applied with great success to a variety of imaging problems including CT [14], magnetic resonance imaging (MRI) [15] and electromagnetic inverse scattering [16]. Its success may be attributed to the particular multiscale structure [27, 164] which matches both the physical description of the imaging problems and the representation of the involved image classes. Many alternatives have been proposed for specific problems where the assumptions that make the U-Net a natural choice do not hold, for example for wave-based problems [248, 266].

On the other hand, trained generative models have been shown to be effective priors [66, 267] in ill-posed inverse problems that can be trained in an unsupervised manner. Normalizing flows in particular were used to approximate MAP estimates using iterative optimization [201, 267]. However, normalizing flows are bijective, requiring large memory and compute budget even at moderate resolution. Moreover, they lack a low-dimensional latent space which has been shown to effectively regularize the inversion. Brehmer and Cranmer [161] proposed the first injective model for densities supported on low-dimensional manifolds. Kothari et al. [21] proposed injective flows with significantly optimized compute and memory requirements which were shown to outperform earlier variants of injective and bijective flows in computational imaging problems.

A variety of methods have been proposed for Bayesian imaging. The latter aims to approximate posterior distribution and / or the various point estimators related to the posterior. The authors of [268] consider a *convex* log-likelihood function for posterior distribution around the MAP estimate, which is suitable for modeling unimodal posteriors. Normalizing flows have been proposed as variational approximators to the posterior distribution for a given measurement [194]. The authors of [21,252] propose to train a flow model to approximate the posterior corresponding to a prior which is also modeled using a flow model. More recently, pre-trained GANs were used in conjunction with MCMC to generate posterior samples in non-linear inverse problems [269]. The authors of [270] leveraged the low-dimensional latent space of variational autoencoders to efficiently sample from posterior for MRI image reconstruction. The authors of [271, 272] use a style-GAN generator [156] to regularize ill-posed inverse problems and generate posterior samples. All these methods train a new generative model or run an iterative process for every measurement, which makes them slow when applied to multiple reconstructions. Further, training for each conditioning sample requires many calls to the forward operator.

To tackle these issues, one may consider *amortized inference* to make the generative models conditioned based on the measurements.

Conditional versions of generative adversarial networks (GANs) [189, 273] and variational autoencoders (VAEs) [197, 274] rely on injecting conditioning data into the different layers of the generator model. However, the lack of access to the posterior distribution make conditional GANs difficult to be used in inference tasks. On the other hand, VAEs provide lower bounds on likelihoods of generated samples. While these bounds can be made tighter by importance weighting [275], C-Trumpets allow one to compute exact end-to-end likelihoods via stochastic estimation, as well as to obtain fast exact values of likelihoods before the high-dimensional expansion. These generative models also suffer from mode collapse and training instabilities. Conditional normalizing flows were introduced in [260] to estimate the posterior by modifying the scale and shift terms of the coupling layers. The authors of [261] additionally make the mean and covariance of the base Gaussian distribution depend on the measurements. More recently, [276] proposed to append the measurements to all layers of the Glow network [70] in order to enable greater information flow from conditioning data to the generated samples. A different approach to conditioning the flow models has been developed by [277, 278], benefitting from two parallel flow models for simultaneously modeling of target and conditioning samples. All these conditional normalizing flows are bijective mappings from latent space to the target domain for each measurement. It is worth mentioning that all these models can be used in the bijective part of C-Trumpets to exploit their advantages in posterior modeling. On the side of theory, Puthawala et al. [279] establish universality of density and manifold approximation of injective flows such as those in [21, 161].

7.2 Variational Bayesian inference

In this section, we work formally and assume that all probability measures have a density; this allows us to simply present the main ideas. A training strategy suitable for distributions supported on low-dimensional manifolds is detailed in Section 7.3.

We consider random vectors $X \in \mathcal{X}$ (representing the unknown object) and $Y \in \mathcal{Y}$ (representing the observed noisy measurements). As discussed in Section 6.3, the posterior distribution $p_{X|Y}$ can be expressed by (6.9). In high-dimensional imaging applications, calculating $\int_x p_{X,Y}(x, y)dx$ is intractable. Moreover, the prior distribution p_X is unknown and needs to be estimated [41, 269]. MCMC-based sampling methods do not require access to $\int_x p_{X,Y}(x, y)dx$ but they are slow if one desires likely posterior samples [259,280]. An alternative to MCMC is to perform variational inference [250,251] as discussed in Section 6.3: we define a parameterized class of distributions

$$\mathcal{Q} = \{q_\theta \in \mathcal{P}(\mathcal{X}) \ : \ \theta \in \Theta\},$$

where $\mathcal{P}(\mathcal{X})$ is the space of probability distributions over $\mathcal{X}$, and we search for θ such that q_θ is close to $p_{X|Y}$. Examples of $\mathcal{Q}$ include Gaussian mixtures and densities induced by generative neural networks.

The remaining ingredients are a measure of "closeness" and a fitting algorithm. In Section 6.3, we used Kullback–Leibler (KL) divergence as a measure of fit. Given a measurement y, we choose q_θ by solving either

$$\theta^*_{\text{rev}}(y) = \operatorname*{argmin}_{\theta\in\Theta} \ \mathrm{KL}(q_\theta \| p_{X|Y}(\cdot \,|\, y)),$$

or

$$\theta^*_{\text{fwd}}(y) = \operatorname*{argmin}_{\theta\in\Theta} \ \mathrm{KL}(p_{X|Y}(\cdot \,|\, y) \| q_\theta),$$

respectively called reverse and forward KL minimization [258, 281]. The two estimates are in general different due to the asymmetry of the KL divergence.

In Section 6.3, we used reverse KL to approximate the posterior distribution. As discussed, minimizing the reverse KL requires computing the expectation $\mathbb{E}_{X\sim q_\theta} \log p_{Y|X}(y|X)p_X(X)$. While $p_{Y|X}$ is usually assumed known in imaging problems with a known forward operator—in (7.2) it corresponds to additive Gaussian noise—the prior distribution p_X is unknown. In Section 6.3, normalizing flows were used to estimate p_X to then allow for downstream minimization of the reverse KL divergence. The authors of [282] leveraged variational autoencoders to approximate the prior density. On the other hand, minimizing the forward KL does not require access to the prior distribution, the noise model or the forward operator [283, 284].

Computing $\theta^\star(y)$ in both forward and reverse KL formulation involves solving a separate optimization problem for every new measurement y. The reason is that $q_\theta \in \mathcal{Q}$ is a function of x alone, not x and y, and we hope that for each y,

$$q_{\theta^\star(y)}(x) \approx p_{X|Y}(x|y).$$

This separate optimization for every y may be inefficient if implemented via standard iterative solvers as shown in Section 6.3.

We could, however, work directly with a family of conditional distributions $q_\theta(x|y)$ which depend on both x and y,

$$\mathcal{Q}_{\text{cond}} = \{(x, y) \mapsto q_\theta(x|y) \ : \ \theta \in \Theta_{\text{cond}}\},$$

where for each y, $q_\theta(\cdot|y) \in \mathcal{P}(\mathcal{X})$ is a probability distribution over $\mathcal{X}$.

We can now compute a conditional variational approximator $q_\theta(x|y)$ by minimizing the *average* KL divergence over all measurements y: this procedure is known as *amortized inference*. It leads to the following optimization problem:

$$\begin{aligned} \theta^\star &= \operatorname*{argmin}_{\theta} \ \mathbb{E}_{Y\sim p_Y} \mathrm{KL}(p_{X|Y}(\,\cdot\,|Y) \| q_\theta(\,\cdot\,|Y)) \\ &= \operatorname*{argmax}_{\theta} \ \mathbb{E}_{X,Y\sim p_{X,Y}} \log q_\theta(X|Y). \end{aligned} \tag{7.3}$$

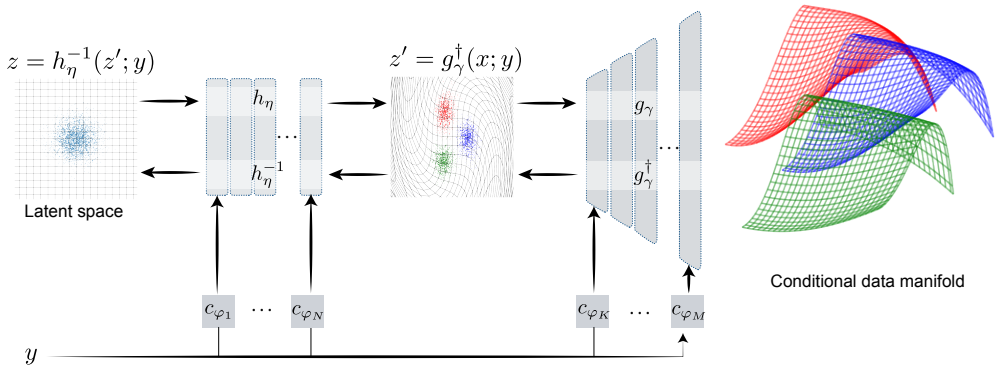


Figure 7.2: Conditional injective flows. Different measurements y_1, y_2 and y_3 give different manifolds.

The key observation is that the population expectation over $p_{X,Y}$ can now be approximated by an empirical expectation over the training data $\{(x^{(i)}, y^{(i)})\}_{i=1}^N$.

The question that remains is: how to parameterize $q_\theta(x|y)$ so that we can 1) learn $\theta^\star$ efficiently from data, 2) easily obtain conditional samples from $q_{\theta^\star}(x|y) \approx p_{X|Y}(x|y)$ for a given y, and 3) efficiently compute standard point estimators such as the MAP estimator? We answer this question in the remainder of the chapter by describing conditional injective flows called C-Trumpets.

In a nutshell, we will define a family of neural networks $f_\theta(Z; y)$ where y is the conditioning input. The first argument, Z, will be a standard Gaussian random vector over a low-dimensional latent space, and the parameter θ adjusted so that for each y, the random vector $f_\theta(Z; y)$ is close in distribution to $X|Y{=}y$. In other words, denoting the standard Gaussian distribution by p_Z, we will obtain $q_\theta(\cdot \mid y)$ as a pushforward of p_Z via $f_\theta(z; y)$,[2]

$$q_\theta(\cdot \mid y) = [f_\theta(\,\cdot\,; y)]_{\#}\, p_Z. \tag{7.4}$$

The approximate posterior samples can then be obtained in a standard way as $f_\theta(Z; y)$.

7.3 C-Trumpets: conditional injective flows

For many natural and medical image classes, the posterior distribution $p_{X|Y}$ is low-dimensional as its support is a subset of the support of the prior distribution p_X which is assumed to concentrate close to a low-dimensional manifold [285, 286]. C-Trumpets (Figure 7.2) are conditional injective normalizing flows that map a low-dimensional latent space to a high dimensional data space using an injective transformation for each conditioning sample. Injectivity guarantees that for each conditioning sample the range

[2]For simplicity we lightly abuse notation by identifying a probability measure and its density.

of the network is a manifold. Moreover, the efficiently-invertible layers facilitate training and inference. As shown in Figure 7.2, the proposed model has two subnetworks: an injective generator g_γ that maps a low-dimensional space $\mathcal{Z} = \mathbb{R}^d$ to the data space $\mathcal{X} = \mathbb{R}^D$, $d \ll D$, and a bijective mapping $h_\eta : \mathcal{Z} \to \mathcal{Z}$ maintaining dimensionality. The end-to-end mapping is then given as,

$$f_\theta(z; y) = g_\gamma(h_\eta(z; y); y)$$

with learnable parameters $\theta = (\gamma, \eta)$.

C-Trumpets are inspired by non-conditional injective flows [21] Trumpets, proposed in Chapter 5. They comprise a sequence of injective [21] and bijective [70] revnet blocks. Each revnet block consists of three components as explained in Section 5.1.1: activation normalization, (injective or bijective) 1×1 convolution and affine coupling layer. It is worth noting that all these components are non-conditional.

To make the generative process conditional, we adapt the conditional affine coupling layers proposed in [260]. Conditional affine coupling layers keep the advantages of a regular flow model—fast inverses and tractable Jacobian computations, while benefiting from the conditioning framework. Since scale $s(\cdot)$ and shift $b(\cdot)$ networks in (5.3) are never inverted, we can concatenate the features of the conditioning sample y to their input without losing invertibility and tractable $\log\det$ Jacobian computation. Accordingly, $s(\cdot)$ and $b(\cdot)$ are replaced by $s(\cdot, c_\varphi(y))$ and $b(\cdot, c_\varphi(y))$,

$$\begin{aligned}
&\text{FORWARD:} && x_1 = z_1 \\
&&& x_2 = s(z_1, c_\varphi(y)) \circ z_2 + b(z_2, c_\varphi(y)) \\
&\text{INVERSE:} && z_1 = x_1 \\
&&& z_2 = s(x_1, c_\varphi(y))^{-1} \circ (x_2 - b(x_1, c_\varphi(y))),
\end{aligned}$$

where $z = [z_1, z_2]^T$, $x = [x_1, x_2]^T$ and $c_\varphi(\cdot)$ represents the conditioning network that extracts appropriate features from y. We deploy conditional affine coupling layers in both injective and bijective subnetworks of C-Trumpets.

7.3.1 Conditioning network

The role of the conditioning network $c_\varphi(\cdot)$ is to extract features from conditioning samples y to be used by the affine coupling layers. The architecture of the conditioning network depends on the nature of the conditioning samples. When y is structured as an image, we use convolutional layers; when it is an unstructured 1D vector, we use fully connected layers, as for example in class-based image generation (see Appendix 7.7.4) where the conditioning data are one-hot class encodings. The output dimension of the conditioning network is set to match the input dimension of the scale and shift modules of the coupling layers. The weights of the conditioning networks are trained jointly with the remaining parameters in C-Trumpets via back-propagation using paired training data.

Further details about conditioning networks in all numerical experiments are given in the Appendix 7.7.2 and 7.7.4.

7.3.2 Skip connections

The fact that we use expanding revnet layers allows us to augment the architecture with elements that are not compatible with the standard injective layers. The conditioning samples often have a pixel-wise dependency on target signals. For example, in image inpaiting with a fixed mask location, the out-of-mask pixels should be simply forwarded to the output. In order to not re-learn these features, we introduce skip connections after every revnet block between the measurements, y and the different resolution levels of the injective subnetwork of C-Trumpets,

$$\begin{aligned} &\text{FORWARD:} \quad x = (\mathbf{1} - S) \circ \text{rev}(z; y) + S \circ \text{resize}(y) \\ &\text{INVERSE:} \quad z = \text{rev}^{-1}\left(\frac{x - S \circ \text{resize}(y)}{\mathbf{1} - S}; y\right), \end{aligned} \tag{7.5}$$

where $\text{rev}(\,\cdot\,)$ is the conditional revnet block, S is a learnable matrix with entries between 0 and 1; S adjusts the amount of the direct mixing of the measurements in the generated posterior sample. We empirically find that skip connections help the injective part of C-Trumpets converge faster and they yield better reconstruction in several imaging problems.

7.3.3 Fast MAP estimation via fixed volume-change layers

As shown in Section 6.2, MAP estimation often requires an iterative solution of a maximization problem. Deep neural networks for image reconstruction are traditionally trained with an ℓ^p loss ($p = 2$ giving the ubiquitous MSE loss and ultimately an approximation of the MMSE estimator $\mathbb{E}[X|Y]$). There are, however, few attempts at using deep neural networks to approximate the MAP estimate in imaging, possibly because the associated "loss" would be a tempered distribution $\delta(x - x')$ [263]. While the MAP and the MMSE estimates coincide when X and Y are jointly Gaussian, they are in general different. For posteriors supported on a low-dimensional manifold, the MMSE estimate will generally not lie on the manifold. (We show a qualitative manifestation of this effect in Section 7.5.4)

A notable deep-learning approach to amortized MAP inference for image super-resolution has been proposed by Sønderby et al. [264]. However, their method is only applicable for super-resolution in the ideal noise-free scenario. In this section, we propose a new variant of affine coupling layers, which enables us to obtain the MAP estimate instantaneously with a single forward pass of the trained network. This method is exact

when used with bijective flows and approximate for injective flows where it computes the MAP estimate of the pre-image samples z' in Figure 7.2.

Consider a trained conditional normalizing flow model $x = f(z; y)$. The MAP estimate can be obtained by solving

$$\begin{aligned} x_{\text{MAP}} &= \underset{x}{\text{argmax}} \ \log(p_{X|Y}(x|y)) \\ &= \underset{x}{\text{argmax}} \ \log(p_Z(z)) - \sum_{k=1}^{K} \log|\det J_{f^k}|, \end{aligned} \tag{7.6}$$

where $z = f^{-1}(x; y)$. In principle, we can run an iterative maximization process to compute x_{MAP} like what we did in Section 6.2. In general, this may be slow and it is not guaranteed to converge, even with multiple random restarts.

Let us analyze (7.6) more closely. While the first term, $\log(p_Z(z))$, has the highest value at $z = \mu_z$ (the mean of the Gaussian), the second term has three components: activation normalization, 1×1 convolution, and the conditional affine coupling layer. The first two components are linear layers with data-independent Jacobians; their $\log\det$s are thus constant with respect to x and can be omitted from (7.6). The $\log\det$ of the Jacobian of conditional affine coupling layers is

$$\log(|\det J(z)|) = \sum_{i=1}^{l} \log(s^i(z_1, c_\varphi(y))), \tag{7.7}$$

where $s^i(\cdot)$ is the ith element of the output of the scale network. This term is in general data-dependent. In order to make it data-independent, we propose to use the following activation for the scale network,

$$s_{\text{FVC}}(z, c_\varphi(y)) = \exp(\text{softmax}(m(z, c_\varphi(y)))), \tag{7.8}$$

where $m(\cdot)$ is an arbitrary neural network and $\text{softmax}(x)$ is defined as

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{l} e^{x_j}} \quad \text{for} \quad i = 1, 2, ..., l.$$

Then we have,

$$\log(|\det J_{\text{FVC}}(z)|) = \sum_{i=1}^{l} \log(s^i_{\text{FVC}}(z_1, c_\varphi(y))) = 1. \tag{7.9}$$

The newly proposed layer has a data-independent $\log\det$ Jacobian, without losing expressivity, as verified empirically in Sections 7.5.2 and 7.5.3.

Now all terms in (7.6) are independent of x, so that

$$\begin{aligned} x_{\text{MAP}} &= \underset{x}{\text{argmax}} \ \log(p_{X|Y}(x|y)) \\ &= \underset{x}{\text{argmax}} \ \log(p_Z(z)) \\ &= f(\mu_z; y); y), \end{aligned} \tag{7.10}$$

which is to say that the MAP estimate is obtained simply by feeding the mean of the Gaussian base distribution into the (bijective) flow.

While the proposed technique is exact for bijective conditional flows, however the $\log\det$ term for conditional *injective* flows is given as

$$\log p_{X|Y}(x|y) = \log p_Z(f_\theta^\dagger(x;y)) - \frac{1}{2}\log|\det[J_{f_\theta}(f_\theta^\dagger(x;y))^T J_{f_\theta}(f_\theta^\dagger(x;y))]|, \quad (7.11)$$

and equation (7.11) cannot be decomposed into a sum of the $\log\det$s of its constituent components. We can nevertheless use this technique to obtain a MAP estimate in the intermediate z'-space as in Figure 7.2. Therefore, we obtain a surrogate of the end-to-end MAP estimate and call it surrogate MAP and denote it as $g^\dagger$-MAP.

7.3.4 Training strategy

Thanks to simple, tractable inverses and $\log\det$s of layer Jacobians, the parameters of normalizing flows can be fitted via maximum likelihood (ML). However, as C-Trumpets have a low-dimensional latent space, the likelihood is only defined in the range of the injective generator. Since the weights of the networks are initialized randomly, prior to training this range does not contain the targets in the training set and their end-to-end likelihoods are not defined. We thus split the training of C-Trumpets, $f_\theta(z;y) = g_\gamma(h_\eta(z;y);y)$, into two phases, following the method of Brehmer and Cranmer [161] for non-conditional injective flows as discussed in Section 5.1.3: (1) The MSE training phase where we only fit the trainable parameters γ of the injective network with the goal of adjusting the range of f_θ to contain the training data, and (2) The ML training phase where we optimize the trainable parameters η of the low-dimensional bijective part, maximizing the likelihood of the pre-image (through g_γ) of the training data.[3]

In the first phase, we optimize γ by minimizing

$$\mathcal{L}_{\text{MSE}}(\gamma) = \frac{1}{N}\sum_{i=1}^{N} \|x^{(i)} - g_\gamma(g_\gamma^\dagger(x^{(i)};y^{(i)});y^{(i)})\|_2^2, \quad (7.12)$$

where $\{(x^{(i)},y^{(i)})\}_{i=1}^N$ are the training samples and $g^\dagger$ is the layer-wise inverse of the injective subnetwork; therefore, the projection operator on the range of $g_\gamma(\,\cdot\,;y)$ is given as $P_{g_\gamma}(x;y) := g_\gamma(g_\gamma^\dagger(x;y);y)$. After training the injective network for a fixed number of epochs, the range of the network approximately contains the training data.

We now switch to the second phase: maximizing the likelihood of the projected training samples in the intermediate space (z' in Figure 7.2), $\{g_\gamma^\dagger(x^{(i)})\}_{i=1}^N$, by minimizing

[3] For simplicity we denote all trainable parameters of the injective network, *including the weights of the conditioning networks and the skip connections* by γ, and all trainable parameters of the bijective network, *including the weights of the conditioning networks* by η.

the following KL divergence over η (cf. Section 7.2):

$$\mathcal{L}_{\text{ML}}(\eta) = \frac{1}{N}\sum_{i=1}^{N}\left(-\log p_Z(z^{(i)}) + \log|\det J_{h_\eta}|\right), \tag{7.13}$$

where $z^{(i)} = h_\eta^{-1}(g_\gamma^\dagger(x^{(i)}; y^{(i)}); y^{(i)})$ and $p_Z(z)$ is a standard Gaussian distribution in $\mathbb{R}^d$. In summary, this training strategy first ensures that the range of the injective generator "interpolates" the training data and then maximizes the intermediate space likelihoods as proxies to the image-space likelihoods. After training, sampling an approximate posterior sample for a given y is performed by sampling a z from a normal distribution and using the forward pass: $x_{\text{gen}} = g_\gamma(h_\eta(z; y); y)$.

7.4 The C-Trumpets signal model

In this section, we briefly discuss the geometric and topological aspects of C-Trumpets. Readers who care mostly about the practical aspects and numerical results may safely skip ahead to Section 7.5.

The primary motivation for introducing (non-conditional) injective flows is to model data supported on low-dimensional manifolds [21, 161, 193, 279]. It was empirically shown that for common structured datasets these models are indeed simpler, faster to train and generate higher quality samples than globally invertible normalizing flows which maintain the same dimension across all layers.

On the other hand, we introduce C-Trumpets from the point of view of uncertainty quantification and posterior sampling rather than geometrical and topological considerations about the class of signals it models. It is nevertheless important to make this class explicit, since this understanding will guide design choices and circumscribe the range of problems in which C-Trumpets are the right tool of choice.

For every (fixed) conditioning sample y, a C-Trumpet becomes a conditional flow modeling the corresponding conditional distribution. Thus for every conditioning sample y the range is a low-dimensional manifold

$$\mathcal{M}_y = \{f_\theta(z; y) \ : \ z \in \mathcal{Z}\}$$

with topology induced by the topology of the latent space (the support of p_Z). The range of a C-Trumpet as a function of z and y then corresponds to the union of the family of these manifolds indexed by y,

$$\mathcal{R}_\theta = \{f_\theta(z; y) \ : \ z \in \mathcal{Z}, y \in \mathcal{Y}\} = \bigcup_{y\in\mathcal{Y}} \mathcal{M}_y.$$

For a given (y, z) let

$$\beta(y) = \{x \ : \ x = f_\theta(z; y), \ z \in \mathcal{Z}\},$$

denote the set of all possible signals x that could have caused the observation y, or in other words the support of the distribution $p_{X|Y=y}$.

Let us now assume a slightly stronger condition than injectivity of f_θ in the first argument: that the Jacobian (with respect to both z *and* y) of f_θ has full rank $\dim(\mathcal{Z}) + \dim(\mathcal{Y})$. Note that this does not guarantee global injectivity of the map $(z, y) \mapsto f_\theta(z; y)$, but it does imply that there is a (finite) collection of open sets $\{U_i\}$ (thought of as neighborhoods in $\mathcal{Y}$) which cover $\mathcal{Y}$, $\bigcup_i U_i = \mathcal{Y}$, such that f_θ is injective on $\mathcal{Z} \times U_i$ for all i and in fact a homeomorphism between $\mathcal{Z} \times U_i$ and $\beta(U_i)$. The signal model we just described is called a *fiber bundle* [265, 287].

Formally, a generic fiber bundle comprises sets $(\mathcal{E}, \mathcal{B}, \mathcal{Z})$ called the total space $(\mathcal{E})$, the base space $(\mathcal{B})$, and the (typical) fiber $(\mathcal{Z})$, together with a surjective map $\pi : \mathcal{E} \to \mathcal{B}$, with the property that for each $b \in \mathcal{B}$, there exists an open neighborhood $b \in U_i \subseteq \mathcal{B}$ on which $\phi : \mathcal{Z} \times U_i \to \pi^{-1}(U_i)$ is a homeomorphism. The definition can be illustrated by a commutative diagram [265],

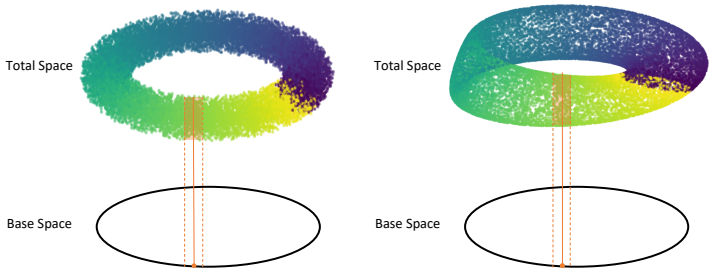
$$\begin{array}{ccc} \pi^{-1}(U_i) & \underset{\phi^{-1}}{\overset{\phi}{\leftrightarrows}} & \mathcal{Z} \times U_i \\ {\scriptstyle\pi}\downarrow & \swarrow_{\text{proj}} & \\ U_i & & \end{array} \tag{7.14}$$

where proj : $\mathcal{Z} \times U_i \to U_i$. We have the following correspondence between the standard fiber bundle notation and the notation specific to C-Trumpets used in this chapter:

C-Trumpets		**Fiber bundles**
$\mathcal{R}_\theta$	$\longleftrightarrow$	$\mathcal{E}$
$\mathcal{Z}$	$\longleftrightarrow$	$\mathcal{Z}$
$\mathcal{Y}$	$\longleftrightarrow$	$\mathcal{B}$
β	$\longleftrightarrow$	π^{-1}
f_θ	$\longleftrightarrow$	ϕ.

Fiber bundles model a variety of situations in imaging, scientific computing, and physics. In CryoEM we can see the 3D rotation group as a bundle of circles over the 2-sphere, which leads to denoising algorithms [288, 289]. Symmetries for the forward operator (for example in phase retrieval) naturally split the data space into equivalence classes that can be modeled as fiber bundles. More generally, any many-to-one machine learning task is well-modeled by a fiber bundle [265]. We also mention natural connections with gauge equivariance and geometric deep learning [81].

That said, our aim here is to identify a natural assumption we make when modeling data with a conditional generative model, analogously to how unconditional generative models are naturally compatible with data that lives close to a low-dimensional manifold. Continuing the analogy, just like manifolds are used to regularize ill-posed inverse



(a) Torus (b) Elliptic Möbius

Figure 7.3: Two examples of fiber bundles: Torus and elliptic Möbius as fiber bundles over the base circle.

problems, we imagine fiber bundle models can be used as regularizers with additional structure even when the data only approximately lives on a fiber bundle.

Fiber bundles model spaces that are locally product spaces but may have non-trivial topology globally. An example of a space that is globally a product space is the cylinder $\mathcal{E} = S^1 \times [0, 1]$ with the base space being the circle $\mathcal{B} = S^1$ and all fibers being translates of the line segment $\mathcal{Z} = [0, 1]$. The projection π associates to each point $x \in \mathcal{E}$ its position along the base circle; π^{-1} takes a position along the base circle and returns the corresponding line segment. The same $\mathcal{B}$ and $\mathcal{Z}$ can generate a rather different space—the Möbius band—which is globally not a product space.[4]

Thus (under appropriate conditions) the range of a C-Trumpet is locally homeomorphic to a product space (it is a product space up to a stretch), even though it globally need not be depending on the topologies of $\mathcal{Z}$ and $\mathcal{Y}$. The requirement that ϕ be a homeomorphism on U_i implies that (locally) the fibers do not intersect. This can of course only hold if $\dim(\mathcal{Y}) + \dim(\mathcal{Z}) \leq \dim(\mathcal{X})$ and the appropriate conditions on the Jacobian are satisfied, but in any case it gives a useful intuition for the kind of problems and datasets that admit modeling by C-Trumpets.

We show that C-Trumpets can indeed model simple low-dimensional fiber bundles: an embedded solid[5] torus in Figure 7.3a and a solid Möbius band with elliptic cross-sections (fibers) in Figure 7.3b. As shown in Figure 7.3, the conditioning samples for both the torus and the elliptic Möbius are taken to live on the base circle ($y \in [0, 2\pi)$). For each angle y, C-Trumpets generate samples from a distribution on the disk for the torus and an elliptical disk for the elliptic Möbius band. Accordingly, we train C-Trumpets with latent dimension two. Figure 7.4 demonstrates the generated samples by C-Trumpets; we sample the resulting models in increments of 6° to make the fiber bundle structure clear.

[4]To explain this mathematically we would need to introduce the transition maps and the fundamental group, which is beyond the scope of our sketch.

[5]By a "solid" torus we refer to a bundle whose fibers are disks. Modeling a standard torus is challenging with coupling layers which can only be defined starting in dimension 2. Even in this case their expressivity is limited so this low-dimensional example is only meant as illustration.

(a) Torus (b) Elliptic Möbius

Figure 7.4: Samples on fiber bundles generated by a C-Trumpet; in both (a) and (b) the base manifold S^1 is sampled every $6°$.

7.5 Experiments

We start by showcasing how C-Trumpets provide MMSE, MAP and uncertainty estimates in a variety of imaging inverse problems. The MMSE estimate, $\mathbb{E}[X|Y = y]$, is approximated by averaging a fixed number of posterior samples from a C-Trumpet fitted to the training data. The MAP estimate can be efficiently approximated using fixed volume-change layers in Section 7.3.3, without iterative maximization over x. Finally, we compute the uncertainty estimate (UQ) through a simple pixel-wise standard deviation as

$$\begin{aligned} \widehat{\text{MMSE}} &= \frac{1}{K}\sum_{k=1}^{K} f_\theta(z_k; y), \\ \text{UQ} &= \sqrt{\frac{1}{K}\sum_{k=1}^{K} |f_\theta(z_k; y) - \widehat{\text{MMSE}}|^2}, \end{aligned} \tag{7.15}$$

where $|\cdot|^2$ is applied to each pixel. In all experiments, we use $K = 25$ posterior samples to approximate the MMSE and UQ estimates (we find that the quality of MMSE and UQ saturate for a higher number of posterior samples).

We experiment with limited-view computed tomography (CT), nonlinear electromagnetic inverse scattering [9], and (linearized) seismic travel-time tomography [290]. Additional evaluations on popular compressed-sensing-style benchmarks—denoising, inpainting, super-resolution and random masking—are given in Section 7.5.3.

We compare C-Trumpets with two different types of conditional bijective flows, C-INN [260] and C-Glow [277]. C-INN used conditional coupling layers for the first time, while C-Glow has a rather different architecture by using two parallel bijective flows for simultaneously modeling the target and conditioning samples. In order to emphasize the importance of an expansive injective model over a bijective one, we build a comparison baseline model, C-Rev, which consists of the bijective portion of C-Trumpets with latent-space dimension equaling that of the image data, i.e., $h_\eta(z, y)$. Finally, in Section 7.5.4, we visually compare the MMSE and MAP estimates for several ill-posed inverse problems.

7.5.1 Gaussian random fields

We choose realizations of Gaussian random fields (GRF) as instances of $p_X(x)$ where we have access to exact posterior distributions. We compare the quality of the posterior approximated by C-Trumpets with the true posterior obtained by analytical solution. The closed-form posterior distribution $p_{X|Y}(x|y)$ where $y = Ax + n$, $x \sim \mathcal{N}(\mu_x, \Sigma_x)$ and $n \sim \mathcal{N}(0, \lambda^2 I)$ as follows,

$$\begin{aligned} p_{X|Y}(x|y) &= \mathcal{N}(\mu_{x|y}, \Sigma_{x|y}), \\ \mu_{x|y} &= \mu_x + \Sigma_x A^T (A\Sigma_x A^T + \lambda^2 I)^{-1}(y - A\mu_x), \\ \Sigma_{x|y} &= \Sigma_x - \Sigma_x A^T (A\Sigma_x A^T + \lambda^2 I)^{-1} A\Sigma_x. \end{aligned} \tag{7.16}$$

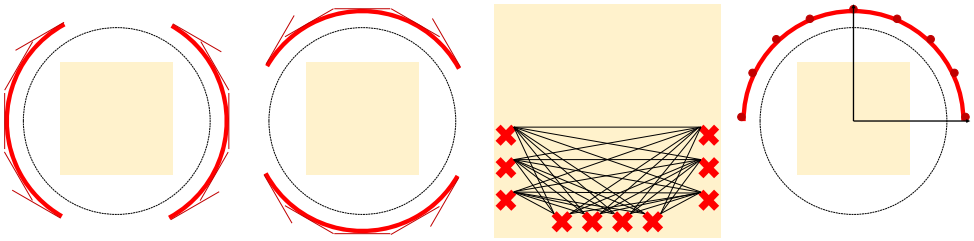
We consider GRFs in resolution 64×64, $\lambda = 5 \times 10^{-3}$, and let A be the mask operator that replaces a 32×32 patch at the center of the image with zeros. We train C-Trumpets over 60000 training samples. In Figure 7.5 (second to fifth columns), we show the MMSE estimate followed by three posterior samples, and finally the UQ obtained from both the analytical solution (7.16) and C-Trumpets. MMSE and UQ are computed over 500 random posterior samples. We compare the MMSE estimates obtained from C-Trumpet and the analytical solution (7.16) against the ground truth. In Table 7.1 we find that C-Trumpet performs similar to the analytical solution in both SNR and SSIM metrics. Since the analytical posterior distribution is Gaussian, MAP and MMSE estimators are equal. We find that the MAP and MMSE estimates obtained from C-Trumpet are quite close: an SNR of 44.43dB hinting at the efficacy of our way of computing MAP estimates.

Table 7.1: Performance comparison of MMSE estimates between C-Trumpets and the analytical solution obtained from 7.16; C-Trumpets presents MMSE estimates close to the analytical solution, which shows the effectivity of the proposed method.

	SNR (dB)	SSIM
C-Trumpets	19.71	0.89
Analytical solution	**20.22**	**0.91**

7.5.2 Computational imaging

Limited-view CT In limited-view CT (cf. cryo-electron tomography [145] and dental CT [291]), a contiguous cone of angles is missing from the acquisition; Figures 7.6a and 7.6b, illustrate vertical and horizontal missing cones of $60°$. As there are no measurements in a vertical (horizontal) missing cone, we should expect higher uncertainty in horizontal (vertical) components. We use the filtered back projection (FBP) reconstruction as our measurements y, and train on 40000 256×256 samples from the LoDoPaB-CT [137] dataset. The measurement SNR is set to 40 dB. In Figure 7.7, we show posterior samples,



(a) CT vertical (b) CT horizontal (c) Travel time (d) Scattering

Figure 7.6: Sensing geometry for the various imaging problems explored in Section 7.5. The red regions show the locations of the sensors. In the case of CT ((a) and (b)), the line segments indicate that "sensors" measure entire projection images as opposed to complex scalars in nonlinear scattering (d).

$g^\dagger$-MAP, MMSE, and uncertainty quantification (UQ) estimates from C-Trumpets. The real-space UQ estimate shows higher uncertainty in the vertical (horizontal) components where we have horizontal (vertical) missing cone. This is consistent with our expectations from the physics of CT.

We further show the UQ estimate in the Fourier domain computed by averaging over the individual DFT bins. Quite pleasingly, this estimate aligns perfectly with the theoretical prediction from the Fourier slice theorem; higher uncertainty (bright regions) are indeed inside the missing cone while it is worth emphasizing that C-Trumpets are not specifically designed for the forward operator of the CT problem. Moreover, as expected, there is higher uncertainty in higher frequency components (see Figure 7.15 in the appendix). We emphasize that C-Trumpets are the only conditional generative architecture that gives such physically meaningful posterior samples.

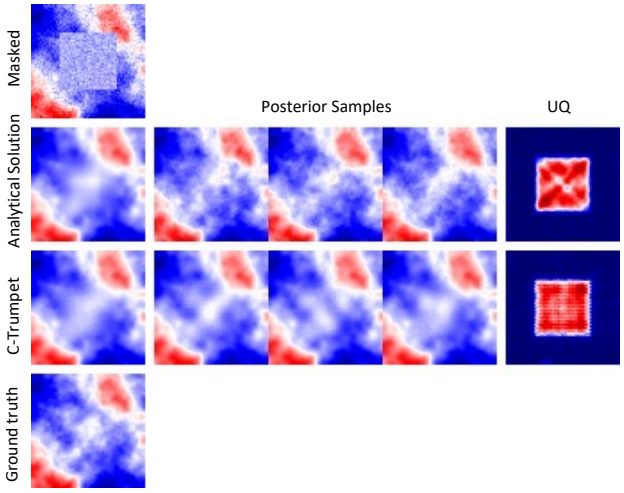
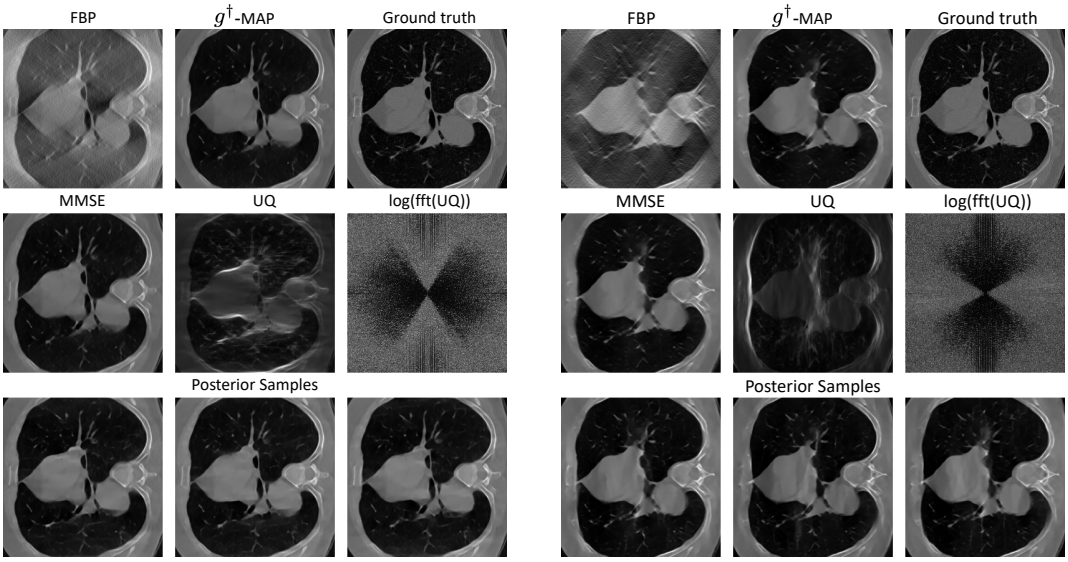


Figure 7.5: Performance comparison between C-Trumpet and analytical solution (true posterior) (7.16) in mask problem where the target signals come from a Gaussian distribution; C-Trumpets present MMSE and UQ close to the analytical solution.

It is also worth mentioning that C-Trumpets model this 256×256 resolution dataset with only 10M trainable parameters in less than 24 hours of training time on a single NVIDIA V100 GPU. Due to memory constraints, we could simply not train the baseline bijective models C-INN, C-Glow, and C-Rev on 256×256 images. The



(a) Vertical missing cone (60° to 120°) (b) Horizontal missing cone (-30° to 30°)

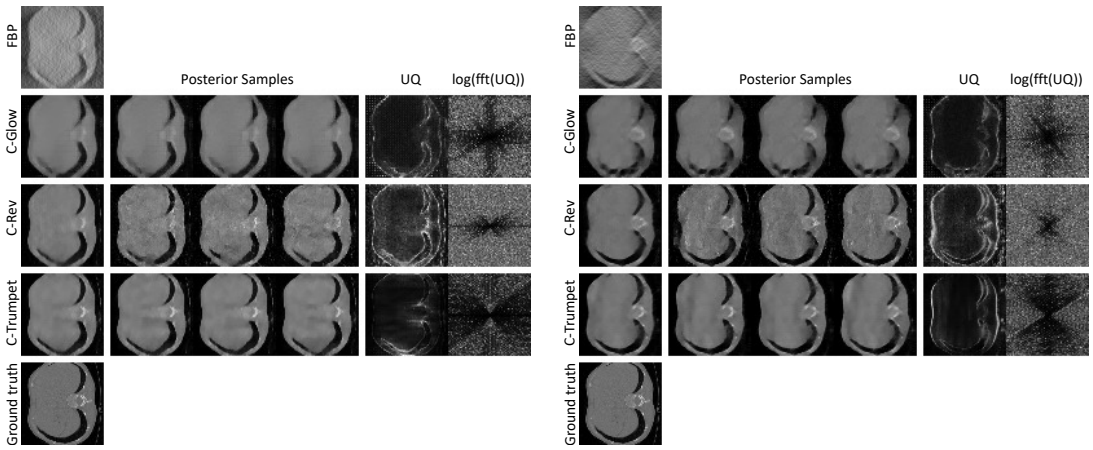
Figure 7.7: Limited-view CT in resolution 256×256. The frequency-domain uncertainty estimate shows the log-scale standard deviation of DFT-bins. Alignment with the theoretical prediction from the Fourier slice theorem signifies physically-meaningful uncertainty quantification; higher uncertainty (bright regions) inside the missing cone.

large latent space dimension of these models leads to very large memory footprints.

We can still compare our uncertainty estimates with these models at a lower resolution of 64×64. Figure 7.8 shows such an experiment at the SNR of 25dB. In the first panel (second to fifth row), we show MMSE estimates of different flow models and three random posterior samples. We further provide UQ estimates in real and Fourier space. As we can see, not only do C-Trumpets outperform the conditional bijective flows in terms of reconstruction quality (MMSE estimate) (see also Table 7.2) but they also give a more meaningful uncertainty estimate even in high noise. Bijective models generate significant uncertainty outside the missing cone, indicating that they do not learn a reconstruction map consistent with the physics of CT.

Electromagnetic inverse scattering We consider the non-linear electromagnetic inverse scattering reviewed in Section 6.1. In the first experiment, we use 36 incident plane waves and 36 receivers, distributed uniformly around the object with maximum permittivity of $\epsilon_r = 6$ and dimension 20 cm $\times$ 20 cm. We work at the frequency of 3 GHz. and simulate the measurements by solving the Helmholtz equation explained in Section 6.1. We add noise to the measurements for a target measurement SNR of 30 dB. We build a dataset of 64×64 images of overlapping ellipses with 60000 samples.

Figure 7.9 shows the performance of C-Trumpets where the scattered fields are used



(a) Vertical missing cone (60° to 120°) (b) Horizontal missing cone (-30° to 30°)

Figure 7.8: Performance comparison in limited-view CT with resolution 64×64. The frequency-domain uncertainty estimate shows the log-scale standard deviation of DFT-bins. The estimated $\log(\text{fft}(\text{UQ}))$ by C-Trumpets aligns with the theoretical prediction from the Fourier slice theorem; higher uncertainty (bright regions) inside the missing cone.

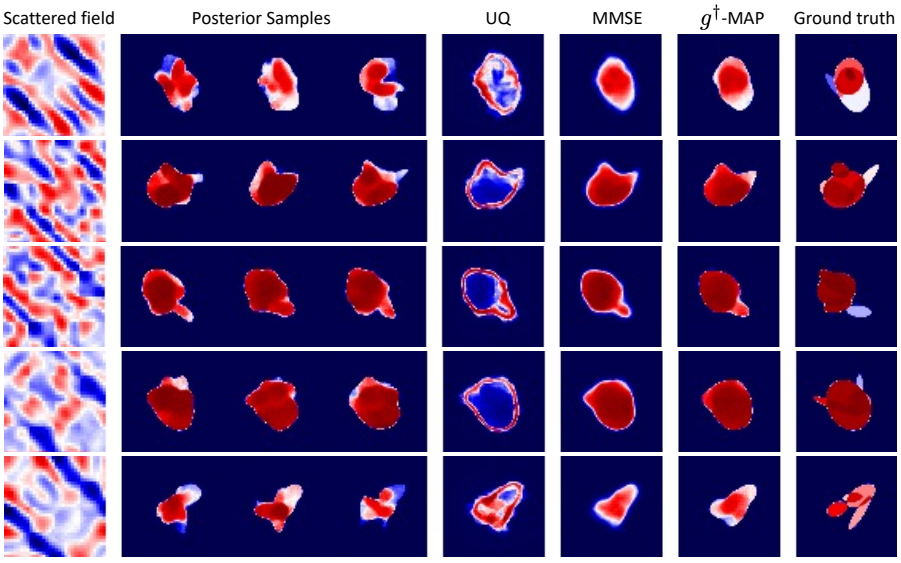


Figure 7.9: Performance of C-Trumpets on nonlinear inverse scattering with $\epsilon_r = 6$ with a full-view sensing geometry. Conditioning inputs are scattered fields.

as conditioning samples. This experiment clearly shows that C-Trumpets can generate meaningful posterior samples, even for a highly non-linear ($\epsilon_r = 6$) and ill-posed problem. Additional experiments with different contrasts ϵ_r and conditioning schemes (scattered fields vs back projections) are illustrated in Figures 7.16, 7.17, 7.18 and 7.19 in the appendix.

In order to assess the performance of different conditional flows in uncertainty quantification, we design another experiment with 180 incident plane waves and 180 receivers only on the *top* side of the object (7.6d). In this setup, we expect higher uncertainty in the lower part of the object.

Figure 7.11 illustrates the performance of C-Glow, C-Rev and C-Trumpets for $\epsilon_r = 6$. We use a simple back projection (BP) [292] as conditioning measurements y. In the first panel (from the second row to the fifth row), we show MMSE estimates of different flow models: C-Trumpets perform better than bijective flows (see also Table 7.2). As expected, however, the reconstructions are not as good as in the previous experiment since we only have measurements on one side of the domain. We see that C-Trumpets again provide meaningful uncertainty estimates, with more uncertainty in the lower part of the object (red regions in the UQ panel correspond to higher uncertainty).

Seismic travel-time tomography (NS) We work with the linearized seismic tomography operator as described in [290]. Here we are given travel times of a seismic wave between each sensor pair in a network of NS sensors placed on the ground. The travel times are assumed to linearly depend on the "slowness" map which is taken to be an MNIST image [204]. We use NS$= 10$ sensors on the boundary of the lower part of the domain which yields 33 measurements as shown in Figure 7.6c. We use the pseudo-inverse of the measurements y as the conditioning samples and work at the SNR of 40dB. Figure 7.10 compares the performance of C-Trumpets and C-Rev. In the first panel, the second and the third rows show MMSE estimates of C-Rev and C-Trumpets. C-Trumpets outperform C-Rev in both MMSE estimate and posterior sampling (see also Table 7.2). Furthermore, given no sensors (Figure 7.6c) in the top regions of the image (or slowness map), we would expect higher uncertainty there. The UQ column shows that estimates from C-Trumpets assign higher uncertainty to the top half of the domain compared to C-Rev. Additional results are shown in Figure 7.20 in the appendix.

7.5.3 Image restoration

In this section, we compare C-Trumpets with models on various image restoration tasks. Similarly as in computational imaging problems, each task requires training a different model but once trained, the model can be used instantaneously for arbitrary measurements y. We consider four standard restoration tasks: (i) **Image denoising**: We train a model to generate plausible clean images given a noisy image at an SNR of -1 dB; (ii) **Image**

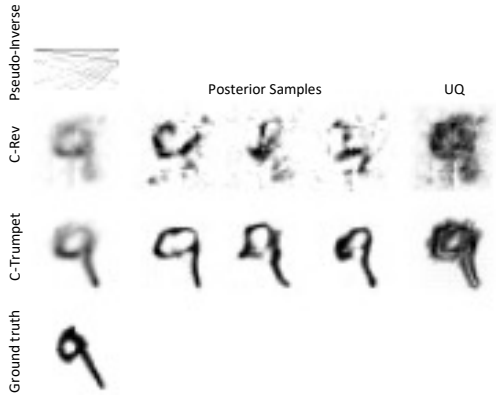


Figure 7.10: Performance comparison in seismic travel-time tomography with 10 sensors. C-Trumpets demonstrate better posterior samples and MMSE estimate, both models assign higher uncertainty in top of the image (black regions) which lacks measurements.

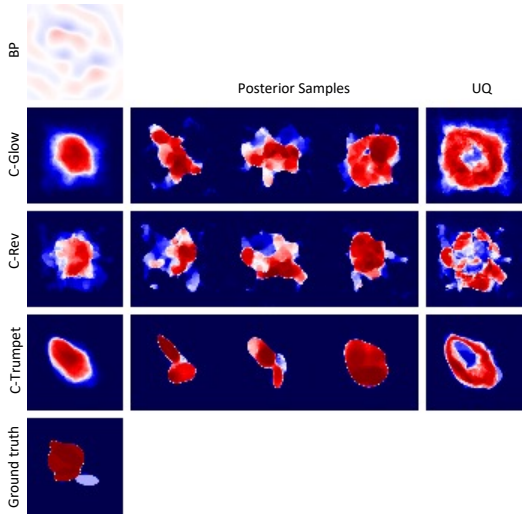


Figure 7.11: Performance comparison in electromagnetic inverse scattering ($\epsilon_r = 6$, top-view, BP conditioned); C-Trumpets demonstrate more meaningful UQ by assigning higher uncertainty in the bottom of the object (red regions), which lacks measurements.

super-resolution (f): Generate high-resolution images given an image downsampled by a factor of f along each axis; (iii) **Random mask** (p): The degradation process replaces every pixel with zero with probability p; and (iv) **Mask** (s): The degradation process replaces an $s \times s$ patch of the image with zeros.

We used the 8-bit RGB CelebA [206] dataset with 80000 64×64 training samples. Figure 7.14 in the appendix compares the performance of flow models on different image restoration tasks. Table 7.2 further gives the SNR and SSIM of the MMSE estimate. C-Trumpets consistently outperform other conditional bijective flows. The difference in the performance of C-Rev and C-Trumpets suggests that the low-dimensional latent space of C-Trumpets acts as an effective regularizer in the restoration mapping, f_θ. C-Trumpets also provide a meaningful uncertainty quantification. For example, although the forward operator is random in the random mask problem, C-Trumpets still capture a meaningful uncertainty estimate by assigning higher uncertainty inside the masked region (see Figure 7.14d).

In order to assess the memory requirements of the different models, we compare the number of trainable parameters used for training over 64×64 RGB images: C-INN: 13M, C-Glow: 14M, C-Rev: 22M and C-Trumpets: **4M**. We did not have sufficient resources to train bijective models over 256×256 images but it is clear that the differences in the memory footprint at that resolution would be further exacerbated.

Table 7.2: Performance of MMSE estimate (computed over 25 posterior samples) of different models on solving inverse problems averaged over 5 test images

(a) SNR (dB)

	C-INN	C-Glow	C-Rev	C-Trumpets
Denoising	15.98	15.67	15.99	**16.86**
Super-resolution ($\times 4$)	18.57	19.24	19.22	**20.70**
RandMask ($p = 0.2$)	**20.84**	19.70	13.31	20.11
Mask ($s = 32$)	18.91	17.86	18.54	**21.01**
Limited-view CT	-	11.63	13.13	**13.58**
Scattering ($\epsilon_r = 6$)	-	1.45	1.19	**4.04**
Travel-time (NS = 10)	-	-	14.83	**18.19**

(b) SSIM

	C-INN	C-Glow	C-Rev	C-Trumpets
Denoising	0.63	0.60	0.66	**0.70**
Super Resolution ($\times 4$)	0.79	0.82	0.79	**0.84**
RandMask ($p = 0.2$)	0.86	0.82	0.57	**0.87**
Mask ($s = 32$)	0.83	0.80	0.84	**0.88**
Limited-view CT	-	0.59	0.71	**0.74**
Scattering ($\epsilon_r = 6$)	-	0.67	0.65	**0.73**
Travel-time (NS = 10)	-	-	0.59	**0.64**

7.5.4 MAP vs MMSE for ill-posed inverse problems

Figure 7.12 demonstrates the MMSE, $g^\dagger$-MAP and a random posterior sample for four types of ill-posed inverse problems. The MMSE estimate is obtained by averaging over 25 posterior samples. Although the MMSE estimate is the optimal reconstruction in terms of the ℓ^2-error, we see from Figure 7.12 that it is often blurry, especially when the true posterior is multi-modal; $g^\dagger$-MAP estimates are sharper. Moreover, as the MMSE estimate is obtained by averaging over the posterior samples, it is not generally on the manifold, while the $g^\dagger$-MAP estimate is always on the manifold.

7.6 Summary

We proposed C-Trumpets, a conditional injective flow model that enables amortized inference with approximate posteriors that live on low-dimensional manifolds. Our proposed model is considerably cheaper to train in terms of memory and compute costs compared to the regular conditional flows. The experiments we performed indicate that C-Trumpets generate better posterior samples and more accurate uncertainty estimates over a variety of ill-posed inverse problems. The proposed fixed-volume-change coupling layers enable us to approximate the sharp MAP estimates instantaneously after training. High computational demands of training bijective flows at high-resolution have thus far impeded their wider adoption in computational imaging workflows. The comparably lightweight memory footprint of C-Trumpets together with physically-consistent UQ makes them an attractive architecture for imaging problems where characterizing uncertainty is paramount.

C-Trumpets have several limitations that warrant discussion. The latent space dimension in C-Trumpets is chosen arbitrarily and it may be quite different from the true dimension of the posterior support. Recent work [293] proposes an injective flow architecture that estimates the dimension of the data manifold. Similar ideas may extend to C-Trumpets but for the moment we rely on rules of thumb rather than principled choices.

Another limitation is that it is not straightforward to estimate the likelihoods of samples generated by C-Trumpets (cf. Section 7.3.3), the reason being that the Jacobian determinant of compositions of maps between spaces of different dimensions cannot be written as a product of Jacobian determinants of the constituent maps. Likelihood estimates can still be obtained by sampling but that is considerably slower than what is possible with bijective flows. Recently, Ross et al. [242] proposed an injective generator which provides access to the exact likelihood of the generated samples, but the constraints they impose on the architecture in order to enable this feat seem to severely limit expressivity. The design of an injective model that is at once expressive, lightweight, and gives fast exact likelihoods remains an open problem. On the theoretical side, further studies are needed to characterize the types of posterior distributions that can be modeled by C-Trumpets, especially with fixed-volume-change layers. The important open question is that of universality of C-Trumpets as models of conditional distributions. Finally, Siahkoohi et al. [294, 295] fully exploit the depth-independent memory complexity of normalizing flows to handle high-dimensional data. This strategy can also be used in C-Trumpets to further improve memory efficiency and apply the model to super high-dimensional imaging problems.

7.7 Appendix

7.7.1 Fast inverses

Inverting the revnet coupling layers requires the matrix inverse of the kernel of the 1×1 convolution layer. As this may be slow, Kingma and Dhariwal [70] proposed to use the LU decomposition to reduce the computational complexity,

$$w = PL(U + \text{diag}(s)), \qquad (7.17)$$

where P is a permutation matrix, L is a lower-triangular matrix with ones on diagonal and s is a vector. Computing the $\log\det$ of the layer Jacobian then simplifies to

$$\log|\det(w)| = \sum_{i=1}^{d} \log(|s_i|). \qquad (7.18)$$

The LU decomposition thus reduces the complexity of computing the $\log\det$ of the Jacobian of the 1×1 convolution layer

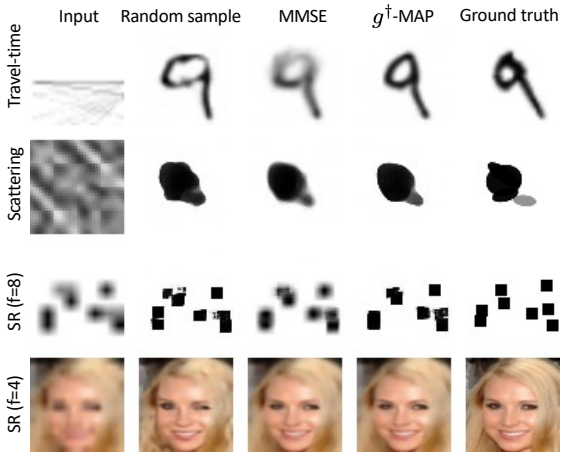


Figure 7.12: MMSE and $g^\dagger$-MAP estimations in different inverse problems; the proposed $g^\dagger$-MAP estimate gives much sharper reconstruction than MMSE.

from $\mathcal{O}(c^3)$ to $\mathcal{O}(c)$ (and one-time factorization cost). This trick, however, was only used in inference (when sampling from the model), but not for training the bijective flows as matrix inversion is not critical in that case. Unlike bijective flows, C-Trumpets do require matrix inversion during the MSE training phase (cf. (7.12)). In order to reduce the computational cost of inversion, we leverage the LU decomposition as follows

$$w^{-1} = (U + \text{diag}(s))^{-1} L^{-1} P^{-1}. \tag{7.19}$$

Since L and U are triangular matrices and P is a fixed rotation, computing the inverses costs $\mathcal{O}(2 \times c)$ instead of $\mathcal{O}(c^3)$. This significantly reduces the training time of the injective part of C-Trumpets, especially in high-dimensional problems.

7.7.2 Experimental details

Fiber bundles

Datasets: We analyzed the performance of C-Trumpets on two fiber bundle datasets:

- Torus

$$\begin{aligned} x &= \cos(t)[R + r\cos(s)] \\ y &= \sin(t)[R + r\cos(s)] \\ z &= r\sin(s) \end{aligned} \tag{7.20}$$

  where $t \in [0, 2\pi)$ and $s \in [0, 2\pi)$. We set $R = 1$ and $r = 0.25$ and generate 18000 training samples.

- Elliptic Möbius

$$\begin{aligned} x &= \cos(t)[R - b\sin(t/2)\sin(s) + a\cos(t/2)cos(s)] \\ y &= \sin(t)[R - b\sin(t/2)\sin(s) + a\cos(t/2)cos(s)] \\ z &= b\cos(t/2)\sin(s) + a\sin(t/2)\cos(s) \end{aligned} \tag{7.21}$$

  where $t \in [0, 2\pi)$ and $s \in [0, 2\pi)$. We set $R = 1$, $a = 0.4$ and $b = 0.1$ and generate 18000 training samples.

Network architecture and training details: The injective part of C-Trumpets maps $\mathbb{R}^2 \to \mathbb{R}^3$ and consists of an expansion layer (a matrix of size $\mathbb{R}^{3\times 2}$) followed by 24 revent blocks without activation normalization layer. The bijective part which maps $\mathbb{R}^2 \to \mathbb{R}^2$ also consists of 32 revent blocks. We use coupling layers proposed in [70] with 3 fully-connected layers for scale and bias. We train for 100 epochs using the Adam optimizer [97] with a learning rate of 10^{-3} for both the injective and the bijective parts of C-Trumpets. We note that this minimal input dimension of the coupling layers has yields poor expressivity, but we only use this example for illustration.

Limited-view CT

We consider the 2D parallel-beam CT problem with a missing cone of sensors.

Network architecture: We describe the architecture of C-Trumpets for 256×256 images. The injective part consists of 6 injective revnet blocks, each increasing the dimension by a factor of 2. Between the injective layers, we intersperse 36 bijective revnet blocks. We choose a latent space of size 2048. The bijective part consists of 48 bijective revnet blocks. We use 3 convolutional layers for the scale and bias networks of fixed-volume-change coupling layers. The conditioning networks have 3 conv layers along with a 'squeeze' layer that helps match the dimension of the conditioning sample to the input of the scale network.

The architecture of C-Trumpets for 64×64 images is similar to the 256×256 variant except that we use 18 (resp. 24) revnet blocks in injective (resp. bijective) subnetworks and choose a 64-dimensional latent space. The C-Rev model used for comparison has 24 revnet blocks, all at the highest resolution (i.e 64×64).

We initialize the weights of the revnet blocks as in [70]. All elements of the skip connection matrix S are initialized to 0.5. All models are trained for 300 epochs (150 per phase) using the Adam optimizer [97] with a learning rate of 10^{-4}.

Linearized travel-time tomography

Linearized travel-time tomography is inspired by seismic travel-time tomography that aims to reconstruct the wavespeed variation inside a planet. Sensors are placed on the ground and we measure arrival times of surface waves. We assume that the receivers and transmitting sensors are co-located. We further assume that the pixel intensities represent the "slowness" (inverse wave speed). We can therefore measure the travel times between a transmitting sensor, s_i and a receiving sensor s_j as

$$t(s_i, s_j) = \int_0^1 f(s_i + \lambda(s_j - s_i))d\lambda, \tag{7.22}$$

where f represents the image. Note that $t(s_i, s_j) = t(s_j, s_i)$.

Network architecture: The injective subnetwork consists of 4 injective revnet blocks, each increasing the dimension by a factor of 2. We use 12 bijective revnet blocks interspersed in between the injective layers. The bijective part takes 64-dimensional latent vectors and consists of 12 bijective revnet blocks. C-Rev used in comparison has 16 revnet blocks all at the 32×32 resolution. The training hyperparameters are same as the CT problem.

(a) MNIST

(b) Voxceleb

Figure 7.13: Class-based image generation

Image restoration tasks

We use the same architecture and training hyperparameters as that of the CT problem except that we work with $64 \times 64 \times 3$ resolution images. The latent dimension of C-Trumpets is chosen to be 192 (64×3).

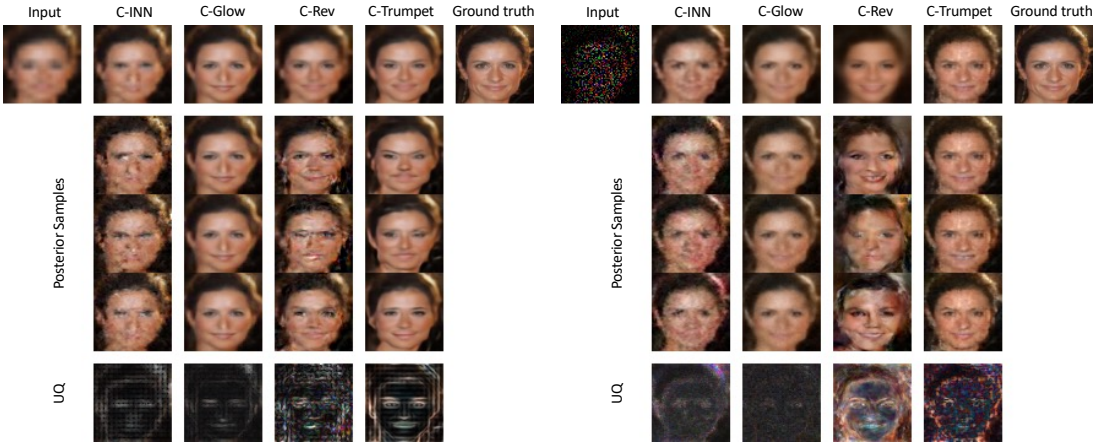
7.7.3 Additional experiments

7.7.4 Class-based image generation

We perform class-based image generation over MNIST digits [204] and a subset of 10 people from the voxceleb [296] face dataset with 5000 64×64 training samples. We use the one-hot class labels as conditioning vectors to generate samples from the given class and use two fully-connected layers followed by a reshaping module for the conditioning network of C-Trumpets. Figure 7.13 shows the class-based generated samples by C-Trumpets. This experiment indicates that our proposed model can generate good quality class-conditioned samples.

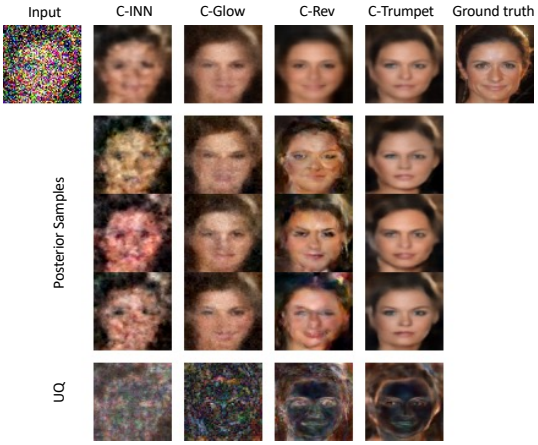
7.7.5 Image restoration and inverse problems

Figure 7.14 compares the performance of conditional normalizing flows in image restoration tasks. Figures 7.15 to 7.24 demonstrate further results on different ill-posed inverse problems. We can observe that C-Trumpets have a significant edge over the baselines.

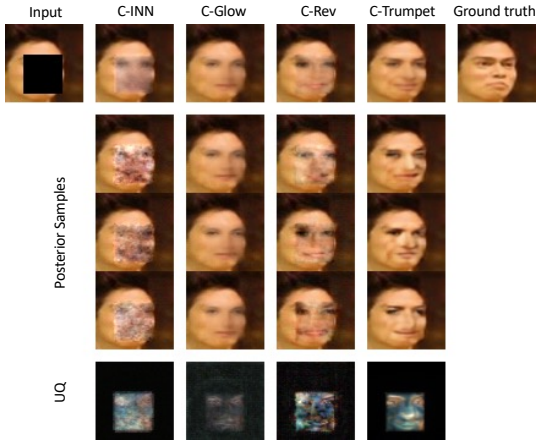


(a) Super-resolution ($f = 4$)

(b) Random mask ($p = 0.2$)

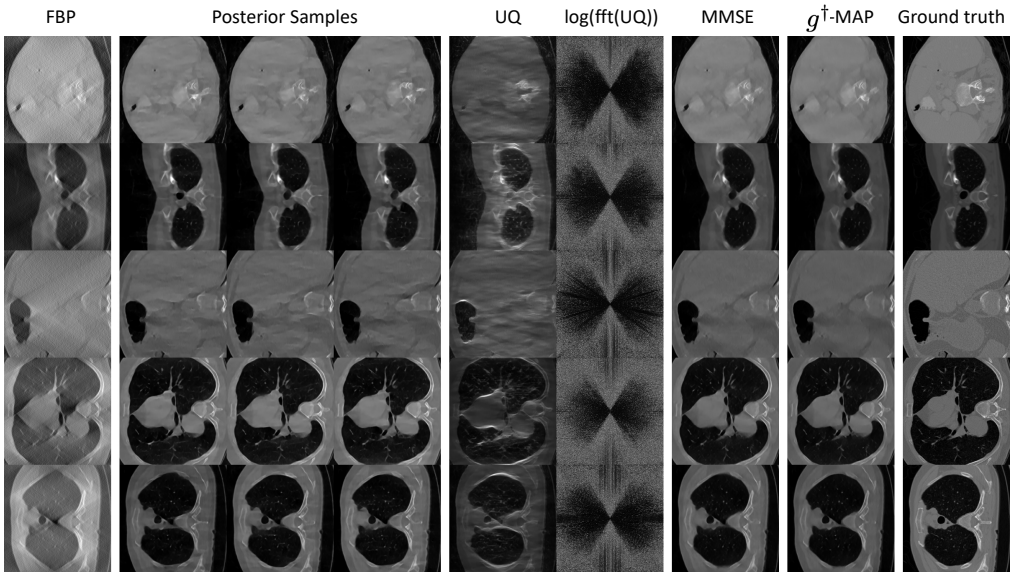


(c) Denoising

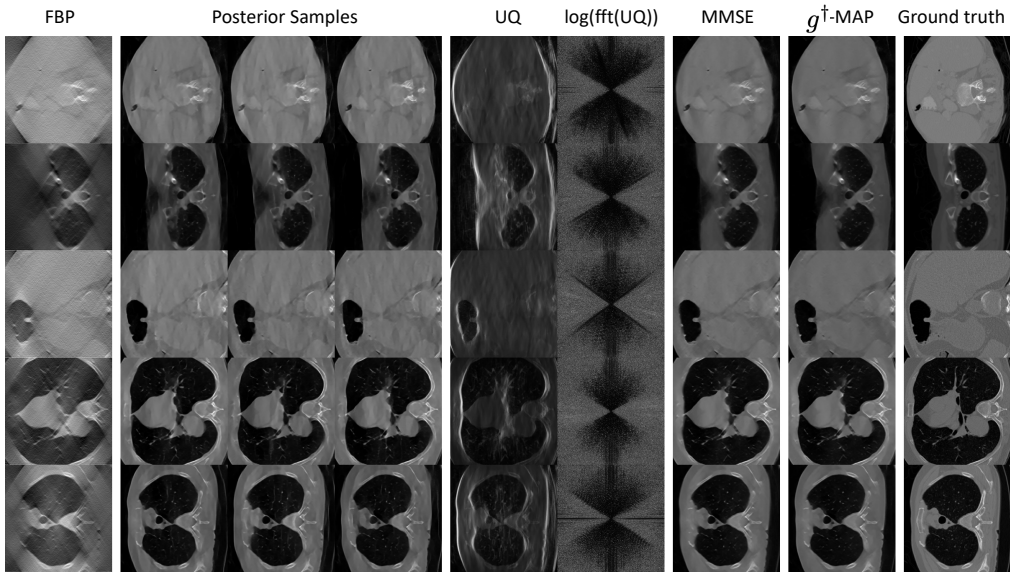


(d) Mask ($s = 32$)

Figure 7.14: Performance comparison over image restoration problems.



(a) Vertical missing cone (60° to 120°)



(b) Horizontal missing cone(-30° to 30°)

Figure 7.15: Limited-view CT in resolution 256×256

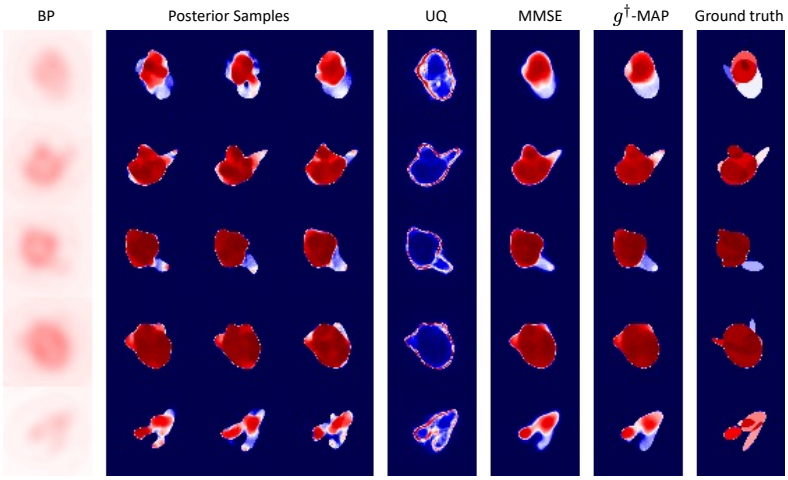


Figure 7.16: Inverse scattering ($\epsilon_r = 1.5$, full-view)

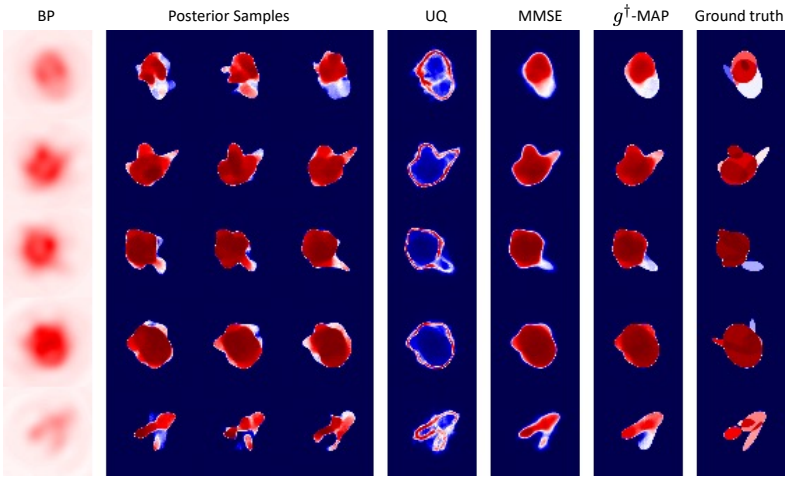


Figure 7.17: Inverse scattering ($\epsilon_r = 2$, full-view)

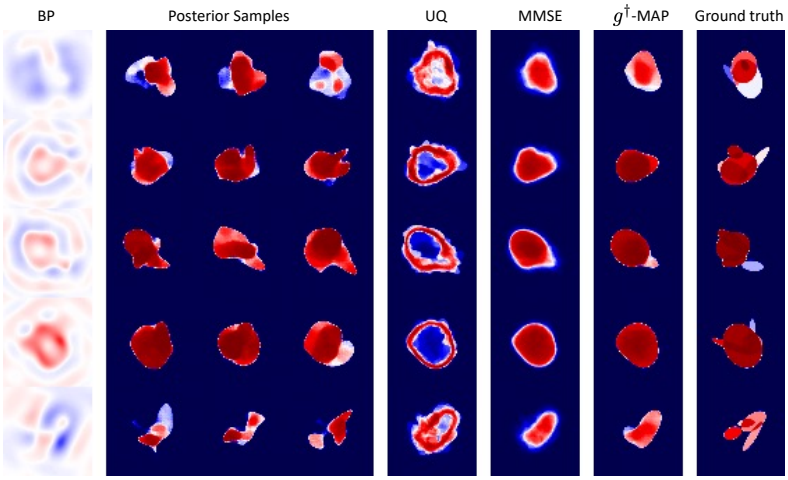


Figure 7.18: Inverse scattering ($\epsilon_r = 6$, full-view)

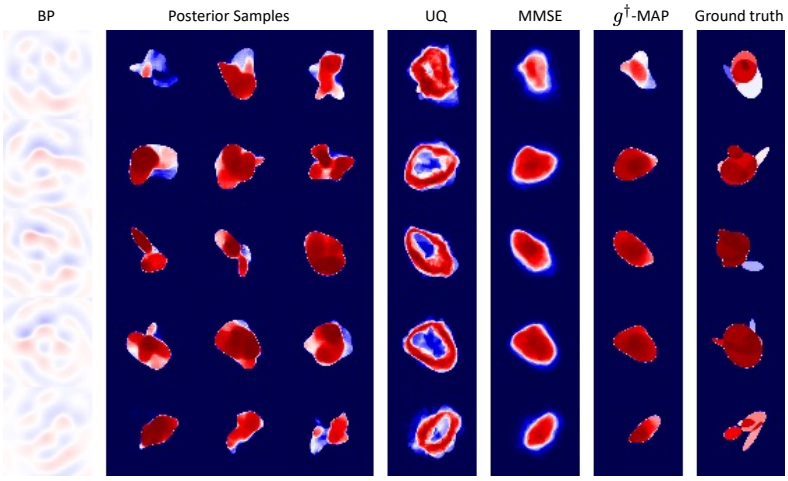


Figure 7.19: Inverse scattering ($\epsilon_r = 6$, top-view)

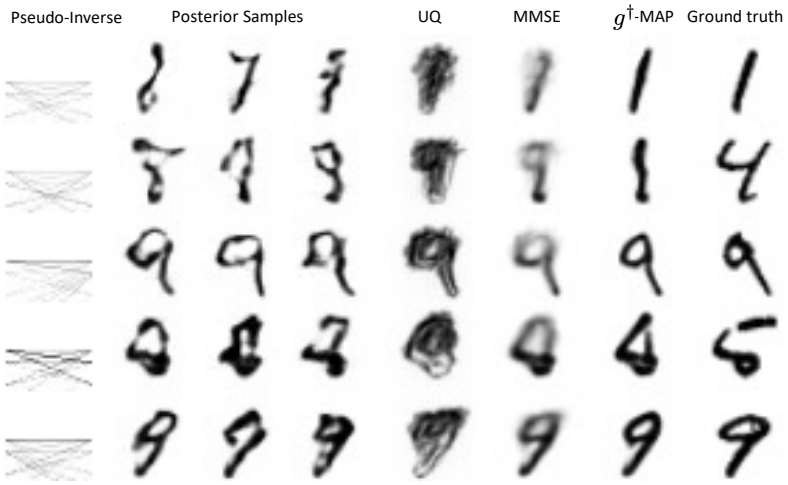


Figure 7.20: Travel-time tomography ($NS = 10$)

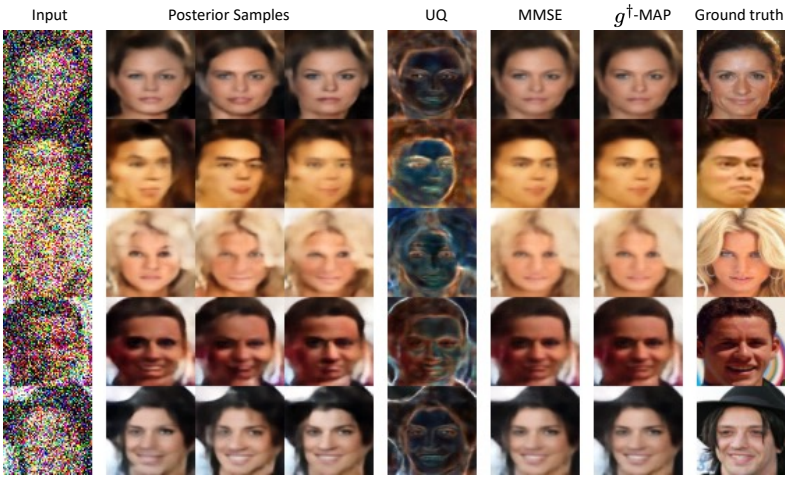


Figure 7.21: Denoising

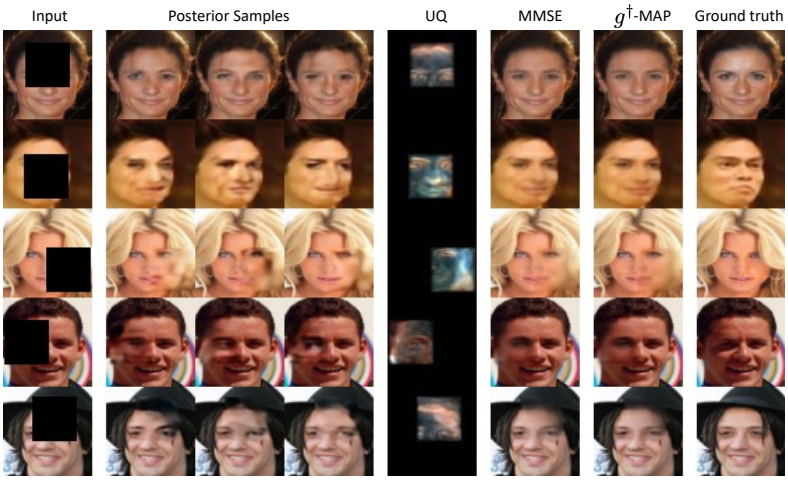


Figure 7.22: Mask ($s = 32$)

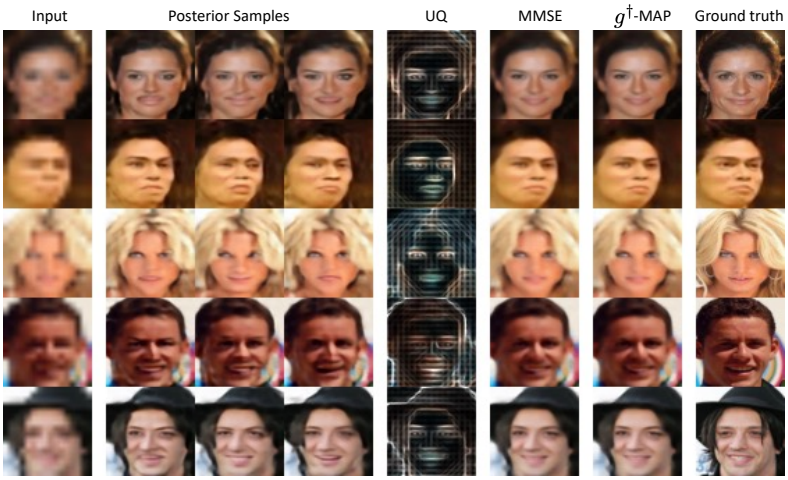


Figure 7.23: Super-resolution ($f = 4$)

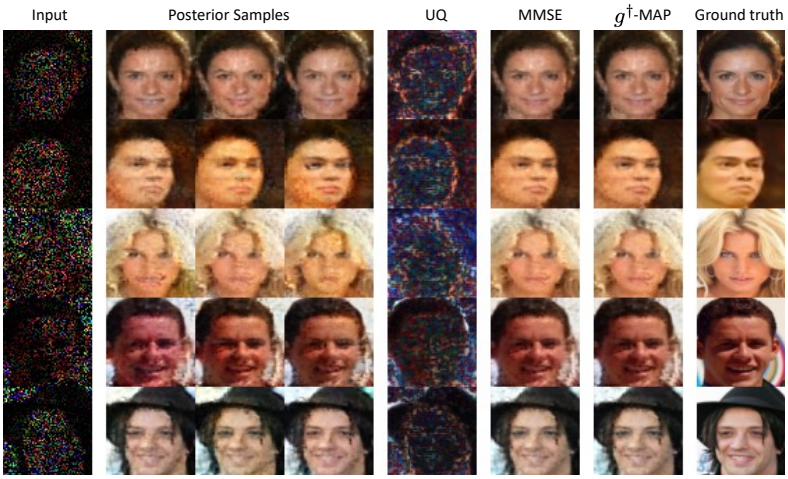


Figure 7.24: Random mask ($p = 0.2$)

Chapter 8

Deep Variational Inverse Scattering

In Chapter 7, we introduced C-Trumpets for rapid posterior sampling and uncertainty quantification across various imaging modalities. While C-Trumpets provide physically meaningful uncertainty quantification, the quality of the posterior samples, including the MMSE estimate, is significantly lower compared to single-point estimators like U-Net, especially for challenging tasks such as inverse scattering. In this chapter, we present a Bayesian version of U-Net, leveraging conditional normalizing flows for high-quality posterior sampling and uncertainty quantification. We demonstrate the model's performance in non-linear inverse scattering.

This variety of solutions for inverse scattering suggests using methods that recover more than a single reconstruction [23, 194]. As discussed earlier, a probabilistic characterization of solutions enables us more reliable interpretation of reconstructions and gives an important measure of uncertainty as the problem gets more ill-posed. Estimating uncertainty is paramount in safety-critical tasks such as medical imaging [297], nuclear stockpile [298, 299], and more recently self-driving vehicles [300, 301].

There are a number of approaches to approximate or sample the posterior. Tarantola [302], as well as Stuart [303], provided a comprehensive review of inverse problems from a statistical point of view. Traditional approaches include variants of Markov chain Monte Carlo [304, 305] which exploits the operator structure. The main challenge is a large number of required forward simulations. To alleviate the computational cost, a class of methods [306, 307] employ data-driven model reduction. More recently, neural networks have shown promising results for posterior approximation. Variational U-Net is proposed by Esser *et al.* [308] to generate images from poses and exploited by Jin *et al.* [309] for reservoir simulations where the network is trained with the evidence lower bound (ELBO), similar to variational autoencoders [197]. The authors of [310] propose a hierarchical probabilistic model to generate posterior samples and uncertainty estimates for image segmentation. Bayesian convolutional neural networks are used for posterior sampling in several computational imaging problems [244, 311]. As shown in

Chapter 7, conditional injective flows [23] can generate physically meaningful uncertainty estimates for inverse scattering. However, the quality of reconstructions is inferior to highly successful image-to-image regression models like the U-Net [25], especially for non-linear inverse problems.

In this chapter, we propose U-Flow, a Bayesian U-Net based on conditional normalizing flows. U-Flow benefits from favorable aspects of the U-Net: it yields high-quality reconstructions even for non-linear inverse problems, while enabling regularized posterior sampling and meaningful uncertainty estimates. We show that the MMSE estimate from U-Flow has comparable quality to that of U-Net and it significantly outperforms C-Trumpet in posterior sampling and uncertainty quantification.

8.1 Wave scattering model

We use the Helmholtz (time-harmonic wave) equation as the forward model,[1]

$$\triangle u - \frac{\omega^2}{c^2} u = -ig \text{ in } \Omega. \tag{8.1}$$

The medium is characterized by the wave speed c. In this work, we restrict ourselves to the structural heterogeneity in the wave speed, while leaving the density constant and the attenuation zero. The source g is a point source. The domain is unbounded, thus we append perfectly matched layers [313] to the computation domain. We use the default GMRES method to solve (8.1).

8.1.1 Measurement setup

To image the heterogeneity, we place the (collocated) sources and sensors on a circle. Each source takes a turn to emit a circular wave that scatters through the medium and gets measured at all the sensors. Thus our measurement data are square matrices of complex-valued entries.

The inverse problem is to recover the medium from the measurement data. To set up the notation, let us denote the unknown parameters as $\mathbf{x} := c|_\Omega$ and the (back-projected) measurements as $\mathbf{y} := u|_{\partial\Omega}$. Figure 8.1 illustrates two examples of the measurement setup.

The `j-wave` package [312] gives easy access to the back-projection (BP) images thanks to automatic differentiation. The BP is obtained by applying the Jacobian of the discrete forward model to the measurement mismatch. This auto-diff is the main component in the discretize-then-optimize regime. The advantage is that the Jacobian

[1] Here we numerically solve the equation using the `j-wave` package [312].

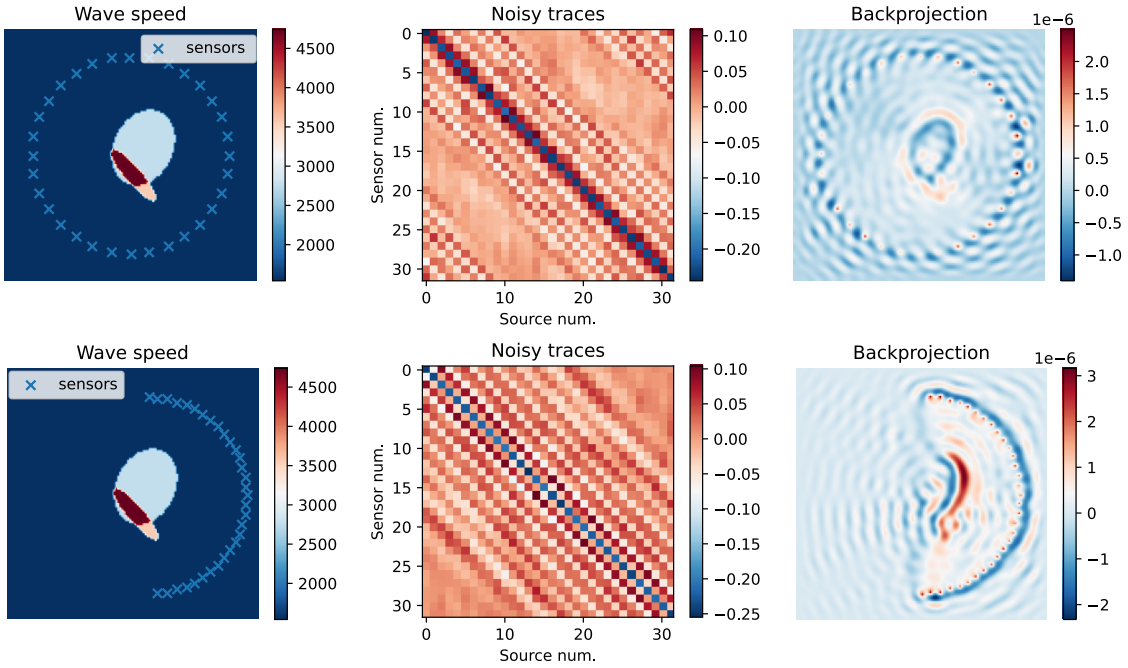


Figure 8.1: Examples of the inverse medium scattering problem. Top row: full-view measurement. Bottom row: limited-view measurement. We add 30 dB noise to the measurements. The back-projections are computed from auto-differentiation.

exactly matches the numerical model being used and it can easily extend to other medium parameters such as density and attenuation.

The BP image is computed by taking the derivative of the measurement loss

$$\mathbf{y} \mapsto \frac{\partial}{\partial x} \|\mathbf{y} - \hat{y}(x)\|_2^2.$$

With a slight abuse of notation, we will use these BP images, denoted as $\mathbf{y}$, to estimate the posterior distribution of medium wave speed.

8.1.2 Training data

We generate 4000 medium samples, each containing a set of random ellipsoidal scatterers, along with their discrete boundary measurements. The computation domain is a 128×128 grid with resolution $\Delta x = 10^{-3}$ m. The background wave speed is 1540 m/s and the maximum contrast is close to 4 times the background. The angular frequency is $\omega = 7 \cdot 10^5$ s^{-1}, which corresponds to 13 grid points per wavelength. We use 32 complex-valued measurements corrupted with Gaussian noise.

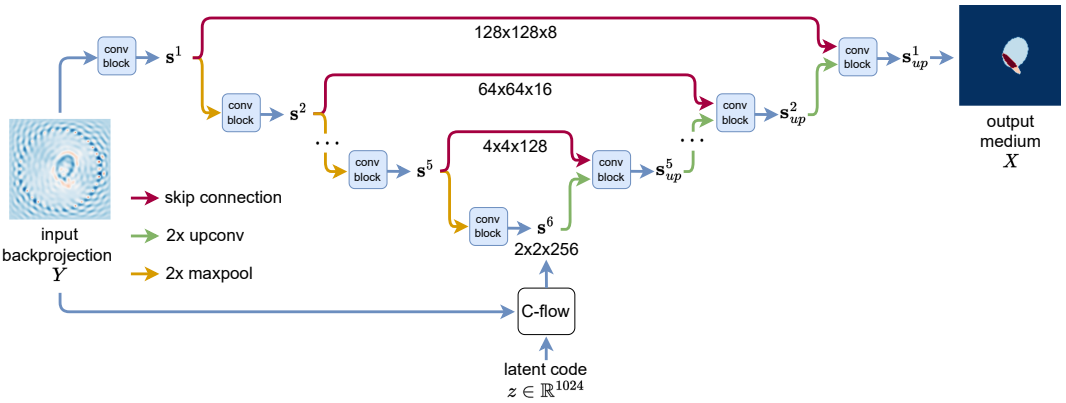


Figure 8.2: Network architecture of U-Flow. In our implementation, the U-Net component has 6 scale levels $(\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^6)$. At the coarsest level, the latent variable $\mathbf{s}^6$ is a $2 \times 2 \times 256$ tensor, which is the output of a conditional flow model, conditioned by original input.

8.2 U-Flow

We propose U-Flow, a probabilistic model combining U-Net [25] and conditional normalizing flows [314] to approximate the posterior distribution.

8.2.1 U-Net

Ronneberger *et al.* [25] originally developed the U-Net for medical image segmentation. It has since been adapted to many image-to-image tasks, often achieving (near-)state-of-the-art performance. The U-Net is an encoder-decoder network

$$\text{UNet}_\phi \overset{\text{def.}}{=} \text{dec} \circ \text{enc}.$$

The encoder and the decoder are both convolutional neural networks with pooling layers. The encoder takes the measurements $\mathbf{y}$ as input and produces features in different scales. The decoder then takes the computed features and reconstructs the target signal $\mathbf{x}$.

The encoder and decoder are jointly optimized using the mean-square error (MSE) loss

$$\phi^\star = \operatorname*{argmin}_\phi \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \text{UNet}_\phi(\mathbf{y}_i)\|_2^2. \tag{8.2}$$

While the U-Net produces high-quality reconstructions, its output is a single estimate. In this chapter, we present a probabilistic version of the U-Net for amortized Bayesian inference.

8.2.2 Conditional normalizing flows

As explained in Section 5.1.1, normalizing flows [21, 70, 187] are a class of likelihood-based generative models. They transform a simple and known distribution into an unknown data distribution by a sequence of invertible mappings. Invertibility enables efficient likelihood estimation and maximum-likelihood (ML) parameter fitting. Vlašić et al. [31] demonstrated the effectiveness of flow-based generative models in regularizing the inverse obstacle scattering problem. In Section 7.3, we reviewed conditional versions of normalizing flows [314] that allow for approximation of posterior distributions. However, regular conditional flows require the latent space dimension to equal the data dimension, which leads to a large network and slow training. Moreover, as the range of conditional flows covers the entire space, the posterior samples are not constrained to an image distribution and are often of low quality in ill-posed nonlinear inverse problems [23].

8.2.3 Our approach

Since the input and output of flow models must have the same dimension, it is opportune to use flows to model low-dimensional latent spaces rather than images directly. We use flows to approximate the posterior distribution of the coarsest scale in the U-Net. Concretely, as shown in Figure 8.2, the encoder of the U-Net takes the BPs and produces features in six scales, $\text{enc}(\mathbf{y}) = (\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^6)$ for $\mathbf{y} \in \mathbb{R}^D$. These multi-scale features feed to the decoder to reconstruct a *single* estimate of the target signal $\hat{\mathbf{x}}(\mathbf{y}) = \text{dec}(\mathbf{s}^6, \mathbf{s}^5, \ldots, \mathbf{s}^1)$. To generate posterior samples, we let a flow model learn the conditional distribution of features at the coarsest scale $p_{\mathbf{s}^6|Y}$ where $\mathbf{s}^6 \in \mathbb{R}^d$ and $d \ll D$. We first train a U-Net with the loss in (8.2) and compute the coarsest scale features of the BPs samples in the training set. Having obtained a paired training set of the BPs and the corresponding features $\{(\mathbf{s}_i^6, \mathbf{y}_i)\}_{i=1}^N$, we then train a flow model. We use the conditional version of Glow [70], where we deploy conditional coupling blocks proposed in [314] to condition the generation on back-projections. The conditional flow model is trained using amortized inference loss as,

$$\theta^\star = \underset{\theta}{\text{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left(-\log p_Z(\mathbf{z}_i) + \log |\det J_{f_\theta}| \right), \tag{8.3}$$

where $\mathbf{z}_i = f_\theta^{-1}(\mathbf{s}_i^6, \mathbf{y}_i)$, J_{f_θ} is the Jacobian matrix of f_θ and p_Z is a multivariate Gaussian distribution. In this chapter, instead of directly approximating $p_{X|Y}$, we approximate the posterior distribution of the features in the lowest scale of the U-Net $p_{s_i^6|Y}$ using conditional normalizing flows as shown in Figure 8.2. When the conditional flow model is trained, we can generate posterior samples for each BP $y^\star$,

$$\mathbf{x}_{\text{post}}(\mathbf{y}^\star) = \text{dec}(f_\theta(\mathbf{z}), \mathbf{s}^5, ..., \mathbf{s}^1) \tag{8.4}$$

Table 8.1: SNR of MMSE estimate (computed over 25 posterior samples for flow-based models) of different models over inverse medium scattering in two setups

	BP	C-Trumpets [23]	U-Net [25]	U-Flow
Full-view	-17.8	5.7	**11.3**	**11.3**
Side-view	-17.7	5.2	**9.7**	9.4

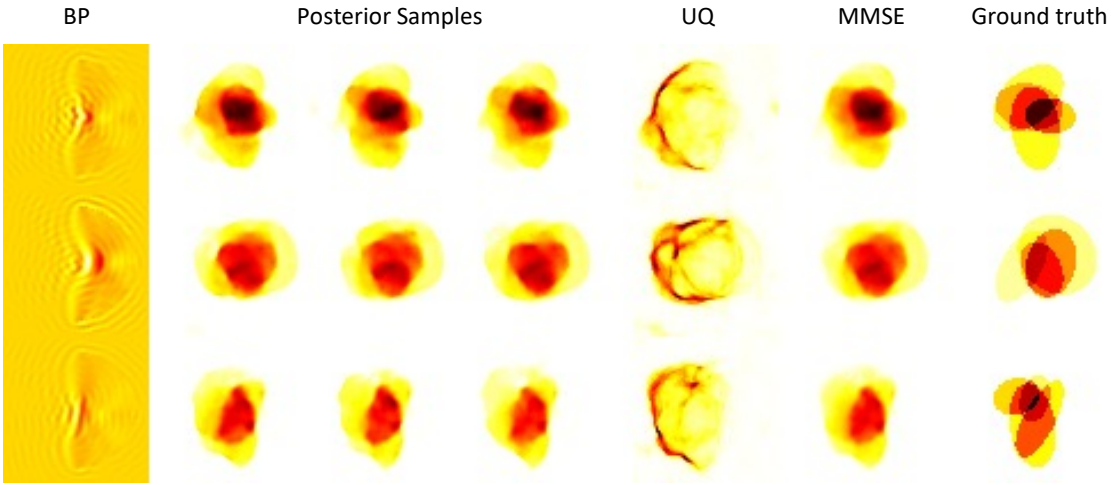
where $\mathbf{z} \sim \mathcal{N}(0, I)$ and $(\mathbf{s}^1, ..., \mathbf{s}^5, \cdot) = \text{enc}(\mathbf{y}^\star)$. The key advantage of the proposed model is that the posterior samples have a low-dimensional structure, which acts as a strong regularizer for ill-posed inverse problems.

8.3 Experiments

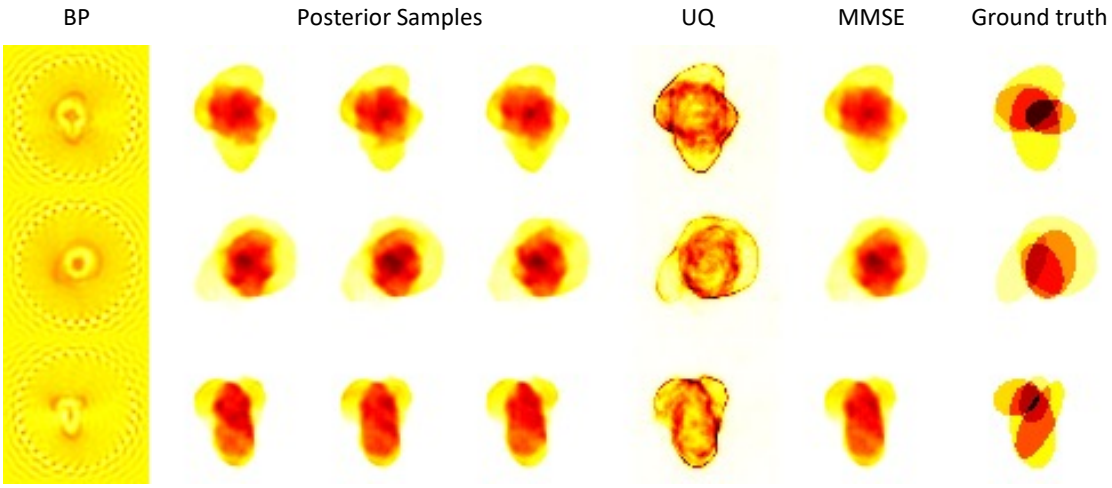
We trained U-Flow for 600 epochs in total, 300 for the U-Net and 300 for the conditional flow model. We used the Adam optimizer [97] with the learning rate set to 10^{-4}. The conditional flow model was composed of 24 Glow-blocks, each containing activation normalization, the 1×1 convolution, and a conditional coupling layer. We compared U-Flow with C-Trumpet [23], a conditional injective flow which is well-suited for solving ill-posed inverse problems. In our experiments, U-Flow and C-Trumpet had 9M and 14M trainable parameters, respectively. The MMSE estimate was calculated by averaging 25 posterior samples. The UQ is performed by dividing the pixel-wise standard deviation on 25 posterior samples to the MMSE estimate to show the relative error.

Figure 8.3a illustrates the performance of U-Flow on inverse medium scattering with a limited-view sensing configuration, where the receivers and incident waves are located on the *right-hand* side of the medium. This experiment shows that U-Flow can generate various posterior samples and capture a physically meaningful UQ. Notice that U-Flow assigned more uncertainty to the *left-hand* side of the medium (red region) which aligns with the measurements configuration. This experiment can clearly show the significance of posterior sampling and uncertainty quantification in practice which can help the practitioners to make a more informed decision or order more measurements in the critical regions. The results for the full-view configuration are shown in Figure 8.3b. As expected, the full-view configuration yields better posterior samples than the limited-view configuration.

Figure 8.4 compares the performance of U-Flow and C-Trumpet. The experiment shows that U-Flow remarkably outperforms C-Trumpet both in posterior sampling and UQ. Table 8.1 gives a quantitative comparison of U-Flow with baselines, including the basic U-Net [25]. U-Flow exhibits comparable results to the U-Net while giving access to posterior samples and UQ.



(a) Limited-view: the receivers and incident waves are only on the *right-hand* side of the object; U-Flow could reliably capture a physically meaningful uncertainty estimate by showing more uncertainty (red regions) on the *left* part of the object.



(b) Full-view: the receivers and incident waves are uniformly distributed around the object.

Figure 8.3: Performance of U-Flow over inverse medium scattering

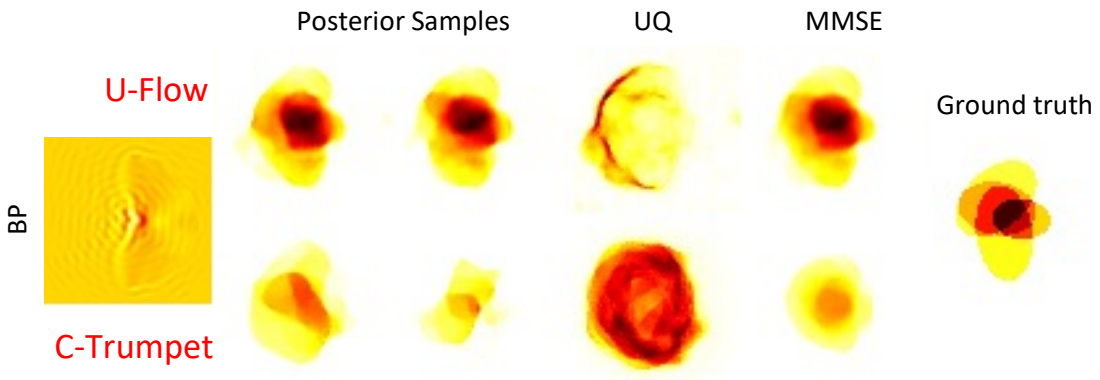


Figure 8.4: Performance comparison of U-Flow and C-Trumpet [23] for the limited-view problem (no sensors on the *left-hand* side); U-Flow outperforms C-Trumpet in both posterior sampling and uncertainty quantification by assigning more uncertainty in the *left* part of the object (red regions).

8.4 Summary

We demonstrated that the dichotomy between high-quality inversions without UQ by standard point-estimate networks, and low-quality inversions with UQ by the various conditional generative models is a false one. By combining a low-dimensional flow with a U-Net, we get the best of both worlds. The reconstructions are very fast—orders of magnitude faster than with the standard iterative methods.

Chapter 9

Looking Forward

In this thesis, we focused on enhancing the reliability of deep learning models for imaging inverse problems. In the first part, we introduced the notion of locality in the context of implicit neural representation for various imaging modalities from image denoising, super-resolution, and CT imaging. Our experiments demonstrated that the proposed local processing model not only exhibits strong generalization capabilities but also benefits from computationally efficient training, making it a practical and scalable approach for diverse imaging tasks.

In the second part of this thesis, we investigated injective neural networks within the context of Bayesian imaging. We developed both unsupervised and supervised learning frameworks using variational inference to approximate the posterior distribution. Through extensive experiments on various imaging problems, such as inverse scattering and CT imaging, we demonstrated that the proposed framework generates high-quality posterior samples and provides physically meaningful uncertainty quantification. This is essential for ensuring reliable analysis and accurate downstream interpretation. In the following, we briefly discuss the potential opportunities for future research.

9.1 Locality for 3D reconstruction

While deep learning has shown promising results in solving 2D imaging problems, its application to 3D reconstruction remains challenging due to high computational and memory demands. In Chapters 2, 3, and 4, we developed local processing models for 2D imaging tasks. Future research could extend these models to handle large 3D volumes, leveraging their strong generalization capabilities and computational efficiency. Such an extension would make deep learning more accessible for scientific applications, helping bridge the implementation gap for real-world, high-dimensional problems [2, 315, 316].

9.2 Bayesian modeling of local processing models

One interesting avenue for future research is the development of probabilistic versions of the proposed local imaging frameworks. This approach is particularly relevant for imaging problems characterized by significant uncertainties arising from noise, ill-posedness, and model mismatches. In scenarios where pixel-wise uncertainty is sufficient, we can streamline our methodology by estimating the associated uncertainty for each recovered coordinate independently. Although this method may not yield meaningful posterior samples, it is well-suited for uncertainty quantification, as it allows for the estimation of pixel-wise uncertainties with reduced computational complexity.

9.3 Solving wave-based PDEs with a generative prior

In Chapter 4, we introduced our functional generator, FunkNN, which learns the distribution of functions by combining any off-the-shelf generator with our local processing super-resolution network. We utilized this functional generator as a prior for addressing simple stylized PDE-based inverse problems. For future work, this expressive framework could be applied to solve scientific PDEs, such as the Helmholtz or wave equations. This robust prior holds promise for ill-posed scenarios, particularly when measurements are limited.

Bibliography

[1] G. Wang, H. Yu, and B. De Man, "An outlook on x-ray ct research and development," *Medical physics*, vol. 35, no. 3, pp. 1051–1064, 2008.

[2] M. Holler, M. Guizar-Sicairos, E. H. Tsai, R. Dinapoli, E. Müller, O. Bunk, J. Raabe, and G. Aeppli, "High-resolution non-destructive three-dimensional imaging of integrated circuits," *Nature*, vol. 543, no. 7645, pp. 402–406, 2017.

[3] R. E. Blahut, *Theory of remote image formation*. Cambridge University Press, 2004.

[4] N. Kaiser and G. Squires, "Mapping the dark matter with weak gravitational lensing," *Astrophysical Journal, Part 1 (ISSN 0004-637X), vol. 404, no. 2, p. 441-450.*, vol. 404, pp. 441–450, 1993.

[5] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on image processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[6] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Advances and challenges in super-resolution," *International Journal of Imaging Systems and Technology*, vol. 14, no. 2, pp. 47–57, 2004.

[7] I. A. Elbakri and J. A. Fessler, "Statistical image reconstruction for polyenergetic x-ray computed tomography," *IEEE transactions on medical imaging*, vol. 21, no. 2, pp. 89–99, 2002.

[8] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse mri: The application of compressed sensing for rapid mr imaging," *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[9] X. Chen, *Computational methods for electromagnetic inverse scattering*, vol. 244. Wiley Online Library, 2018.

[10] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 460–489, 2005.

[11] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 39–56, 2020.

[12] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[13] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[14] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017.

[15] C. M. Hyun, H. P. Kim, S. M. Lee, S. Lee, and J. K. Seo, "Deep learning for undersampled MRI reconstruction," *Physics in Medicine & Biology*, vol. 63, no. 13, p. 135007, 2018.

[16] Z. Wei and X. Chen, "Deep-learning schemes for full-wave nonlinear inverse scattering problems," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 4, pp. 1849–1860, 2018.

[17] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

[18] X. Li, Y. Dai, Y. Ge, J. Liu, Y. Shan, and L.-Y. Duan, "Uncertainty modeling for out-of-distribution generalization," in *International Conference on Learning Representations*, 2022.

[19] A. Khorashadizadeh, V. Debarnot, T. Liu, and I. Dokmanić, "Glimpse: Generalized local imaging with mlps," *arXiv preprint arXiv:2401.00816*, 2024.

[20] A. Khorashadizadeh, A. Chaman, V. Debarnot, and I. Dokmanić, "Funknn: Neural interpolation for functional generation," in *The Eleventh International Conference on Learning Representations*, 2023.

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

[22] A. Khorashadizadeh, V. Khorashadizadeh, S. Eskandari, G. A. Vandenbosch, and I. Dokmanić, "Deep injective prior for inverse scattering," *IEEE Transactions on Antennas and Propagation*, 2023.

[23] A. Khorashadizadeh, K. Kothari, L. Salsi, A. A. Harandi, M. de Hoop, and I. Dokmanić, "Conditional injective flows for bayesian imaging," *IEEE Transactions on Computational Imaging*, vol. 9, pp. 224–237, 2023.

[24] A. Khorashadizadeh, A. Aghababaei, T. Vlašić, H. Nguyen, and I. Dokmanić, "Deep variational inverse scattering," in *2023 17th European Conference on Antennas and Propagation (EuCAP)*, pp. 1–5, IEEE, 2023.

[25] 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*, pp. 234–241, Springer, 2015.

[26] N. Davoudi, X. L. Deán-Ben, and D. Razansky, "Deep learning optoacoustic tomography with sparse data," *Nature Machine Intelligence*, vol. 1, no. 10, pp. 453–460, 2019.

[27] T. Liu, A. Chaman, D. Belius, and I. Dokmanić, "Learning multiscale convolutional dictionaries for image reconstruction," *IEEE Transactions on Computational Imaging*, vol. 8, pp. 425–437, 2022.

[28] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7462–7473, 2020.

[29] M. Atzmon and Y. Lipman, "Sal: Sign agnostic learning of shapes from raw data," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2565–2574, 2020.

[30] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, "Deep local shapes: Learning local sdf priors for detailed 3d reconstruction," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pp. 608–625, Springer, 2020.

[31] T. Vlašić, H. Nguyen, A. Khorashadizadeh, and I. Dokmanić, "Implicit neural representation for mesh-free inverse obstacle scattering," in *2022 56th Asilomar Conference on Signals, Systems, and Computers*, pp. 947–952, IEEE, 2022.

[32] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019.

[33] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pp. 523–540, Springer, 2020.

[34] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser, *et al.*, "Local implicit grid representations for 3d scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6001–6010, 2020.

[35] Y. Chen, S. Liu, and X. Wang, "Learning continuous image representation with local implicit image function," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8628–8638, 2021.

[36] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[37] A. Susmelj, M. Macuglia, N. Tagasovska, R. Sutter, S. Caprara, J.-P. Thiran, and E. Konukoglu, "Uncertainty modeling for fine-tuned implicit functions," *arXiv preprint arXiv:2406.12082*, 2024.

[38] E. Dupont, Y. W. Teh, and A. Doucet, "Generative models as distributions of functions," *arXiv preprint arXiv:2102.04776*, 2021.

[39] E. Dupont, H. Kim, S. Eslami, D. Rezende, and D. Rosenbaum, "From data to functa: Your data point is a function and you can treat it like one," *arXiv preprint arXiv:2201.12204*, 2022.

[40] 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*, pp. 945–948, IEEE, 2013.

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

[42] 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.

[43] A. Levin and B. Nadler, "Natural image denoising: Optimality and inherent bounds," in *CVPR 2011*, pp. 2833–2840, IEEE, 2011.

[44] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 2392–2399, IEEE, 2012.

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

[46] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3929–3938, 2017.

[47] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-cnn for image restoration," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 773–782, 2018.

[48] B. Park, S. Yu, and J. Jeong, "Densely connected hierarchical network for image denoising," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 0–0, 2019.

[49] F. Jia, W. H. Wong, and T. Zeng, "Ddunet: Dense dense u-net with applications in image denoising," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 354–364, 2021.

[50] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, "Plug-and-play image restoration with deep denoiser prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 6360–6376, 2021.

[51] 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*, pp. 770–778, 2016.

[52] R. Mandelbaum, "Weak lensing for precision cosmology," *Annual Review of Astronomy and Astrophysics*, vol. 56, no. 1, pp. 393–433, 2018.

[53] B. Horowitz, U. Seljak, and G. Aslanyan, "Efficient optimal reconstruction of linear fields and band-powers from cosmological data," *Journal of Cosmology and Astroparticle Physics*, vol. 2019, p. 035, oct 2019.

[54] Lanusse, F., Starck, J.-L., Leonard, A., and Pires, S., "High resolution weak lensing mass mapping combining shear and flexion," *A&A*, vol. 591, p. A2, 2016.

[55] M. A. Price, X. Cai, J. D. McEwen, M. Pereyra, T. D. Kitching, and L. D. E. S. Collaboration, "Sparse Bayesian mass mapping with uncertainties: local credible intervals," *Monthly Notices of the Royal Astronomical Society*, vol. 492, pp. 394–404, 12 2019.

[56] M. A. Price, J. D. McEwen, X. Cai, T. D. Kitching, C. G. R. Wallis, and (for the LSST Dark Energy Science Collaboration), "Sparse Bayesian mass mapping with uncertainties: hypothesis testing of structure," *Monthly Notices of the Royal Astronomical Society*, vol. 506, pp. 3678–3690, 07 2021.

[57] Starck, J.-L., Themelis, K. E., Jeffrey, N., Peel, A., and Lanusse, F., "Weak-lensing mass reconstruction using sparsity and a gaussian random field," *A&A*, vol. 649, p. A99, 2021.

[58] N. Jeffrey, F. Lanusse, O. Lahav, and J.-L. Starck, "Deep learning dark matter map reconstructions from DES SV weak lensing data," *Monthly Notices of the Royal Astronomical Society*, vol. 492, pp. 5023–5029, 01 2020.

[59] Remy, B., Lanusse, F., Jeffrey, N., Liu, J., Starck, J.-L., Osato, K., and Schrabback, T., "Probabilistic mass-mapping with neural score estimation," *A & A*, vol. 672, p. A51, 2023.

[60] J. Zbontar, F. Knoll, A. Sriram, T. Murrell, Z. Huang, M. J. Muckley, A. Defazio, R. Stern, P. Johnson, M. Bruno, M. Parente, K. J. Geras, J. Katsnelson, H. Chandarana, Z. Zhang, M. Drozdzal, A. Romero, M. Rabbat, P. Vincent, N. Yakubova, J. Pinkerton, D. Wang, E. Owens, C. L. Zitnick, M. P. Recht, D. K. Sodickson, and Y. W. Lui, "fastMRI: An open dataset and benchmarks for accelerated MRI," 2018.

[61] Z. Ramzi, P. Ciuciu, and J.-L. Starck, "Benchmarking mri reconstruction neural networks on large public datasets," *Applied Sciences*, vol. 10, no. 5, 2020.

[62] A. R. Thompson, J. M. Moran, and J. Swenson, George W., *Interferometry and Synthesis in Radio Astronomy, 3rd Edition*. 2017.

[63] L. Pratley, J. D. McEwen, M. d'Avezac, R. E. Carrillo, A. Onose, and Y. Wiaux, "Robust sparse image reconstruction of radio interferometric observations with purify," *Monthly Notices of the Royal Astronomical Society*, vol. 473, pp. 1038–1058, 09 2017.

[64] T. I. Liaudat, M. Mars, M. A. Price, M. Pereyra, M. M. Betcke, and J. D. McEwen, "Scalable Bayesian uncertainty quantification with data-driven priors for radio interferometric imaging," *arXiv e-prints*, p. arXiv:2312.00125, Nov. 2023.

[65] M. T. McCann, K. H. Jin, and M. Unser, "Convolutional neural networks for inverse problems in imaging: A review," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 85–95, 2017.

[66] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed sensing using generative models," in *International conference on machine learning*, pp. 537–546, PMLR, 2017.

[67] 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*, pp. 399–409, PMLR, 2020.

[68] B. Kawar, M. Elad, S. Ermon, and J. Song, "Denoising diffusion restoration models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23593–23606, 2022.

[69] T. Liu, T. Yang, Q. Zhang, and Q. Lei, "Optimization for amortized inverse problems," in *International Conference on Machine Learning*, pp. 22289–22319, PMLR, 2023.

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

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

[72] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play admm for image restoration: Fixed-point convergence and applications," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 84–98, 2016.

[73] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (red)," *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 1804–1844, 2017.

[74] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2305–2318, 2018.

[75] K. Wei, A. Aviles-Rivero, J. Liang, Y. Fu, H. Huang, and C.-B. Schönlieb, "Tfpnp: Tuning-free plug-and-play proximal algorithms with applications to inverse imaging problems," *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 699–746, 2022.

[76] R. Ahmad, C. A. Bouman, G. T. Buzzard, S. Chan, S. Liu, E. T. Reehorst, and P. Schniter, "Plug-and-play methods for magnetic resonance imaging: Using denoisers for image recovery," *IEEE signal processing magazine*, vol. 37, no. 1, pp. 105–116, 2020.

[77] K. Wei, A. Aviles-Rivero, J. Liang, Y. Fu, C.-B. Schönlieb, and H. Huang, "Tuning-free plug-and-play proximal algorithm for inverse imaging problems," in *International Conference on Machine Learning*, pp. 10158–10169, PMLR, 2020.

[78] Y. Zhu, K. Zhang, J. Liang, J. Cao, B. Wen, R. Timofte, and L. Van Gool, "Denoising diffusion models for plug-and-play image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1219–1229, 2023.

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

[80] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *International conference on machine learning*, pp. 2990–2999, PMLR, 2016.

[81] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv preprint arXiv:2104.13478*, 2021.

[82] T. S. Cohen and M. Welling, "Steerable cnns," *arXiv preprint arXiv:1612.08498*, 2016.

[83] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation equivariant cnns for digital pathology," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11*, pp. 210–218, Springer, 2018.

[84] A. Chaman and I. Dokmanić, "Truly shift-equivariant convolutional neural networks with adaptive polyphase upsampling," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*, pp. 1113–1120, IEEE, 2021.

[85] A. Chaman and I. Dokmanić, "Truly shift-invariant convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3773–3783, 2021.

[86] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[87] O. Puny, M. Atzmon, H. Ben-Hamu, I. Misra, A. Grover, E. J. Smith, and Y. Lipman, "Frame averaging for invariant and equivariant network design," *arXiv preprint arXiv:2110.03336*, 2021.

[88] A. Sannai, M. Kawano, and W. Kumagai, "Equivariant and invariant reynolds networks," *arXiv preprint arXiv:2110.08092*, 2021.

[89] Z. Zhao and A. Singer, "Rotationally invariant image representation for viewing direction classification in cryo-em," *Journal of structural biology*, vol. 186, no. 1, pp. 153–166, 2014.

[90] M. Diwakar and M. Kumar, "A review on ct image noise and its denoising," *Biomedical Signal Processing and Control*, vol. 42, pp. 73–88, 2018.

[91] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[92] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[93] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[94] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[95] D. Nelson, A. Pillepich, V. Springel, R. Pakmor, R. Weinberger, S. Genel, P. Torrey, M. Vogelsberger, F. Marinacci, and L. Hernquist, "First results from the tng50 simulation: galactic outflows driven by supernovae and black hole feedback," *Monthly Notices of the Royal Astronomical Society*, vol. 490, no. 3, pp. 3234–3261, 2019.

[96] 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*, vol. 32, 2019.

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

[98] K. Osato, J. Liu, and Z. Haiman, "κTNG: effect of baryonic processes on weak lensing with IllustrisTNG simulations," *Monthly Notices of the Royal Astronomical Society*, vol. 502, pp. 5593–5602, 02 2021.

[99] A. Pillepich, D. Nelson, L. Hernquist, V. Springel, R. Pakmor, P. Torrey, R. Weinberger, S. Genel, J. P. Naiman, F. Marinacci, and M. Vogelsberger, "First results from the IllustrisTNG simulations: the stellar mass content of groups and clusters of galaxies," *Monthly Notices of the Royal Astronomical Society*, vol. 475, pp. 648–675, Mar. 2018.

[100] V. Springel, R. Pakmor, A. Pillepich, R. Weinberger, D. Nelson, L. Hernquist, M. Vogelsberger, S. Genel, P. Torrey, F. Marinacci, and J. Naiman, "First results from the IllustrisTNG simulations: matter and galaxy clustering," *Monthly Notices of the Royal Astronomical Society*, vol. 475, pp. 676–698, Mar. 2018.

[101] J. P. Naiman, A. Pillepich, V. Springel, E. Ramirez-Ruiz, P. Torrey, M. Vogelsberger, R. Pakmor, D. Nelson, F. Marinacci, L. Hernquist, R. Weinberger, and S. Genel, "First results from the IllustrisTNG simulations: a tale of two elements - chemical evolution of magnesium and europium," *Monthly Notices of the Royal Astronomical Society*, vol. 477, pp. 1206–1224, June 2018.

[102] D. Nelson, A. Pillepich, V. Springel, R. Weinberger, L. Hernquist, R. Pakmor, S. Genel, P. Torrey, M. Vogelsberger, G. Kauffmann, F. Marinacci, and J. Naiman, "First results from the IllustrisTNG simulations: the galaxy colour bimodality," *Monthly Notices of the Royal Astronomical Society*, vol. 475, pp. 624–647, Mar. 2018.

[103] F. Marinacci, M. Vogelsberger, R. Pakmor, P. Torrey, V. Springel, L. Hernquist, D. Nelson, R. Weinberger, A. Pillepich, J. Naiman, and S. Genel, "First results from the IllustrisTNG simulations: radio haloes and magnetic fields," *Monthly Notices of the Royal Astronomical Society*, vol. 480, pp. 5113–5139, Nov. 2018.

[104] M. A. Price, J. D. McEwen, L. Pratley, and T. D. Kitching, "Sparse Bayesian mass-mapping with uncertainties: Full sky observations on the celestial sphere," *Monthly Notices of the Royal Astronomical Society*, vol. 500, pp. 5436–5452, 11 2020.

[105] R. Laureijs, J. Amiaux, S. Arduini, J. L. Augueres, J. Brinchmann, R. Cole, M. Cropper, C. Dabin, L. Duvet, A. Ealet, B. Garilli, P. Gondoin, L. Guzzo, J. Hoar, H. Hoekstra, R. Holmes, T. Kitching, T. Maciaszek, Y. Mellier, F. Pasian, W. Percival, J. Rhodes, G. Saavedra Criado, M. Sauvage, R. Scaramella, L. Valenziano, S. Warren, R. Bender, F. Castander, A. Cimatti, O. Le Fèvre, H. Kurki-Suonio, M. Levi, P. Lilje, G. Meylan, R. Nichol, K. Pedersen, V. Popa, R. Rebolo Lopez, H. W. Rix, H. Rottgering, W. Zeilinger, F. Grupp, P. Hudelot, R. Massey, M. Meneghetti, L. Miller, S. Paltani, S. Paulin-Henriksson, S. Pires, C. Saxton, T. Schrabback, G. Seidel, J. Walsh, N. Aghanim, L. Amendola, J. Bartlett, C. Baccigalupi, J. P. Beaulieu, K. Benabed, J. G. Cuby, D. Elbaz, P. Fosalba, G. Gavazzi, A. Helmi, I. Hook, M. Irwin, J. P. Kneib, M. Kunz, F. Mannucci, L. Moscardini, C. Tao, R. Teyssier, J. Weller, G. Zamorani, M. R. Zapatero Osorio, O. Boulade, J. J. Foumond, A. Di Giorgio, P. Guttridge, A. James, M. Kemp, J. Martignac, A. Spencer, D. Walton, T. Blümchen, C. Bonoli, F. Bortoletto, C. Cerna, L. Corcione, C. Fabron, K. Jahnke, S. Ligori, F. Madrid, L. Martin, G. Morgante, T. Pamplona, E. Prieto, M. Riva, R. Toledo, M. Trifoglio, F. Zerbi, F. Abdalla, M. Douspis, C. Grenet, S. Borgani, R. Bouwens, F. Courbin, J. M. Delouis, P. Dubath, A. Fontana, M. Frailis, A. Grazian, J. Koppenhöfer, O. Mansutti, M. Melchior, M. Mignoli, J. Mohr, C. Neissner, K. Noddle, M. Poncet, M. Scodeggio, S. Serrano, N. Shane, J. L. Starck, C. Surace, A. Taylor, G. Verdoes-Kleijn, C. Vuerli, O. R. Williams, A. Zacchei, B. Altieri, I. Escudero Sanz, R. Kohley, T. Oosterbroek, P. Astier, D. Bacon, S. Bardelli, C. Baugh, F. Bellagamba, C. Benoist, D. Bianchi, A. Biviano, E. Branchini, C. Carbone, V. Cardone, D. Clements, S. Colombi, C. Conselice, G. Cresci, N. Deacon, J. Dunlop, C. Fedeli, F. Fontanot, P. Franzetti, C. Giocoli, J. Garcia-Bellido, J. Gow, A. Heavens, P. Hewett, C. Heymans, A. Holland, Z. Huang, O. Ilbert, B. Joachimi, E. Jennins, E. Kerins, A. Kiessling, D. Kirk, R. Kotak, O. Krause, O. Lahav, F. van Leeuwen, J. Lesgourgues, M. Lombardi,

M. Magliocchetti, K. Maguire, E. Majerotto, R. Maoli, F. Marulli, S. Maurogordato, H. McCracken, R. McLure, A. Melchiorri, A. Merson, M. Moresco, M. Nonino, P. Norberg, J. Peacock, R. Pello, M. Penny, V. Pettorino, C. Di Porto, L. Pozzetti, C. Quercellini, M. Radovich, A. Rassat, N. Roche, S. Ronayette, E. Rossetti, B. Sartoris, P. Schneider, E. Semboloni, S. Serjeant, F. Simpson, C. Skordis, G. Smadja, S. Smartt, P. Spano, S. Spiro, M. Sullivan, A. Tilquin, R. Trotta, L. Verde, Y. Wang, G. Williger, G. Zhao, J. Zoubian, and E. Zucca, "Euclid Definition Study Report," *arXiv e-prints*, p. arXiv:1110.3193, Oct. 2011.

[106] G. Wang, J. C. Ye, and B. De Man, "Deep learning for tomographic image reconstruction," *Nature Machine Intelligence*, vol. 2, no. 12, pp. 737–748, 2020.

[107] J. Adler and O. Öktem, "Solving ill-posed inverse problems using iterative deep neural networks," *Inverse Problems*, vol. 33, p. 124007, Nov 2017.

[108] J. Adler and O. Öktem, "Learned primal-dual reconstruction," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1322–1332, 2018.

[109] D. Gilton, G. Ongie, and R. Willett, "Neumann networks for linear inverse problems in imaging," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 328–343, 2019.

[110] A. K. Maier, C. Syben, B. Stimpel, T. Würfl, M. Hoffmann, F. Schebesch, W. Fu, L. Mill, L. Kling, and S. Christiansen, "Learning with known operators reduces maximum error bounds," *Nature machine intelligence*, vol. 1, no. 8, pp. 373–380, 2019.

[111] A. Hauptmann, J. Adler, S. Arridge, and O. Öktem, "Multi-scale learned iterative reconstruction," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 843–856, 2020.

[112] Y. B. Sahel, J. P. Bryan, B. Cleary, S. L. Farhi, and Y. C. Eldar, "Deep unrolled recovery in sparse biological imaging," 2021.

[113] J. Leuschner, M. Schmidt, P. S. Ganguly, V. Andriiashen, S. B. Coban, A. Denker, D. Bauer, A. Hadjifaradji, K. J. Batenburg, P. Maass, and M. van Eijnatten, "Quantitative comparison of deep learning-based image reconstruction methods for low-dose and sparse-angle CT applications," *Journal of Imaging*, vol. 7, no. 3, 2021.

[114] L. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone-beam algorithm," *Josa a*, vol. 1, no. 6, pp. 612–619, 1984.

[115] H. K. Aggarwal, M. P. Mani, and M. Jacob, "Modl: Model-based deep learning architecture for inverse problems," *IEEE transactions on medical imaging*, vol. 38, no. 2, pp. 394–405, 2018.

[116] A. Graas, S. B. Coban, K. J. Batenburg, and F. Lucka, "Just-in-time deep learning for real-time x-ray computed tomography," *Scientific Reports*, vol. 13, no. 1, p. 20070, 2023.

[117] A. H. Andersen and A. C. Kak, "Simultaneous algebraic reconstruction technique (sart): a superior implementation of the art algorithm," *Ultrasonic imaging*, vol. 6, no. 1, pp. 81–94, 1984.

[118] S. Lunz, A. Hauptmann, T. Tarvainen, C.-B. Schonlieb, and S. Arridge, "On learned operator correction in inverse problems," *SIAM Journal on Imaging Sciences*, vol. 14, no. 1, pp. 92–127, 2021.

[119] S. Gupta, K. Kothari, V. Debarnot, and I. Dokmanić, "Differentiable uncalibrated imaging," *IEEE Transactions on Computational Imaging*, 2023.

[120] E. Kang, J. Min, and J. C. Ye, "A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction," *Medical physics*, vol. 44, no. 10, pp. e360–e375, 2017.

[121] B. Hamoud, Y. Bahat, and T. Michaeli, "Beyond local processing: Adapting cnns for ct reconstruction," in *European Conference on Computer Vision*, pp. 513–526, Springer, 2022.

[122] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[123] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24261–24272, 2021.

[124] Z. Pan, B. Zhuang, J. Liu, H. He, and J. Cai, "Scalable vision transformers with hierarchical pooling," in *Proceedings of the IEEE/cvf international conference on computer vision*, pp. 377–386, 2021.

[125] Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li, "Uformer: A general u-shaped transformer for image restoration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17683–17693, 2022.

[126] G. Bachmann, S. Anagnostidis, and T. Hofmann, "Scaling mlps: A tale of inductive bias," *arXiv preprint arXiv:2306.13575*, 2023.

[127] G. H. Golub and C. F. Van Loan, "An analysis of the total least squares problem," *SIAM journal on numerical analysis*, vol. 17, no. 6, pp. 883–893, 1980.

[128] I. Markovsky and S. Van Huffel, "Overview of total least-squares methods," *Signal processing*, vol. 87, no. 10, pp. 2283–2302, 2007.

[129] S. Gupta and I. Dokmanić, "Total least squares phase retrieval," *IEEE Transactions on Signal Processing*, vol. 70, pp. 536–549, 2021.

[130] A. C. Kak and M. Slaney, *Principles of computerized tomographic imaging*. SIAM, 2001.

[131] L. De Chiffre, S. Carmignato, J.-P. Kruth, R. Schmitt, and A. Weckenmann, "Industrial applications of computed tomography," *CIRP annals*, vol. 63, no. 2, pp. 655–677, 2014.

[132] K. Wells and D. Bradley, "A review of x-ray explosives detection techniques for checked baggage," *Applied Radiation and Isotopes*, vol. 70, no. 8, pp. 1729–1746, 2012.

[133] S. Helgason, "The radon transform on euclidean spaces, compact two-point homogeneous spaces and grassmann manifolds," *Acta Mathematica*, vol. 113, no. 1, pp. 153–180, 1965.

[134] S. Helgason, "Support of radon transforms," *Advances in Mathematics*, vol. 38, no. 1, pp. 91–100, 1980.

[135] J. Boman and E. T. Quinto, "Support theorems for real-analytic radon transforms," 1987.

[136] J. Boman, "Helgason's support theorem for radon transforms—a new proof and a generalization," in *Mathematical Methods in Tomography: Proceedings of a Conference held in Oberwolfach, Germany, 5–11 June, 1990*, pp. 1–5, Springer, 2006.

[137] J. Leuschner, M. Schmidt, D. O. Baguer, and P. Maass, "Lodopab-ct, a benchmark dataset for low-dose computed tomography reconstruction," *Scientific Data*, vol. 8, no. 1, p. 109, 2021.

[138] M. Hssayeni, M. Croock, A. Salman, H. Al-khafaji, Z. Yahya, and B. Ghoraani, "Computed tomography images for intracranial hemorrhage detection and segmentation," *Intracranial Hemorrhage Segmentation Using A Deep Convolutional Model. Data*, vol. 5, no. 1, p. 14, 2020.

[139] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *International Conference on Learning Representations*, 2017.

[140] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. Susano Pinto, D. Keysers, and N. Houlsby, "Scaling vision with sparse mixture of experts," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8583–8595, 2021.

[141] W. Fedus, J. Dean, and B. Zoph, "A review of sparse expert models in deep learning," *arXiv preprint arXiv:2209.01667*, 2022.

[142] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[143] A. P. Jathoul, J. Laufer, O. Ogunlade, B. Treeby, B. Cox, E. Zhang, P. Johnson, A. R. Pizzey, B. Philip, T. Marafioti, *et al.*, "Deep in vivo photoacoustic imaging of mammalian tissues using a tyrosinase-based genetic reporter," *Nature Photonics*, vol. 9, no. 4, pp. 239–246, 2015.

[144] J. Yao, L. Wang, J.-M. Yang, K. I. Maslov, T. T. Wong, L. Li, C.-H. Huang, J. Zou, and L. V. Wang, "High-speed label-free functional photoacoustic microscopy of mouse brain in action," *Nature methods*, vol. 12, no. 5, pp. 407–410, 2015.

[145] A. Doerr, "Cryo-electron tomography," *Nature Methods*, vol. 14, no. 1, pp. 34–34, 2017.

[146] V. Debarnot, V. Kishore, R. D. Righetto, and I. Dokmanić, "Ice-tide: Implicit cryo-et imaging and deformation estimation," *arXiv preprint arXiv:2403.02182*, 2024.

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

[148] 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*, pp. 4379–4388, 2021.

[149] V. Shah and C. Hegde, "Solving linear inverse problems using gan priors: An algorithm with provable guarantees," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4609–4613, IEEE, 2018.

[150] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*, pp. 1530–1538, PMLR, 2015.

[151] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020.

[152] J. N. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein, "Acorn: Adaptive coordinate networks for neural scene representation," *arXiv preprint arXiv:2105.02788*, 2021.

[153] V. Saragadam, J. Tan, G. Balakrishnan, R. G. Baraniuk, and A. Veeraraghavan, "Miner: Multiscale implicit neural representations," *arXiv preprint arXiv:2202.03532*, 2022.

[154] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

[155] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4460–4470, 2019.

[156] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[157] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

[158] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[159] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

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

[161] J. Brehmer and K. Cranmer, "Flows for simultaneous manifold learning and density estimation," *arXiv preprint arXiv:2003.13913*, 2020.

[162] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*, pp. 694–711, Springer, 2016.

[163] K. J. Bergen, P. A. Johnson, M. V. de Hoop, and G. C. Beroza, "Machine learning for data-driven discovery in solid earth geoscience," *Science*, vol. 363, no. 6433, p. eaau0323, 2019.

[164] D. Lee, J. Yoo, and J. C. Ye, "Deep residual learning for compressed sensing MRI," in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pp. 15–18, 2017.

[165] H. Arabi and H. Zaidi, "Applications of artificial intelligence and deep learning in molecular imaging and radiotherapy," *European Journal of Hybrid Imaging*, vol. 4, no. 1, pp. 1–23, 2020.

[166] S. A. Hussein, T. Tirer, and R. Giryes, "Image-adaptive gan based reconstruction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 3121–3129, 2020.

[167] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[168] P. Weiss, P. Escande, G. Bathie, and Y. Dong, "Contrast invariant snr and isotonic regressions," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 1144–1161, 2019.

[169] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE transactions on image processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[170] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3791–3799, 2015.

[171] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*, pp. 184–199, Springer, 2014.

[172] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European conference on computer vision*, pp. 391–407, Springer, 2016.

[173] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1646–1654, 2016.

[174] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 136–144, 2017.

[175] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 624–632, 2017.

[176] X. Jia, H. Chang, and T. Tuytelaars, "Super-resolution with deep adaptive image resampling," *arXiv preprint arXiv:1712.06463*, 2017.

[177] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

[178] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0, 2018.

[179] A. Lugmayr, M. Danelljan, L. V. Gool, and R. Timofte, "Srflow: Learning the super-resolution space with normalizing flow," in *European conference on computer vision*, pp. 715–732, Springer, 2020.

[180] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, "Meta-sr: A magnification-arbitrary network for super-resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1575–1584, 2019.

[181] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[182] R. Xu, X. Wang, K. Chen, B. Zhou, and C. C. Loy, "Positional encoding as spatial inductive bias in gans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13569–13578, 2021.

[183] E. Ntavelis, M. Shahbazi, I. Kastanis, R. Timofte, M. Danelljan, and L. Van Gool, "Arbitrary-scale image synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11533–11542, 2022.

[184] I. Skorokhodov, S. Ignatyev, and M. Elhoseiny, "Adversarial generation of continuous images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10753–10764, 2021.

[185] I. Anokhin, K. Demochkin, T. Khakhulin, G. Sterkin, V. Lempitsky, and D. Korzhenkov, "Image generators with conditionally-independent pixel synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14278–14287, 2021.

[186] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation.," *J. Mach. Learn. Res.*, vol. 23, pp. 47–1, 2022.

[187] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[188] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[189] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[190] H. Thanh-Tung and T. Tran, "Catastrophic forgetting and mode collapse in gans," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10, IEEE, 2020.

[191] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *International Conference on Learning Representations, ICLR*, 2017.

[192] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," in *International Conference on Learning Representations, ICLR*, 2015.

[193] M. Puthawala, K. Kothari, M. Lassas, I. Dokmanić, and M. de Hoop, "Globally injective relu networks," *arXiv preprint arXiv:2006.08464*, 2020.

[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*, vol. 35, pp. 2628–2637, 2021.

[195] K. Mosegaard and A. Tarantola, "Monte carlo sampling of solutions to inverse problems," *Journal of Geophysical Research: Solid Earth*, vol. 100, no. B7, pp. 12431–12447, 1995.

[196] F. Monard, R. Nickl, and G. P. Paternain, "Consistent inversion of noisy non-abelian x-ray transforms," *Communications on Pure and Applied Mathematics*, 2020.

[197] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[198] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.

[199] W. M. Boothby, *An introduction to differentiable manifolds and Riemannian geometry*. Academic press, 1986.

[200] J.-H. Jacobsen, A. Smeulders, and E. Oyallon, "i-revnet: Deep invertible networks," *arXiv preprint arXiv:1802.07088*, 2018.

[201] J. Whang, Q. Lei, and A. Dimakis, "Compressed sensing with invertible generative models and dependent noise," in *NeurIPS 2020 Workshop on Deep Learning and Inverse Problems*, 2020.

[202] M. F. Hutchinson, "A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines," *Communications in Statistics-Simulation and Computation*, vol. 18, no. 3, pp. 1059–1076, 1989.

[203] R. T. Chen, J. Behrmann, D. Duvenaud, and J.-H. Jacobsen, "Residual flows for invertible generative modeling," *arXiv preprint arXiv:1906.02735*, 2019.

[204] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[205] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[206] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

[207] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2097–2106, 2017.

[208] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *arXiv preprint arXiv:1706.08500*, 2017.

[209] A. Kumar, B. Poole, and K. Murphy, "Regularized autoencoders via relaxed injective probability flow," in *International Conference on Artificial Intelligence and Statistics*, pp. 4292–4301, PMLR, 2020.

[210] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9446–9454, 2018.

[211] A. Raj, Y. Li, and Y. Bresler, "Gan-based projector for faster recovery with convergence guarantees in linear inverse problems," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5602–5611, 2019.

[212] E. Cunningham, R. Zabounidis, A. Agrawal, I. Fiterau, and D. Sheldon, "Normalizing flows across dimensions," *arXiv preprint arXiv:2006.13070*, 2020.

[213] K. Flouris and E. Konukoglu, "Canonical normalizing flows for manifold learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 27294–27314, 2023.

[214] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

[215] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improving variational inference with inverse autoregressive flow.," *URL http://arxiv. org/abs/1606.04934*, 2016.

[216] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 1747–1756, PMLR, 20–22 Jun 2016.

[217] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural spline flows," *arXiv preprint arXiv:1906.04032*, 2019.

[218] T. Teshima, I. Ishikawa, K. Tojo, K. Oono, M. Ikeda, and M. Sugiyama, "Coupling-based invertible neural networks are universal diffeomorphism approximators," *arXiv preprint arXiv:2006.11469*, 2020.

[219] A. Pardo, S. S. Streeter, B. W. Maloney, J. A. Gutiérrez-Gutiérrez, D. M. McClatchy, W. A. Wells, K. D. Paulsen, J. M. López-Higuera, B. W. Pogue, and O. M. Conde, "Modeling and synthesis of breast cancer optical property signatures with generative models," *IEEE Transactions on medical imaging*, vol. 40, no. 6, pp. 1687–1701, 2021.

[220] Q. Dai, Y. H. Lee, H.-H. Sun, G. Ow, M. L. M. Yusof, and A. C. Yucel, "3din-vnet: A deep learning-based 3d ground-penetrating radar data inversion," *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

[221] Q. Cao, I. L. Al-Qadi, and L. Abufares, "Pavement moisture content prediction: A deep residual neural network approach for analyzing ground penetrating radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–11, 2022.

[222] V. Khorashadi-Zadeh and M. Dehmollaian, "Through a cinder block wall refocusing using sar back projection method," *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 2, pp. 1212–1222, 2018.

[223] J. Song, H. Chen, C. Du, and J. Li, "Semi-mapgen: Translation of remote sensing image into map via semisupervised adversarial learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–19, 2023.

[224] A. I. Nachman, "Global uniqueness for a two-dimensional inverse boundary value problem," *Annals of Mathematics*, pp. 71–96, 1996.

[225] Y. Wang and W. C. Chew, "An iterative solution of the two-dimensional electromagnetic inverse scattering problem," *International Journal of Imaging Systems and Technology*, vol. 1, no. 1, pp. 100–108, 1989.

[226] W. C. Chew and Y.-M. Wang, "Reconstruction of two-dimensional permittivity distribution using the distorted born iterative method," *IEEE transactions on medical imaging*, vol. 9, no. 2, pp. 218–225, 1990.

[227] P. M. Van Den Berg and R. E. Kleinman, "A contrast source inversion method," *Inverse problems*, vol. 13, no. 6, p. 1607, 1997.

[228] X. Chen, "Subspace-based optimization method for solving inverse-scattering problems," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 1, pp. 42–49, 2009.

[229] Y. Khoo and L. Ying, "Switchnet: a neural network model for forward and inverse scattering problems," *SIAM Journal on Scientific Computing*, vol. 41, no. 5, pp. A3182–A3201, 2019.

[230] P. Ran, Y. Qin, and D. Lesselier, "Electromagnetic imaging of a dielectric microstructure via convolutional neural networks," in *2019 27th European Signal Processing Conference (EUSIPCO)*, pp. 1–5, IEEE, 2019.

[231] L. Li, L. G. Wang, F. L. Teixeira, C. Liu, A. Nehorai, and T. J. Cui, "DeepNIS: Deep neural network for nonlinear electromagnetic inverse scattering," *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 3, pp. 1819–1825, 2018.

[232] J. E. Fajardo, J. Galván, F. Vericat, C. M. Carlevaro, and R. M. Irastorza, "Phaseless microwave imaging of dielectric cylinders: An artificial neural networks-based approach," *Progress In Electromagnetics Research*, vol. 166, pp. 95–105, 2019.

[233] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *International Conference on Learning Representations, ICLR*, 2018.

[234] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen, "On instabilities of deep learning in image reconstruction and the potential costs of AI," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30088–30095, 2020.

[235] X. Chen, Z. Wei, M. Li, and P. Rocca, "A review of deep learning approaches for inverse scattering problems (invited review)," *Progress In Electromagnetics Research*, vol. 167, pp. 67–81, 2020.

[236] J. Fei, Y. Chen, M. Zhong, and F. Han, "Fast 3-d electromagnetic full-wave inversion of dielectric anisotropic objects based on resu-net enhanced by variational born iterative method," *IEEE Transactions on Antennas and Propagation*, vol. 70, no. 8, pp. 6229–6239, 2022.

[237] T. Shan, Z. Lin, X. Song, M. Li, F. Yang, and S. Xu, "Neural born iterative method for solving inverse scattering problems: 2d cases," *IEEE Transactions on Antennas and Propagation*, 2022.

[238] H. Zhou, Y. Cheng, H. Zheng, Q. Liu, and Y. Wang, "Deep unfolding contrast source inversion for strong scatterers via generative adversarial mechanism," *IEEE Transactions on Microwave Theory and Techniques*, vol. 70, no. 11, pp. 4966–4979, 2022.

[239] Y. Liu, H. Zhao, R. Song, X. Chen, C. Li, and X. Chen, "Som-net: Unrolling the subspace-based optimization for solving full-wave inverse scattering problems," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.

[240] R. Guo, Z. Lin, T. Shan, X. Song, M. Li, F. Yang, S. Xu, and A. Abubakar, "Physics embedded deep neural network for solving full-wave inverse scattering problems," *IEEE transactions on antennas and propagation*, vol. 70, no. 8, pp. 6148–6159, 2021.

[241] V. A. Kelkar and M. Anastasio, "Prior image-constrained reconstruction using style-based generative models," in *International Conference on Machine Learning*, pp. 5367–5377, PMLR, 2021.

[242] B. Ross and J. Cresswell, "Tractable density estimation on learned manifolds with conformal embedding flows," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[243] R. Guo, Z. Lin, M. Li, F. Yang, S. Xu, and A. Abubakar, "A nonlinear model compression scheme based on variational autoencoder for microwave data inversion," *IEEE Transactions on Antennas and Propagation*, vol. 70, no. 11, pp. 11059–11069, 2022.

[244] Z. Wei and X. Chen, "Uncertainty quantification in inverse scattering problems with Bayesian convolutional neural networks," *IEEE Transactions on Antennas and Propagation*, vol. 69, no. 6, pp. 3409–3418, 2020.

[245] S. He, G. Zhang, and Z. Wei, "Uncertainty calibrations of deep-learning schemes for full-wave inverse scattering problems," *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

[246] P. Y. Chen, D. J. Bergman, and Y. Sivan, "Spectral decomposition of the lippmann-schwinger equation applied to cylinders," *arXiv preprint arXiv:1705.01747*, 2017.

[247] S. R. Rengarajan and Y. Rahmat-Samii, "The field equivalence principle: Illustration of the establishment of the non-intuitive null fields," *IEEE Antennas and Propagation Magazine*, vol. 42, no. 4, pp. 122–128, 2000.

[248] K. Kothari, M. de Hoop, and I. Dokmanić, "Learning the geometry of wave-based imaging," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 8318–8329, Curran Associates, Inc., 2020.

[249] P. M. van den Berg, A. Van Broekhoven, and A. Abubakar, "Extended contrast source inversion," *Inverse problems*, vol. 15, no. 5, p. 1325, 1999.

[250] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13, 1993.

[251] A. Graves, "Practical variational inference for neural networks," *Advances in neural information processing systems*, vol. 24, pp. 2348–2356, 2011.

[252] J. Whang, E. Lindgren, and A. Dimakis, "Composing normalizing flows for inverse problems," in *International Conference on Machine Learning*, pp. 11158–11169, PMLR, 2021.

[253] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*, pp. 1278–1286, PMLR, 2014.

[254] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[255] J.-M. Geffrin, P. Sabouroux, and C. Eyraud, "Free space experimental scattering database continuation: experimental set-up and measurement precision," *inverse Problems*, vol. 21, no. 6, p. S117, 2005.

[256] F. Vasconcelos, B. He, N. Singh, and Y. W. Teh, "UncertaINR: Uncertainty Quantification of End-to-End Implicit Neural Representations for Computed Tomography," *arXiv preprint arXiv:2202.10847*, 2022.

[257] M.-H. Chen, Q.-M. Shao, and J. G. Ibrahim, *Monte Carlo methods in Bayesian computation*. Springer Science & Business Media, 2012.

[258] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[259] J. Marino, Y. Yue, and S. Mandt, "Iterative Amortized Inference," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 3403–3412, PMLR, 10–15 Jul 2018.

[260] L. Ardizzone, J. Kruse, C. Lüth, N. Bracher, C. Rother, and U. Köthe, "Conditional invertible neural networks for diverse image-to-image translation," in *Pattern Recognition: 42nd DAGM German Conference, DAGM GCPR 2020, Tübingen, Germany, September 28–October 1, 2020, Proceedings 42*, pp. 373–387, Springer, 2021.

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

[262] G. A. Padmanabha and N. Zabaras, "Solving inverse problems using conditional invertible neural networks," *Journal of Computational Physics*, vol. 433, p. 110194, 2021.

[263] R. Gribonval, "Should penalized least squares regression be interpreted as maximum a posteriori estimation?," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2405–2410, 2011.

[264] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, "Amortised map inference for image super-resolution," *International Conference on Learning Representations, ICLR*, 2017.

[265] N. Courts and H. Kvinge, "Bundle networks: Fiber bundles, local trivializations, and a generative approach to exploring many-to-one maps," *International Conference on Learning Representations, ICLR*, 2022.

[266] Y. Fan and L. Ying, "Solving inverse wave scattering with deep learning," *Annals of Mathematical Sciences and Applications*, pp. 23–48, 2022.

[267] M. Asim, M. Daniels, O. Leong, A. Ahmed, and P. Hand, "Invertible generative models for inverse problems: mitigating representation error and dataset bias," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 399–409, PMLR, 13–18 Jul 2020.

[268] A. Repetti, M. Pereyra, and Y. Wiaux, "Scalable Bayesian uncertainty quantification in imaging inverse problems via convex optimization," *SIAM Journal on Imaging Sciences*, vol. 12, no. 1, pp. 87–118, 2019.

[269] P. Bohra, T.-a. Pham, J. Dong, and M. Unser, "Bayesian Inversion for Nonlinear Imaging Models using Deep Generative Priors," *arXiv preprint arXiv:2203.10078*, 2022.

[270] K. C. Tezcan, N. Karani, C. F. Baumgartner, and E. Konukoglu, "Sampling possible reconstructions of undersampled acquisitions in mr imaging with a deep learned prior," *IEEE Transactions on Medical Imaging*, vol. 41, no. 7, pp. 1885–1896, 2022.

[271] S. Bhadra, U. Villa, and M. A. Anastasio, "Mining the manifolds of deep generative models for multiple data-consistent solutions of ill-posed tomographic imaging problems," *arXiv preprint arXiv:2202.05311*, 2022.

[272] R. V. Marinescu, D. Moyer, and P. Golland, "Bayesian image reconstruction using deep generative models," *arXiv preprint arXiv:2012.04567*, 2020.

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

[274] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, pp. 3483–3491, 2015.

[275] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance weighted autoencoders," *International Conference on Learning Representations*, 2015.

[276] Y. Lu and B. Huang, "Structured output learning with conditional generative flows," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5005–5012, 2020.

[277] M. Sorkhei, G. E. Henter, and H. Kjellström, "Full-glow: Fully conditional glow for more realistic image generation," in *Pattern Recognition: 43rd DAGM German Conference, DAGM GCPR 2021, Bonn, Germany, September 28–October 1, 2021, Proceedings*, pp. 697–711, Springer, 2022.

[278] A. Pumarola, S. Popov, F. Moreno-Noguer, and V. Ferrari, "C-flow: Conditional generative flow models for images and 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7949–7958, 2020.

[279] M. Puthawala, M. Lassas, I. Dokmanic, and M. De Hoop, "Universal joint approximation of manifolds and densities by simple injective flows," in *International Conference on Machine Learning*, pp. 17959–17983, PMLR, 2022.

[280] K. Wang, T. Bui-Thanh, and O. Ghattas, "A randomized maximum a posteriori method for posterior sampling of high dimensional nonlinear Bayesian inverse problems," *SIAM Journal on Scientific Computing*, vol. 40, no. 1, pp. A142–A171, 2018.

[281] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference.," *J. Mach. Learn. Res.*, vol. 22, no. 57, pp. 1–64, 2021.

[282] K. C. Tezcan, C. F. Baumgartner, R. Luechinger, K. P. Pruessmann, and E. Konukoglu, "Mr image reconstruction using deep density priors," *IEEE transactions on medical imaging*, vol. 38, no. 7, pp. 1633–1642, 2018.

[283] T. P. Minka, "Expectation propagation for approximate bayesian inference," in *Uncertainty in Artificial Intelligence*, p. 362—369, PMLR, 2001.

[284] T. P. Minka, *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[285] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[286] B. C. Brown, A. L. Caterini, B. L. Ross, J. C. Cresswell, and G. Loaiza-Ganem, "Verifying the union of manifolds hypothesis for image data," in *The Eleventh International Conference on Learning Representations*, 2023.

[287] Y. Choquet-Bruhat and C. DeWitt-Morette, *Analysis, Manifolds and Physics Revised Edition*. Gulf Professional Publishing, 1982.

[288] Y. Fan and Z. Zhao, "Cryo-electron microscopy image denoising using multi-frequency vector diffusion maps," in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 3463–3467, IEEE, 2021.

[289] V. Debarnot, V. Kishore, C. Shi, and I. Dokmanić, "Manifold rewiring for unlabeled imaging," *arXiv preprint arXiv:2209.05168*, 2022.

[290] S. Gupta, K. Kothari, M. V. de Hoop, and I. Dokmanić, "Random mesh projectors for inverse problems," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[291] E. Venkatesh and S. V. Elluru, "Cone beam computed tomography: basics and applications in dentistry," *Journal of istanbul University faculty of Dentistry*, vol. 51, no. 3 Suppl 1, p. S102, 2017.

[292] K. Belkebir, P. C. Chaumet, and A. Sentenac, "Superresolution in total internal reflection tomography," *JOSA A*, vol. 22, no. 9, pp. 1889–1897, 2005.

[293] M. Zhang, Y. Sun, S. McDonagh, and C. Zhang, "Flow based models for manifold data," *arXiv preprint arXiv:2109.14216*, 2021.

[294] A. Siahkoohi, G. Rizzuti, M. Louboutin, P. A. Witte, and F. J. Herrmann, "Preconditioned training of normalizing flows for variational inference in inverse problems," *arXiv preprint arXiv:2101.03709*, 2021.

[295] A. Siahkoohi, G. Rizzuti, R. Orozco, and F. J. Herrmann, "Reliable amortized variational inference with physics-based latent distribution correction," *Geophysics*, vol. 88, no. 3, pp. 1–137, 2023.

[296] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Interspeech*, 2017.

[297] E. Begoli, T. Bhattacharya, and D. Kusnezov, "The need for uncertainty quantification in machine-assisted medical decision making," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 20–23, 2019.

[298] M. Stoyer, D. McNabb, J. Burke, and L. Bernstein, "Science based stockpile stewardship, uncertainty quantification, and surrogate reactions," tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2009.

[299] D. Brown, M. Herman, S. Hoblit, E. McCutchan, G. Nobre, B. Pritychenko, and A. Sonzogni, "Uncertainty quantification in the nuclear data program," *Journal of Physics G: nuclear and particle physics*, vol. 42, no. 3, p. 034020, 2015.

[300] F. Arnez, H. Espinoza, A. Radermacher, and F. Terrier, "A comparison of uncertainty estimation approaches in deep learning components for autonomous vehicle applications," *arXiv preprint arXiv:2006.15172*, 2020.

[301] R. Michelmore, M. Kwiatkowska, and Y. Gal, "Evaluating uncertainty quantification in end-to-end autonomous driving control," *arXiv preprint arXiv:1811.06817*, 2018.

[302] A. Tarantola, *Inverse problem theory and methods for model parameter estimation*. SIAM, 2005.

[303] A. M. Stuart, "Inverse problems: a Bayesian perspective," *Acta numerica*, vol. 19, pp. 451–559, 2010.

[304] J. Martin, L. C. Wilcox, C. Burstedde, and O. Ghattas, "A stochastic newton mcmc method for large-scale statistical inverse problems with application to seismic inversion," *SIAM Journal on Scientific Computing*, vol. 34, no. 3, pp. A1460–A1487, 2012.

[305] Z. Zhao and M. K. Sen, "A gradient based mcmc method for fwi and uncertainty analysis," in *SEG International Exposition and Annual Meeting*, OnePetro, 2019.

[306] T. Cui, Y. M. Marzouk, and K. E. Willcox, "Data-driven model reduction for the Bayesian solution of inverse problems," *International Journal for Numerical Methods in Engineering*, vol. 102, no. 5, pp. 966–990, 2015.

[307] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of multifidelity methods in uncertainty propagation, inference, and optimization," *Siam Review*, vol. 60, no. 3, pp. 550–591, 2018.

[308] P. Esser, E. Sutter, and B. Ommer, "A variational u-net for conditional appearance and shape generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8857–8866, 2018.

[309] L. Jin, H. Lu, and G. Wen, "Fast uncertainty quantification of reservoir simulation with variational u-net," *arXiv preprint arXiv:1907.00718*, 2019.

[310] C. F. Baumgartner, K. C. Tezcan, K. Chaitanya, A. M. Hötker, U. J. Muehlematter, K. Schawkat, A. S. Becker, O. Donati, and E. Konukoglu, "Phiseg: Capturing uncertainty in medical image segmentation," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part II 22*, pp. 119–127, Springer, 2019.

[311] A. Siahkoohi, G. Rizzuti, and F. J. Herrmann, "Deep Bayesian inference for seismic imaging with tasks," *Geophysics*, vol. 87, no. 5, pp. S281–S302, 2022.

[312] A. Stanziola, S. R. Arridge, B. T. Cox, and B. E. Treeby, "j-wave: An open-source differentiable wave simulator," *arXiv preprint arXiv:2207.01499*, 2022.

[313] A. Bermúdez, L. Hervella-Nieto, A. Prieto, R. Rodrı, *et al.*, "An optimal perfectly matched layer with unbounded absorbing function for time-harmonic acoustic scattering problems," *Journal of computational Physics*, vol. 223, no. 2, pp. 469–488, 2007.

[314] 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.

[315] C. Donnelly, S. Gliga, V. Scagnoli, M. Holler, J. Raabe, L. J. Heyderman, and M. Guizar-Sicairos, "Tomographic reconstruction of a three-dimensional magnetization vector field," *New Journal of Physics*, vol. 20, no. 8, p. 083009, 2018.

[316] M. Holler, M. Odstrcil, M. Guizar-Sicairos, M. Lebugle, E. Müller, S. Finizio, G. Tinti, C. David, J. Zusman, W. Unglaub, *et al.*, "Three-dimensional imaging of integrated circuits with macro-to nanoscale zoom," *Nature Electronics*, vol. 2, no. 10, pp. 464–470, 2019.