# A dimension-reduced variational approach for solving physics-based inverse problems using generative adversarial network priors and normalizing flows

Agnimitra Dasgupta[a], Dhruv V Patel[b], Deep Ray[c,d], Erik A Johnson[e], Assad A Oberai[a,*]

[a]*Department of Aerospace & Mechanical Engineering, University of Southern California, Los Angeles, 90089, California, USA*
[b]*Department of Mechanical Engineering, Stanford University, Stanford, 94305, California, USA*
[c]*Department of Mathematics, University of Maryland, College Park, 20742, Maryland, USA*
[d]*Institute for Physical Science and Technology, University of Maryland, College Park, 20742, Maryland, USA*
[e]*Sonny Astani Department of Civil & Environmental Engineering, University of Southern California, Los Angeles, 90089, California, USA*

**Abstract**

We propose a novel modular inference approach combining two different generative models — generative adversarial networks (GAN) and normalizing flows — to approximate the posterior distribution of physics-based Bayesian inverse problems framed in high-dimensional ambient spaces. We dub the proposed framework GAN-Flow. The proposed method leverages the intrinsic dimension reduction and superior sample generation capabilities of GANs to define a low-dimensional data-driven prior distribution. Once a trained GAN-prior is available, the inverse problem is solved entirely in the latent space of the GAN using variational Bayesian inference with normalizing flow-based variational distribution, which approximates low-dimensional posterior distribution by transforming realizations from the low-dimensional latent prior (Gaussian) to corresponding realizations of a low-dimensional variational posterior distribution. The trained GAN generator then maps realizations from this approximate posterior distribution in the latent space back to the high-dimensional ambient space. We also propose a two-stage training strategy for GAN-Flow wherein we train the two generative models sequentially. Thereafter, GAN-Flow can estimate the statistics of posterior-predictive quantities of interest at virtually no additional computational cost. The synergy between the two types of generative models allows us to overcome many challenges associated with the application of Bayesian inference to large-scale inverse problems, chief among which are describing an informative prior and sampling from the high-

dimensional posterior. GAN-Flow does not involve Markov chain Monte Carlo simulation, making it particularly suitable for solving large-scale inverse problems. We demonstrate the efficacy and flexibility of GAN-Flow on various physics-based inverse problems of varying ambient dimensionality and prior knowledge using different types of GANs and normalizing flows. Notably, one of the applications we consider involves a 65,536-dimensional inverse problem of phase retrieval wherein an object is reconstructed from sparse noisy measurements of the magnitude of its Fourier transform.
*Keywords:* Inverse problems, Bayesian inference, variational inference, generative modeling, uncertainty quantification

## 1. Introduction

Inverse problems are useful for determining the causal factors behind an observed phenomenon but remain challenging to solve. Inverse problems are ill-posed and, as such, they may admit multiple or, in the extreme case, no solutions [1]. Moreover, in most practical applications, the forward problem is nonlinear, the inferred quantity is high-dimensional, and measurements are noisy: all of these factors makes it challenging to solve inverse problems. Deterministic approaches to solving inverse problems result in point estimates, i.e., a single solution to the inverse problem at hand, which precludes other possible solutions. In contrast, the Bayesian paradigm treats inverse problems in a stochastic setting, with the *posterior* distribution characterizing all possible solutions to the inverse problem at hand. Using Bayes' rule, the posterior distribution results from updating the *prior* distribution through a *likelihood* function: this process is known as Bayesian inference. The posterior distribution is also useful for quantifying the relative plausibility of different solutions, popularly known as uncertainty quantification. Bayesian inference is attractive because it is philosophically appealing and conceptually simple while giving additional useful information about uncertainty in the solution. However, the application of Bayesian inference poses many practical and computational challenges. On a practical note, selecting a well-informed prior distribution is

*Corresponding author
*Email address:* aoberai@usc.edu (Assad A Oberai)

crucial to the success of Bayesian inference. However, the task of choosing such a distribution capable of accommodating the myriad of variations encountered in practical solution fields is far from straightforward. Even with a carefully designed prior, Bayesian inference remains computationally challenging because only a few cases permit closed-form posterior distributions, i.e., those where the prior and likelihood distributions form a conjugate pair. Unfortunately, scenarios where it is possible to leverage Bayesian conjugacy seldom occur, and the posterior distribution must be approximated using appropriate tools.

One way of approximating the posterior distribution is through samples drawn from it. Markov chain Monte Carlo (MCMC) methods have been the workhorse of posterior sampling for almost half a century [2]. However, the application of MCMC can be challenging on large-scale inverse problems, i.e., when the inferred quantity is high-dimensional — this is popularly known as the '*curse of dimensionality*'. The difficulty manifests as long mixing times and larger autocorrelations between successive samples of Markov chains [3, 4]. Many notable advancements have been proposed to improve the performance of MCMC methods. Some advanced methods focus on carefully designing the proposal distributions in high dimensions to reduce mixing times [5, 6]. Alternatively, some approaches try to reduce the stochastic dimensionality of the inverse problem [3, 7]. Despite these advancements, the application of MCMC to large-scale inverse problems continues to present significant challenges.

Variational inference is often considered a computationally efficient alternative approach for approximating the posterior distribution [8]. In this approach, a parameterized family of distributions that permit efficient sampling and density evaluations are used to approximate the posterior distribution. The optimal parameters are chosen by minimizing some measure of divergence between the approximate posterior density induced by the adopted distribution family and the true posterior density. Choosing an expressive approximation family for the posterior distribution is critical to the success of variational inference, and doing so is difficult in high dimensions where very little information is available about the shape of the posterior. Recently, transport maps have emerged as a popular choice in this regard [9]. Instead of approximating the posterior distribution using a parameterized family of distributions, transport maps mold the prior distribution into the posterior distribution. Thus, instead of optimizing the parameters of a family of distributions, the

parameters of the transport map must be optimized. While variational inference has been shown to scale better than MCMC [9, 10], the curse of dimensionality is still a challenge. The number of parameters that must be optimized, when approximating posterior distributions, proliferates as the dimensionality of the inverse problem increases. Similarly, the construction of high-dimensional transport maps can be difficult [11].

More recently, deep learning models capable of carrying out Bayesian inference while meeting or circumventing the challenges posed by Bayesian inference are gaining popularity [12]. In particular, deep generative models are at the forefront of deep learning-driven Bayesian inference. Also popular are the conditional counterparts of deep generative models trained using supervised data: given realizations from the prior distribution, synthetic measurements are generated using the forward model, and the training data consists of pairs of the prior realizations and corresponding measurements. In such a supervised setting, conditional generative models can be used to obtain realizations from the posterior distribution for any new measurement. Some popular deep generative models are generative adversarial networks (GANs) [13], normalizing flows [14, 15, 16], and variational auto-encoders [17]. Among them, GANs possess superior sample generation qualities and intrinsic dimension reduction capabilities [18]. As a result, GANs have been used as a data informative priors [19, 20]. Conditional GANs have also been used to approximate the posterior distribution [21, 22, 23]. However, GANs remain notoriously difficult to train and susceptible to mode collapse. Moreover, GANs are implicit generative models i.e., it is not possible to evaluate point-wise the probability density induced by a GAN. In contrast, variational auto-encoders allow for the computation of a lower bound on point-wise density values. Variational auto-encoders have also been used to perform Bayesian inference [24, 25, 26], but tend to produce blurry outputs compared to those from GANs. In contrast to GANs and variational auto-encoders, normalizing flows are explicit generative models and allow for point-wise evaluation of the probability density they induce [27]. Normalizing flows utilize invertible neural networks to construct a bijective transformation [15, 16]. Therefore, normalizing flows are natural candidates for transport maps and have been used for variational inference [14, 28, 29]. Conditional normalizing flows have also been used to approximate the posterior distribution in inverse problems [30, 22, 23]. However, the application of normalizing flows to large-scale inverse problems is challenging: when the in-

verse problem at hand is high-dimensional, the memory footprint of high-dimensional normalizing flows is so large that training requires access to extraordinary computational resources [31] (for instance, see [32] where a GLOW normalizing flow model is trained with a mini-batch size of 1 per processing unit which approximately amounts to 40 GPU weeks). Aside from deep generative models, Bayesian neural networks have also been used for Bayesian inference [33]. Bayesian neural networks implicitly learn to invert measurements by introducing stochasticity in the weights of a neural network. However, Bayesian neural networks have limited capacity due to the approximations made to make the training tractable [34]. We note that the supervised datasets necessary to train many of the aforementioned models may not be readily available and computationally expensive to acquire.

In this work, we propose a modular unsupervised inference framework — GAN-Flow — that couples together GANs and normalizing flows to solve large-scale physics-based inverse problems when the only prior information available is a sample from the true but inaccessible prior distribution. GAN-Flow aims to circumvent the challenges faced by generative models when they are used to perform inference in high-dimensional settings by exploiting the respective strengths of the two types of generative models it employs: the dimension reduction capability of GANs, and the efficient variational inference capability of normalizing flows. More specifically, GAN-Flow employs a Wasserstein GAN (WGAN) to learn a data-driven prior distribution which will be useful for Bayesian inference. Further, the WGAN helps reduce the dimensionality of the inverse problem as the generator component of the GAN serves as an injective map from the low-dimensional latent space to the high-dimensional ambient space where the inverse problem is framed. Thus, GAN-Flow also leverages the dimension reduction offered by GANs. Recent findings suggest that framing inverse problems in lower-dimensional latent spaces may be advantageous [35]. GAN-Flow also utilizes normalizing flows to approximate the posterior distribution in the latent space. Again, the dimension reduction capability of GANs facilitate the construction of simpler normalizing flow models, which now only need to perform variational inference in the lower-dimensional latent space, thereby reducing the memory footprint of normalizing flow models. As a result, GAN-Flow can be used to tackle large-scale inverse problems. The use of a normalizing flow, which serves as a map between the latent prior and the latent posterior, offers one significant

advantage: once the normalizing flow is trained, new samples from the latent posterior can be efficiently generated without taking recourse to the computationally expensive forward model. A new sample from the high-dimensional posterior is generated by successive transformations of a sample from the latent space of the GAN using the trained normalizing flow model followed by the trained generator. In this work, we demonstrate the wide applicability and flexibility of the proposed framework on different large-scale linear and nonlinear inverse problems, different synthetic (simple geometrical features and Shepp-Logan phantoms [36]) and real-world (MRI scans of human knees [37, 38]) prior distributions, different GAN architectures (self-attention GANs [39] and GANs that progressively grow [40]), and normalizing flows with different invertible neural network architectures (planar [14] and affine-coupling flows [41, 42]).

We must mention a growing body of work that has developed deep learning-based Bayesian inference frameworks with some dimension reduction component. For instance, this work was directly inspired by [35, 19], where GANs are used to learn prior distributions and MCMC methods are used to sample from its posterior. Patel and Oberai [35] used the Hamiltonian MCMC, whereas Bohra et al. [20] used the Metropolis-adjusted Langevin MCMC. However, posterior sampling using an MCMC-based method will fail when the latent space dimensionality continues to be high (for example see Section 4.3). Additionally, it is non-trivial to ascertain the convergence of MCMC chains. In contrast, one can easily gauge the convergence of the latent posterior induced by the normalizing flow model by tracking the loss function used to train the normalizing flow model. Moreover, for a fixed compute budget (as defined by the number of forward problem solves), training the normalizing flow model requires less compute wall times since it is possible to train it using mini-batches. Similarly, sampling is also embarrassingly parallelizable. Whereas MCMC is inherently sequential, and samples are obtained iteratively. Bayesian inference approaches that consist of a generative prior coupled with a way of sampling from its posterior are widely known as *modular* Bayesian approaches. GAN-Flow is also a modular approach in that sense, but distinct because it uses a GAN-based prior and a variational posterior induced by a normalizing flow. There are also several works where normalizing flows are constructed in lower-dimensional spaces and subsequently used to solve inverse problems. Some tools that have been used to derive or learn the injective map include principal component analysis [43], isometric auto-encoders [44] and in-

jective neural networks [45]. Brehmer and Cranmer [46] also explore padding the low dimensional latent variable with zeroes to increase dimensionality. Other approaches simultaneously learn the dimension reduction map and normalizing flow [46, 43, 44, 45]. GAN-Flow is different since it uses a GAN to approximate the prior density, and the generator of the WGAN serves as the injective map.

### *1.1. Summary of contributions*

In summary, the novel contributions of this work are as follows:

1. We introduce GAN-Flow, a novel unsupervised modular Bayesian inference framework, that combines two types of generative models — GANs and normalizing flows — and exploits their respective strengths.

2. We develop a two-stage strategy to train each sub-component of the GAN-Flow. First, the GAN is trained using *a priori* available samples from the prior distribution. Then, the normalizing flow model is used to perform variational Bayesian inference in the low-dimensional latent space of the GAN for efficient posterior approximation.

3. We demonstrate the efficacy of GAN-Flow on three large-scale physics-based inverse problems involving both synthetic and real-world data. We consider three inverse problems — inferring initial conditions in a heat conduction problem, an inverse Radon transform problem wherein an object is recovered from its sinogram, and a phase retrieval problem wherein an object is recovered form the magnitude of its Fourier transform. Where possible we compare GAN-Flow with Monte Carlo simulation and an inference approach previously proposed by Patel et al. [19] that also utilizes a WGAN-GP prior.

4. We also show that GAN-Flow is a flexible framework that can utilize various types of GAN and normalizing flow models. For the various problems we consider, we use different GAN models which include generators with self-attention units and GANs that are progressively grown. We also show that GAN-Flow can accommodate different types of normalizing flows such as planar flows and affine-coupling flows.

The remainder of this paper is organized as follows. Section 2 sets up the problem of interest, and provides a brief background on variational Bayesian inference, GANs and normalizing flows. We introduce GAN-Flow in Section 3 and discuss the training of its two sub-components. In Section 4, we apply GAN-Flow to solve three large-scale inverse problems. Finally, we draw conclusions in Section 5.

## 2. Background

### *2.1. Problem setup and Bayesian inference*

Consider the random vectors $\boldsymbol{x} \in \Omega_{\mathcal{X}} \subseteq \mathbb{R}^{n_{\mathcal{X}}}$ and $\boldsymbol{y} \in \Omega_{\mathcal{Y}} \subseteq \mathbb{R}^{n_{\mathcal{Y}}}$ related by the forward model $F : \Omega_{\mathcal{X}} \to \Omega_{\mathcal{Y}}$ such that $\boldsymbol{y} = F(\boldsymbol{x})$. Herein, $\boldsymbol{x}$, $\Omega_{\mathcal{X}}$ and $n_{\mathcal{X}}$ are called the *ambient* variable, space and dimension, respectively. The inference of $\boldsymbol{x}$ from a noisy measurement vector $\hat{\boldsymbol{y}}$ (a noisy realization of $\boldsymbol{y}$) constitutes an inverse problem. Given a likelihood function $\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{x})$, Bayes' rule is used to update prior belief about $\boldsymbol{x}$, characterized through the prior probability density function $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$, as follows:

$$\mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}}) = \frac{\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{x})\ \mathrm{p}_{\mathcal{X}}(\boldsymbol{x})}{\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})}, \tag{1}$$

where $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})$ is the posterior distribution and $\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})$ is the evidence or marginal likelihood. When measurements are corrupted by an additive noise $\boldsymbol{\eta}$, distributed according to $\mathrm{p}_{\eta}$, the measurement model $\hat{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{\eta}$ leads to the likelihood function $\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{x}) = \mathrm{p}_{\eta}(\hat{\boldsymbol{y}} - F(\boldsymbol{x}))$ in Eq. (1). The posterior distribution is useful for computing posterior-predictive statistics of any desired quantity of interest, herein denoted as $\ell(\boldsymbol{x})$. For instance, the posterior mean of $\ell(\boldsymbol{x})$ can be computed as follows:

$$\mathbb{E}_{\boldsymbol{x}\sim\mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})}\big[\ell(\boldsymbol{x})\big] = \int_{\Omega_{\mathcal{X}}} \ell(\boldsymbol{x})\mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})\ \mathrm{d}\boldsymbol{x} \tag{2}$$

Typically, the integral in Eq. (2) is high-dimensional and intractable for practically interesting problems, and must be approximated using Monte Carlo methods, which requires samples from the posterior distribution. Given a sample of size $n_{\mathrm{s}}$, the Monte Carlo approximation to Eq. (2) is given as:

$$\mathbb{E}_{\boldsymbol{x}\sim\mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})}\big[\ell(\boldsymbol{x})\big] \approx \frac{1}{n_{\mathrm{s}}}\sum_{i=1}^{n_{\mathrm{s}}} \ell(\boldsymbol{x}^{(i)}), \tag{3}$$

where $\boldsymbol{x}^{(i)}$ is the $i^{\text{th}}$ realization of $\boldsymbol{x}$ drawn from the posterior distribution. In this work, we propose the novel inference framework GAN-Flow, which is efficient at sampling the posterior distribution and, ultimately, estimating the statistics of posterior-predictive quantities. GAN-Flow is a hybrid of two types of generative models: generative adversarial networks (GANs) and normalizing flows.

### 2.2. Generative Adversarial Networks

Generative adversarial networks [13] are generative models consisting of two sub-networks: a generator and a discriminator (also known as the critic). GANs are trained *adversarially*: the generator tries to deceive the discriminator while the discriminator tries to distinguish between '*fake*' samples generated from the generator and '*true*' samples available from the target distribution. The generator and critic play an adversarial '*game*' between them with the ultimate goal of generating new realizations from an underlying distribution, the prior distribution $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$ in this case. Let the generator network $G$, parameterized by $\boldsymbol{\theta}$, map the *latent* variable $\boldsymbol{z} \in \Omega_{\mathcal{Z}} \subseteq \mathbb{R}^{n_{\mathcal{Z}}}$ to the target variable $\boldsymbol{x}$, i.e., $G(\cdot, \boldsymbol{\theta}) : \Omega_{\mathcal{Z}} \to \Omega_{\mathcal{X}}$. Herein, we refer to $\boldsymbol{z}$, $\Omega_{\mathcal{Z}}$ and $n_{\mathcal{Z}}$ as the *latent* variable, space and dimension, respectively. Typically, $\boldsymbol{z}$ is sampled from a simple distribution $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$, like the multivariate standard normal distribution. Moreover, the latent dimension $n_{\mathcal{Z}}$ is typically chosen to be much smaller than the ambient dimension $n_{\mathcal{X}}$, i.e., $n_{\mathcal{Z}} \ll n_{\mathcal{X}}$. Thus, GANs are endowed with dimension reduction capabilities and the generator $G$ serves as a map from the low-dimensional latent space to the high-dimensional ambient space. On the other hand, the discriminator $D$, parameterized by $\boldsymbol{\phi}$ such that $D(\cdot, \boldsymbol{\phi}) : \Omega_{\mathcal{X}} \to \mathbb{R}$, tries to differentiate between realizations drawn from $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$ and those generated by the generator.

The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ of the generator and the discriminator networks, respectively, are obtained through the min-max optimization of an appropriate loss function, say $\mathcal{L}_{\text{GAN}}$, i.e.,

$$(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*) = \arg\min_{\boldsymbol{\theta}} \left( \arg\max_{\boldsymbol{\phi}} \mathcal{L}_{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \right). \tag{4}$$

Different types of GANs will use different loss functions $\mathcal{L}_{\text{GAN}}$; interested readers may refer to [47, 48, 49] for an overview. It is important to note that training a GAN requires realizations of $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$, therefore, we assume that $n_{\text{data}}$ independent and identically distributed (iid) realizations of $\boldsymbol{x}$ from

$\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$ are available, which we herein denote using $\mathcal{S} = \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{n_{\text{data}}}$ and refer to $\mathcal{S}$ as the prior dataset. GAN-Flow uses the provided prior dataset to derive a data-driven informative prior. All that's required is a mechanism to sample from the GAN prior.

### *2.3. Variational Bayesian inference*

MCMC methods approximate the posterior distribution using correlated realizations of $\boldsymbol{x}$ that are sampled from an ergodic Markov chain with the stationary distribution $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})$. In contrast, variational Bayesian inference methods attempt to approximate the posterior probability distribution [8]. Variational Bayesian inference starts with a family of distributions $\mathrm{q}_{\mathcal{X}}(\boldsymbol{x};\boldsymbol{\psi})$ parameterized by $\boldsymbol{\psi}$. The optimal parameter vector $\boldsymbol{\psi}^*$ is determined by minimizing some divergence measure $d$ between $\mathrm{q}_{\mathcal{X}}(\boldsymbol{x};\boldsymbol{\psi})$ and $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})$:

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} d\big(\mathrm{q}_{\mathcal{X}}(\boldsymbol{x};\boldsymbol{\psi})\|\mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})\big). \tag{5}$$

The reverse Kullback-Leibler (KL) divergence is a popular choice for $d$ but other divergence measures have also been used [50]. Thus, variational Bayesian inference converts the problem of posterior sampling into an equivalent optimization problem. Once $\boldsymbol{\psi}^*$ has been determined, $\mathrm{q}_{\mathcal{X}}(\boldsymbol{x};\boldsymbol{\psi}^*)$ serves as an approximation to $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})$ and can be repeatedly sampled without additional likelihood evaluations to obtain as many posterior samples as required — unlike MCMC-based methods. As a result, variational Bayesian inference offers a computationally efficient alternative to MCMC sampling in many cases. The performance of variational Bayesian inference relies on the *a priori* chosen parameterized family of distribution $\mathrm{q}_{\mathcal{X}}(\cdot;\boldsymbol{\psi})$ being capable of approximating the posterior distributions, which can have a complex shape. This approximation may be difficult to achieve using standard distribution families like mixture models. Moreover, the computational effort of the optimization problem in Eq. (5) increases as the dimension of $\boldsymbol{\psi}$ increases, which is expected to happen as the ambient dimensionality of the inverse problem grows.

An alternative approach to explicitly working with a family of distributions is to define a *push-forward* map that can induce a good approximation to the posterior distribution. Let $H(\cdot;\boldsymbol{\psi}) : \Omega_{\mathcal{X}} \to \Omega_{\mathcal{X}}$ denote a bijective and differentiable map (also known as a diffeomorphism) that is pa-

rameterized by $\boldsymbol{\psi}$, and let $H_{\#}\mathrm{p}_{\mathcal{X}}(\boldsymbol{x};\boldsymbol{\psi})$ denote the pushforward of the prior distribution $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$. Then

$$H_{\#}\mathrm{p}_{\mathcal{X}}(\boldsymbol{x};\boldsymbol{\psi}) = \mathrm{p}_{\mathcal{X}}(\boldsymbol{x})|\det \nabla_{\boldsymbol{x}} H(\boldsymbol{x};\boldsymbol{\psi})|^{-1}, \tag{6}$$

as a result of change of variables, where $\det \nabla_{\boldsymbol{x}} H(\boldsymbol{x};\boldsymbol{\psi})$ is the Jacobian determinant of the push-forward map $H(\cdot;\boldsymbol{\psi})$. Therefore, one way of approximating the posterior distribution is to use a flexible diffeomorphism such that the pushforward distribution $H_{\#}\mathrm{p}_{\mathcal{X}}(\boldsymbol{x};\boldsymbol{\psi})$ is close (in some sense) to the posterior distribution. However, the successful application of Eq. (6) requires that the Jacobian determinant be easily computable. Many techniques, such as polynomial approximation and radial basis functions, can be used to construct diffeomorphisms that permit efficient Jacobian determinant computations [51]. More recently, normalizing flows [16, 15] have emerged as an efficient tool to construct high-dimensional diffeomorphisms.

### *2.4. Normalizing flows*

Normalizing flows are a class of generative models that uses invertible neural networks to construct diffeomorphisms. Normalizing flows are constructed in a manner that facilitates efficient computation of the Jacobian determinant. In practice, the inference map $H$ is constructed by stacking together multiple, say, $n_{\mathrm{f}}$ invertible layers, which makes

$$H(\boldsymbol{x}) = H_{[n_{\mathrm{f}}]}(H_{[n_{\mathrm{f}}-1]}(\cdots H_{[1]}(\boldsymbol{x}))). \tag{7}$$

Individual bijections $H_{[k]}$ are called flows and the composition $H$ is a normalizing flow. $\boldsymbol{\psi}$, which we intentionally suppress in Eq. (7) and herein, denotes the parameters of all flow layers taken collectively. Note that the composition of bijective functions is also a bijective function, and the Jacobian determinant of which can be computed as:

$$\det \nabla_{\boldsymbol{x}} H(\boldsymbol{x}) = \prod_{k=1}^{n_{\mathrm{f}}} \det \nabla_{\boldsymbol{x}_{[k-1]}} H_{[k]}(\boldsymbol{x}_{[k-1]}), \tag{8}$$

where $\boldsymbol{x}_{[k]} = H_{[k]}(\boldsymbol{x}_{[k-1]})$ and $\boldsymbol{x}_{[0]} = \boldsymbol{x}$.

Many different types of invertible architectures exist that define bijections for which the Jaco-

bian determinant is easily computable; see [16] for a recent review. In this work, we use two types of invertible architecture.

### 2.4.1. *Planar flows*

Rezende and Mohamed [14] proposed an invertible neural network architecture based on planar transformations that apply the following perturbation to the input $\boldsymbol{x}_{[k-1]}$ in $k^{\text{th}}$ flow layer $H_{[k]}$:

$$H_{[k]}(\boldsymbol{x}_{[k-1]}) = \boldsymbol{x}_{[k-1]} + \boldsymbol{u}_{[k]} \cdot S\left(\boldsymbol{w}_{[k]}^{\mathrm{T}} \boldsymbol{x}_{[k-1]} + b_{[k]}\right), \tag{9}$$

where $\boldsymbol{\psi}_{[k]} = \{\boldsymbol{u}_{[k]} \in \mathbb{R}^d, \boldsymbol{w}_{[k]} \in \mathbb{R}^d, b_{[k]} \in \mathbb{R}\}$ are the parameters of $H_{[k]}$, and $S : \mathbb{R} \to \mathbb{R}$ is a nonlinear activation function with derivative $S'$. Rezende and Mohamed [14] adopt the taxonomy 'planar' because they claim that the perturbation introduced to $\boldsymbol{x}_{[k-1]}$ is normal to the hyper-plane $\boldsymbol{w}_{[k]}^{\mathrm{T}} \boldsymbol{x}_{[k-1]} + b_{[k]} = 0$. However, note that, for a fixed $\boldsymbol{u}_{[k]}$, all points that lie on a plane $\boldsymbol{w}_{[k]}^{T} \boldsymbol{x}_{[k-1]} = c_{[k]}$ are perturbed by the same amount, equal to $\left| S\left(c_{[k]} + b_{[k]}\right) \right| \cdot \|\boldsymbol{u}_{[k]}\|$, and in the same direction as $\boldsymbol{u}_{[k]}$. Thus, planar transformations ensure that a collection of points in $d$-dimensional place that lie on a hyper-plane with normal $\boldsymbol{w}_{[k]}$ is merely translated in space and continue to a lie on a new hyper-plane that still has the same normal vector as before. The Jacobian determinant of $H_{[k]}$ is:

$$\left| \det \nabla_{\boldsymbol{x}_{[k-1]}} H_{[k]}(\boldsymbol{x}_{[k-1]}) \right| = \left| 1 + S'(\boldsymbol{w}_{[k]}^{\mathrm{T}} \boldsymbol{x}_{[k-1]} + b_{[k]}) \boldsymbol{u}_{[k]}^{\mathrm{T}} \boldsymbol{w}_{[k]} \right|. \tag{10}$$

Moreover, $\boldsymbol{w}_{[k]}^{\mathrm{T}} \boldsymbol{u}_{[k]} \geq -1$ is a sufficient condition for $H_{[k]}$ to be invertible when $S$ is the hyperbolic tangent function [14], which is what we use in this work.

### 2.4.2. *Affine-coupling flows*

Dinh et al. [41] introduced coupling flows, of which affine-coupling is a specific type. Let $\boldsymbol{x}_{[k-1]}^{a}$ and $\boldsymbol{x}_{[k-1]}^{b}$ be two disjoint partitions of the input vector $\boldsymbol{x}_{[k-1]}$, formed by randomly sampling components of $\boldsymbol{x}_{[k-1]}$, then the coupling flow layer $H_{[k]}$ applies the following transformations to its input $\boldsymbol{x}_{[k-1]}$:

$$\boldsymbol{x}_{[k]}^{b} = \boldsymbol{x}_{[k-1]}^{b} \odot \exp\left[S_1(\boldsymbol{x}_{[k-1]}^{a})\right] + T_1(\boldsymbol{x}_{[k-1]}^{a}) \text{ and } \boldsymbol{x}_{[k]}^{a} = \boldsymbol{x}_{[k-1]}^{a} \odot \exp\left[S_2(\boldsymbol{x}_{[k]}^{b})\right] + T_2(\boldsymbol{x}_{[k]}^{b}), \tag{11}$$

where $\boldsymbol{x}_{[k]}^{\mathrm{T}} = \left[\boldsymbol{x}_{[k]}^{a\mathrm{T}}, \boldsymbol{x}_{[k]}^{b\mathrm{T}}\right]$, $S_1$ and $S_2$ are known as scale networks, $T_1$ and $T_2$ are known as shift networks, and $\odot$ denotes the Hadamard product. The scale and shift networks are modeled using deep neural networks that preserve the dimensionality of their respective inputs. A coupling layer constrains the Jacobian to be upper triangular [41, 42]. The determinant of the Jacobian is [42]:

$$\left|\det \nabla_{\boldsymbol{x}_{[k-1]}} H_{[k]}(\boldsymbol{x}_{[k-1]})\right| = \left(\exp\left[\sum_j \left\{S_1(\boldsymbol{x}_{[k-1]}^a)\right\}_j\right]\right)\left(\exp\left[\sum_j \left\{S_2(\boldsymbol{x}_{[k-1]}^c)\right\}_j\right]\right), \quad (12)$$

where $\{\cdot\}_j$ denotes the $j^{\text{th}}$ component of a vector, and $S_1(\boldsymbol{x}_{[k-1]}^a)$ and $S_2(\boldsymbol{x}_{[k-1]}^c)$ are outputs from the scale networks $S_1$ and $S_2$, respectively.

## 3. Bayesian inference using GAN-Flow

Bayesian inference is useful for solving statistical inverse problems, but its practical application to large-scale inverse problems is far from straightforward. First, it is important to recognize that the quality of inference depends on the prior [52], more so when there is paucity of data. Simple parametric priors derived from tractable distributions are not useful for describing complex entities such as brain scans, thermal conductivity fields, and the matrix of a composite material; recent recognition of this fact has fostered efforts to develop physics-informed data-driven priors [53, 19]. Second, posterior sampling using MCMC methods is difficult in high-dimensional spaces, i.e., when $n_{\mathcal{X}}$ is large. In high-dimensional spaces, Markov chains tend to take a long time before they can reach a '*steady state*', and assessing the convergence of Markov chains is also difficult. Third, MCMC sampling involves repeated evaluations of the likelihood function, which means that the underlying physics-based forward model must be evaluated during sampling and that the cost of obtaining new samples will scale linearly with the cost of forward model evaluations; this is undesirable. Thus, MCMC sampling from high-dimensional posteriors continues to be a challenging and computationally intensive task, which has been a major deterrent to the practical application of Bayesian inference to large-scale inverse problems. GAN-Flow attempts to circumvent these issues by coupling together two types of deep generative models — generative adversarial networks and normalizing flows.

### 3.1. Overview of GAN-Flow

GAN-Flow uses a GAN, specifically a Wasserstein GAN, to form a data-driven informative prior that can synthesize realizations of $\boldsymbol{x}$ similar to the constituents the prior dataset $\mathcal{S}$. Moreover, the generator network of the GAN becomes a map between the low-dimensional latent space and the high-dimensional ambient space. The inverse problem is solved in the low-dimensional latent space using variational Bayesian inference as the normalizing flow model acting as a pushforward operator from the prior distribution to the posterior distribution. Fig. 1 shows the three phases of the GAN-Flow framework.

### 3.2. Phase A: Training a GAN-based prior

GAN-Flow utilizes a Wasserstein GAN with Gradient Penalty (WGAN-GP) [54, 55] to model the prior probability distribution $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$. For a WGAN-GP, the loss function $\mathcal{L}_{\text{GAN}}$ is given as

$$\mathcal{L}_{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{x}\sim \mathrm{p}_{\mathcal{X}}(\boldsymbol{x})}\left[D(\boldsymbol{x}, \boldsymbol{\phi})\right] - \mathbb{E}_{\boldsymbol{z}\sim \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})}\left[D(G(\boldsymbol{z}, \boldsymbol{\theta}), \boldsymbol{\phi})\right], \tag{13}$$

and the min-max optimization problem in Eq. (4) is solved under the constraint that $D(\boldsymbol{z}, \boldsymbol{\phi})$ lies in the space of 1-Lipschitz functions. This constraint ensures that the inner maximization problem in Eq. (4) leads to an approximation of the Wasserstein-1 distance (due to the Kantorovich-Rubinstein duality [56]) between $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$ and the pushforward of $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$ induced by $G(\cdot; \boldsymbol{\theta})$ [54, 55, 19]. The 1-Lipschitz constraint is satisfied by enforcing a soft penalty on the gradients of the critic $D$ with respect to $\boldsymbol{z}$ [55]. The resulting maximization problem that is solved to optimize the parameters of the discriminator is

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\, \mathcal{L}_{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi}) - \lambda \mathbb{E}_{\tilde{\boldsymbol{x}}\sim \mathrm{p}_{\tilde{\boldsymbol{X}}}(\tilde{\boldsymbol{x}})}\left[(\|\nabla_{\tilde{\boldsymbol{x}}} D(\tilde{\boldsymbol{x}}, \boldsymbol{\phi})\|_2 - 1)^2\right], \tag{14}$$

where $\lambda$ is the gradient penalty parameter, and $\mathrm{p}_{\tilde{\boldsymbol{X}}}(\tilde{\boldsymbol{x}})$ is the uniform distribution over the straight line joining two pairs of points sampled from $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$ and the pushforward of $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$ by $G$. The loss function in Eq. (14) minimizes the Wasserstein-1 distance between $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$ and the pushforward distribution of $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$ due to $G$ [35].

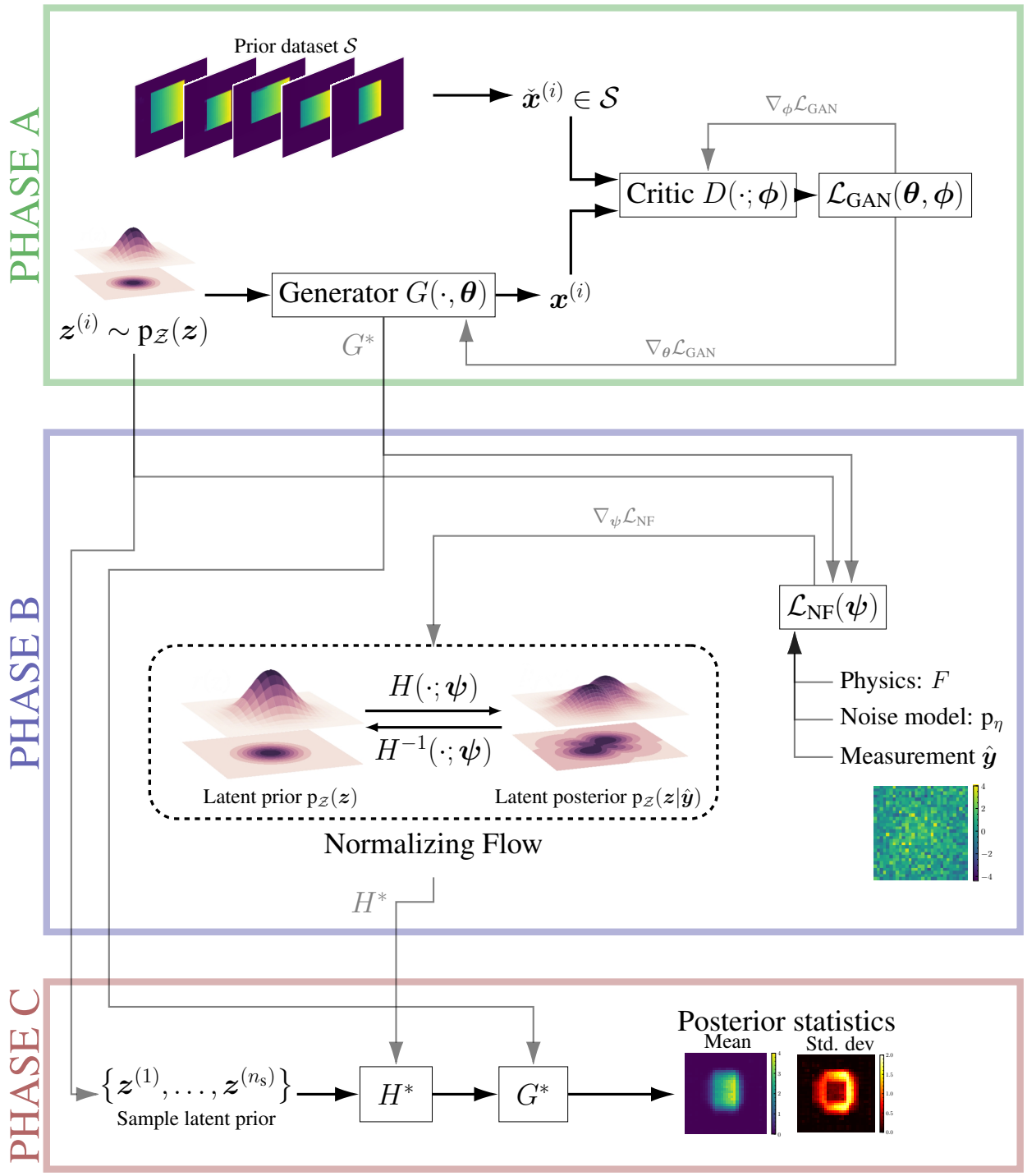


Figure 1. Schematic diagram of the proposed GAN-Flow framework for solving physics-based inverse problems. PHASE A involves the training of a WGAN-GP model with training samples from the prior distribution. In PHASE B, the trained generator $G^*$, the physics model $F$, the noise model $\mathrm{p}_\eta$ and the measurements $\hat{\boldsymbol{y}}$ are used to train the normalizing flow map $H$. PHASE C corresponds to posterior sampling that is achieved by using the trained normalizing flow map $H^*$ to transform realizations from the latent prior into realizations from the latent posterior, which are then passed through the trained generator $G^*$ to obtain realizations from the ambient posterior distribution.

Now, let $G^*$ denote the generator $G$ with optimally chosen parameters $\boldsymbol{\theta}^*$. For a perfectly trained GAN $G^*$,

$$\mathbb{E}_{\boldsymbol{x}\sim \mathrm{p}_{\mathcal{X}}(\boldsymbol{x})}\big[m(\boldsymbol{x})\big] = \mathbb{E}_{\boldsymbol{z}\sim \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})}\big[m(G^*(\boldsymbol{z}))\big] \quad \forall\, m \in C_b(\Omega_{\mathcal{X}}), \tag{15}$$

where $C_b(\cdot)$ is the space of continuously bounded functions. Eqs. (2) and (15) can be combined to compute

$$\mathbb{E}_{\boldsymbol{x}\sim \mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})}\big[\ell(\boldsymbol{x})\big] = \mathbb{E}_{\boldsymbol{z}\sim \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}})}\big[\ell(G^*(\boldsymbol{z}))\big] \quad \forall\, \ell \in C_b(\Omega_{\mathcal{X}}), \tag{16}$$

by choosing

$$m(\boldsymbol{z}) = \frac{\ell(\boldsymbol{z})\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{z})}{\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})}, \tag{17}$$

where

$$\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}}) = \frac{\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{z})\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})}{\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})} \tag{18}$$

is the posterior distribution of the latent variable $\boldsymbol{z}$ and $\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{z})$ is nothing but $\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{x})$ evaluated at $\boldsymbol{x} = G^*(\boldsymbol{z})$, i.e.,

$$\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{z}) = \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{x})|_{\boldsymbol{x}=G^*(\boldsymbol{z})}. \tag{19}$$

For additive noise models, Eq. (19) simplifies to $\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{z}) = \mathrm{p}_{\eta}(\hat{\boldsymbol{y}} - F(G^*(\boldsymbol{z})))$. Significantly, Eq. (16) implies that any statistics with respect to $\boldsymbol{x}$ (respectively $\boldsymbol{x}|\hat{\boldsymbol{y}}$) can be computed using realizations of $\boldsymbol{z}$ (respectively $\boldsymbol{z}|\hat{\boldsymbol{y}}$).

At the end of phase A, a trained generator $G^*$ is available. The GAN not only acts as a data-driven prior, but the trained generator $G^*$ also serves as an injective map that can conveniently transform realizations of the latent variable $\boldsymbol{z}$ from the latent space $\Omega_{\mathcal{Z}}$ to corresponding realizations of the ambient variable $\boldsymbol{x}$ in the ambient space $\Omega_{\mathcal{X}}$. Additionally, $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$ is herein chosen to be an $n_{\mathcal{Z}}$-variate standard normal distribution. Also note that, in most practical cases, $n_{\mathcal{X}}$ will be large but the latent dimensionality of the WGAN-GP model will be such that $n_{\mathcal{Z}} \ll n_{\mathcal{X}}$. Thus, dimension reduction is achieved by using GAN priors. The next step is to solve the inverse problem in this lower-dimensional latent space.

*Remark* 1. Choosing the latent space dimensionality $n_{\mathcal{Z}}$ from $n_{\text{data}}$ realizations of $\mathrm{p}_{\mathcal{X}}(\boldsymbol{x})$ is not

straightforward. This will involve careful judgment from the user. However, meta-heuristic metrics, such as the Fréchet Inception Distance (FID) score [57] and Inception score [58], are useful for comparing the quality of the realizations generated using GANs and real samples. These metrics can be used to evaluate the quality of the trained GAN, and suitably adjust the latent space dimensionality if performance is not satisfactory. Metrics that are more physically motivated may also be utilized in engineering applications. For instance, the consistency of microstructural descriptors, like the distribution of porosity in a bi-phase material [59], across the generated samples can be used when evaluating GANs for generating microstructures of heterogeneous media. We suggest choosing the smallest possible latent dimensionality that performs satisfactorily. This will also help keep the subsequent normalizing flow model relatively lightweight.

*Remark* 2. Unless there is an influx of new prior information that must be incorporated, there is no need to retrain WGAN-GP prior. This means that Phase A, which involves the training of the WGAN-GP model, must be completed only once for a given inverse problem. Thereafter, we can reuse the trained generator for multiple inferences.

### *3.3. Phase B: Inference using normalizing flows*

With the latent prior density $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$ and an optimally trained generator $G^*$, a normalizing flow is used to sample from the conditional posterior $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}})$. This is done by learning a bijective map $H : \Omega_{\mathcal{Z}} \to \Omega_{\mathcal{Z}}$, parameterized by $\boldsymbol{\psi}$, such that $H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi}) \sim \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}})$. The parameters $\boldsymbol{\psi}$ of the bijective map $H$ are chosen by minimizing the loss function

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} d_{\mathrm{KL}}\big(H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi})\|\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}})\big), \tag{20}$$

where $d_{\mathrm{KL}}(\cdot\|\cdot)$ is the reverse KL divergence. On simplifying Eq. (20), the loss function $\mathcal{L}_{\mathrm{NF}}$ for training the normalizing flow takes the form:

$$\mathcal{L}_{\mathrm{NF}}(\boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{z}\sim\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})}\big[-\log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|H(\boldsymbol{z};\boldsymbol{\psi})) - \log \mathrm{p}_{\mathcal{Z}}(H(\boldsymbol{z};\boldsymbol{\psi})) - \log|\det \nabla_{\boldsymbol{z}} H(\boldsymbol{z};\boldsymbol{\psi})|\,\big], \tag{21}$$

where $\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|H(\boldsymbol{z}))$ can be evaluated using Eq. (19) as

$$\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|H(\boldsymbol{z};\boldsymbol{\psi})) = \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{x})|_{\boldsymbol{x}=G^*(H(\boldsymbol{z}))}, \tag{22}$$

and, for additive noise models, $\mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|H(\boldsymbol{z};\boldsymbol{\psi})) = \mathrm{p}_{\eta}(\hat{\boldsymbol{y}} - F(H(G^*(\boldsymbol{z});\boldsymbol{\psi})))$. We provide a detailed derivation of Eq. (21) starting from Eq. (20) in Appendix A; our derivation closely follows a similar derivation by Sun and Bouman [28]. In Eq. (21), $\nabla_{\boldsymbol{z}} H(\boldsymbol{z})$ is the Jacobian of $H$. Thus, the physics model $F$ and the trained generator $G^*$ enters Eq. (21) through the log-likelihood term $\log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|H(\boldsymbol{z};\boldsymbol{\psi}))$. At the end of phase B, a trained normalizing flow map $H(\cdot;\boldsymbol{\psi}^*)$, herein denoted as $H^*$, is available alongside the trained generator $G^*$.

*Remark* 3. Suppose there is a change in the measurement vector $\hat{\boldsymbol{y}}$ as new measurements are available or the forward model $F$ changes, we must retrain the normalizing flow model. Perhaps one can reduce the computational burden of retraining through knowledge transfer from the previously trained normalizing flow model, for instance, by starting training from the old weights, by freezing the weights of some of the flow layers, or by simply appending new flow layers to the existing normalizing flow model. However, we do not consider such cases in this work, and the development of knowledge transfer schemes is beyond the scope of the current work.

*Remark* 4. The training of the normalizing flow model boils down to the minimization problem in Eq. (20) with the loss function given by Eq. (21). The minimization problem can be solved using an appropriate stochastic gradient descent algorithm. In this work, we use the Adam algorithm [60]. Regardless of the optimization algorithm used, training the normalizing flow using gradient descent algorithms involves the computation of the gradients of the output forward model $F$ with respect to its input. This will pose challenges when $F$ is a black-box model that is incompatible with automatic differentiation, ultimately leading to an increase in the overall computational cost; this challenge is not a bottleneck unique to GAN-Flow as advanced MCMC algorithms, like HMC, also require the gradients of $F$ [19]. One potential solution is to couple GAN-Flow with automatically differentiable surrogate models for $F$, such as neural networks [61], but this is beyond the study herein.

### 3.4. Phase C: Posterior sampling and estimation

We can use the optimally trained generator $G^*$ and normalizing flow map $H^*$ to estimate $\mathbb{E}_{\boldsymbol{x}\sim \mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})}\big[\ell(\boldsymbol{x})\big]$ using Monte Carlo (MC) simulation. Let $\bar{\ell}$ denote the MC estimator for $\mathbb{E}_{\boldsymbol{x}\sim \mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})}\big[\ell(\boldsymbol{x})\big]$ given by

$$\bar{\ell} = \frac{1}{n_{\mathrm{s}}}\sum_{i=1}^{n_{\mathrm{s}}} \ell(G^*(H^*(\boldsymbol{z}^{(i)}))), \tag{23}$$

where $\boldsymbol{z}^{(i)}$ are independent realizations drawn from $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$. For instance, the posterior mean of $\boldsymbol{x}$ can be computed by setting $\ell(\boldsymbol{x}) = \boldsymbol{x}$. Let $\bar{\boldsymbol{x}}$ denote the MC estimator for $\mathbb{E}_{\boldsymbol{x}\sim \mathrm{p}_{\mathcal{X}}(\boldsymbol{x}|\hat{\boldsymbol{y}})}\big[\boldsymbol{x}\big]$, then

$$\bar{\boldsymbol{x}} = \frac{1}{n_{\mathrm{s}}}\sum_{i=1}^{n_{\mathrm{s}}} G^*(H^*(\boldsymbol{z}^{(i)})), \tag{24}$$

where $\boldsymbol{z}^{(i)}$ are iid realizations from $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$. Similarly, the standard deviation for the $i^{\text{th}}$ component of $\boldsymbol{x}$, herein denoted as $[\sigma_{\boldsymbol{x}}]_i$, can be estimated as:

$$\{\sigma_{\boldsymbol{x}}\}_i = \sqrt{\frac{1}{n_{\mathrm{s}}-1}\sum_{j=1}^{n_{\mathrm{s}}} \big[\,\{G^*(H^*(\boldsymbol{z}^{(j)}))\}_i - \{\bar{\boldsymbol{x}}\}_i\big]^2}, \tag{25}$$

where $[\cdot]_i$ denotes the $i^{\text{th}}$ component of the vector corresponding vector.

*Remark* 5. Neither evaluating $H^*$ nor $G^*$ requires evaluation of the underlying physics model $F$. Therefore, posterior statistics or posterior predictive quantities can be computed at almost negligible computational cost, and $n_{\mathrm{s}}$ may be set arbitrarily large. This is another advantage of the proposed GAN-Flow framework.

### 3.5. Discussion of the computational cost of each phase of GAN-Flow

Phase A of GAN-Flow involves the training of the WGAN-GP model. However, GAN-Flow does not evaluate the forward model $F$ at this stage. Thus, the computational cost of Phase A will be dominated by the cost of training the WGAN-GP model. Accordingly, the computational cost of Phase A will scale with the total number of trainable parameters in the generator and the critic. We expect that more complex prior information will require expressive models with a large number of trainable parameters to develop a suitable generator and, therefore, will be more

computationally expensive. Phase B, which involves training the normalizing flow model, requires repeated evaluations of the forward model $F$. Thus, the cost of evaluating the forward model $F$ will dominate the computational cost of Phase B. More complicated posterior distributions will require an expressive normalizing flow model. Expressivity can be achieved by adding more flow layers or using complex transformations, like the affine coupling transform, which will require longer training, translating to more evaluations of the forward model $F$, ultimately increasing the computational cost. Phase C neither involves training network models nor evaluating the forward model $F$. Thus, Phase C will be the least computationally demanding stage of GAN-Flow. The computational cost of this step will be dominated by the cost of evaluating the trained generator $G^*$ and the trained normalizing flow model $H^*$ and scale with the sample size $n_{\mathrm{s}}$.

The actual wall time of any particular phase will primarily depend on the particular application at hand, the scale and type of resource available, and the choice of training hyperparameters, among other factors. For instance, for the initial condition inference problem that we consider in Section 4.1, phases A, B and C require approximately 1.6 hours, 8.5 minutes and 6 seconds, respectively, when the generative models are trained using an NVIDIA Quadro RTX 8000 GPU with 48 GB memory (see Fig. B1 and table B1 in Appendix B1 for details of the model architectures and training hyper-parameters); thus, GAN-Flow requires less than 2 hours for completion. Similarly, for the inverse Radon transform problem, phases A, B and C require approximately 3.4 hours, 1.6 hours and 11 seconds, respectively, when the generative models are trained using the same NVIDIA Quadro RTX 8000 GPU (see Fig. B2 and table B1 in Appendix B1 for details of the model architectures and training hyper-parameters). The more challenging phase imaging problem (Section 4.3) requires approximately 23.5 hours, 16.7 hours and 2 minutes for phases A, B and C, respectively when all phases were executed using a NVIDIA A40 GPU with 48 GB of memory (see Appendices B1.1 and B2 and table B1 in Appendix B1 for details of the model architectures and training hyper-parameters). For the problems we consider in this work, the wall-time of phase A dominates because the forward model $F$ is relatively simple. We envisage that Phase B will be the most computationally demanding stage of GAN-Flow in physics-based applications where the underlying forward model $F$ is computationally demanding.

### 3.6. *Approximation errors due to GAN-Flow*

There are two potential sources of error in GAN-Flow when it is used in practice. The first source of error is from the WGAN-GP prior. A WGAN-GP trained using Eq. (14) may only be able to approximately satisfy Eq. (15) [62] or not at all [63]. The error may stem from using an approximate estimator for the Wasserstein distance in Eq. (14), the use of MC estimates for estimating the Wasserstein distance, and failure to reach the optimal point [62]. However, previous research [35, 19] has shown that WGAN-GP is useful as a data-driven prior despite theoretical concerns. In practice, we monitor the loss function and terminate training when its value no longer decreases. A second source of error stems from the normalizing flow map in situations where it is unable to induce a good approximation to the latent posterior distribution [64]. The latent posterior distribution may not belong to the family of distributions that the bijective map is capable of inducing, which may be due to the limited fidelity (expressive power) of the normalizing flow model. The pushforward distribution may also fail to approximate the latent posterior when the reverse KL-divergence loss does not attain a value of zero, possibly due to slow convergence during training. Even if the loss function attains a value of zero, the reverse KL-divergence is known to be 'mode seeking', therefore, it is possible that the pushforward distribution is unable to approximate the tails of the latent posterior. In practice, we monitor the value of the loss function $\mathcal{L}_{\text{NF}}$ and stop training when it no longer decreases. However, it must be noted that, even when the pushforward distribution is poor, estimates computed using the pushforward map may continue to be useful. So, we evaluate the error in the estimated posterior statistics, such as the posterior mean and standard deviation, to determine the quality of inference. In a practical setting, we suggest the use of diagnostic tools to ascertain the quality of the pushforward distribution [64] or stacking multiple normalizing flows with different seeds [65].

## 4. Results

In this section, we use GAN-Flow to solve three physics-based inverse problems: inference of initial conditions (Section 4.1), inverse Radon transform (Section 4.2), and phase retrieval (Section 4.3). By solving these inverse problems, we demonstrate that GAN-Flow can tackle

large-scale linear and non-linear inverse problems, various levels of noise, and challenging prior distributions. Our implementation also reveals that GAN-Flow is flexible in accommodating different types of WGAN-GP and normalizing flow architecture types, training methodologies, and combinations thereof. More specifically, the variety in the numerical examples we present lies in the following:

1. Forward model — The three inverse problems we consider involve different physical phenomena. The initial condition inference problem is based on heat diffusion in a solid body, and the inverse Radon transform problem is based on an object's attenuation of penetrating waves by an object. Both aforementioned physics phenomena can be described using a linear forward model; we adapt these inverse problems from [66, 19]. The third problem we consider is phase retrieval, which forms the underpinnings of many modern coherent diffraction imaging methods, wherein an object is reconstructed from the magnitude of its Fourier transform. In this case, the forward model is highly nonlinear. We adapt the phase imaging problem from [20].

2. Prior dataset — We consider three different prior datasets. The prior dataset for the initial condition inference and inverse Radon transform problems consists of rectangular inclusions in a zero-background and Shepp-Logan head phantoms, respectively; however, these priors are synthetic and adapted from [66, 19]. For the phase retrieval problem we consider a sub-sample of the publicly available NYU fastMRI [38, 37] dataset of human knee slices.

3. Ambient dimensionality — The ambient dimensionality of the inverse problems we consider vary vastly. While the initial condition inference problem has a moderate ambient dimensionality of 1,024, the phase retrieval problem is a large-scale inverse problem with an ambient space of dimension 65,536, an order of magnitude beyond that of the initial condition inference problem.

4. WGAN-GP prior model — For all numerical examples, we consider a WGAN-GP prior, i.e., the loss function used to train the GANs is given by Eq. (14). However, we use different architectures and/or training methodologies. The inverse Radon transform problem utilizes

a simple generator and critic, consisting of fully connected and convolution layers. We use self-attention-based [39] modules, along with convolutions, for the generator and critic for the initial condition inference problem. We found that self-attention modules help render the sharp transitions between the rectangular inclusion and the zero background. The phase retrieval problem requires still more sophisticated training to synthesize large (256×256) knee slices with many fine-scale features. The WGAN-GP model for the phase retrieval problem is trained using the progressive growing of GAN methodology [40].

5. Dimension reduction — The WGAN-GP priors themselves lead to latent spaces of varying dimension. We work with a low-dimensional latent space for the initial condition inference problem ($n_{\mathcal{Z}} = 5$), and a latent space of dimension 512 for the phase retrieval problem. On average, we can achieve approximately $\mathcal{O}(10^2)$ dimension reduction across all three inverse problems while maintaining satisfactory accuracy of the estimated statistics of the posterior distribution.

6. Normalizing flow model — We use two types of flow layers. For the low to moderate dimensional latent spaces, as in the initial condition inference and inverse Radon transform problem, we employ planar flow layers. For the relatively high-dimensional latent space of the phase retrieval problem, we use affine coupling transforms to construct the normalizing flows.

Table 1 provides a summary of the inverse problems we consider, their ambient and latent space dimensionality, and the dimension reduction.

We implement GAN-Flow exclusively on `PyTorch` [67]. Where possible, we compare the posterior statistics estimated using GAN-Flow with the method outlined in [19]; herein, we refer to the latter method as GAN-HMC because it uses HMC to sample the latent posterior and estimate the posterior statistics of the ambient variable. We implement HMC within `PyTorch` using the `hamiltorch` package [68]. In all cases, we specify the number of leap-frog steps to be 10, and discard 50% of the accepted states considering burn-in. The step size is adapted during the burn-in phase so as to maintain a desired acceptance rate of 0.75. For the initial condition inference problem, where the underlying (hidden) ambient dimensionality of the synthetic prior is

small (only four), we even compare the posterior statistics estimated using GAN-Flow with MC simulation (MCS).

For all examples, following Patel et al. [19], we re-scale the prior realizations between $[-1, 1]$ before training the WGAN-GP model and use hyperbolic tangent (TanH) activation in the last layer of the generator. We invert this re-scaling operation before evaluating the likelihood function. In this way, we satisfy physical constraints such as positive values of the temperature fields for the initial condition inference problem, or positive values of material density for the inverse Radon transform problem, or positive refractive index of an object in the phase retrieval problem.

### *4.1. Inferring the initial conditions in heat conduction*

The first problem we consider is a two-dimensional unsteady heat conduction problem where the initial condition of the temperature field $\boldsymbol{x}$ (at time $t = 0$) must be inferred from a noisy measurement of the temperature field $\hat{\boldsymbol{y}}$ taken after some time (at time $t = 1$). Inverse problems of this type often arise when designing thermal equipment [69, 70]. The two-dimensional time-dependent heat conduction partial differential equation over the bounded domain $\Omega$ is given as:

$$\begin{aligned} \frac{\partial u(\boldsymbol{s},t)}{\partial t} - \nabla \cdot (\kappa(\boldsymbol{s}) \nabla u(\boldsymbol{s},t)) &= b(\boldsymbol{s},t), \qquad \forall (\boldsymbol{s},t) \in \Omega \times (0,T) \\ u(\boldsymbol{s},0) &= m(\boldsymbol{s}), \qquad \forall \boldsymbol{s} \in \Omega \\ u(\boldsymbol{s},t) &= 0, \qquad \forall (\boldsymbol{s},t) \in \partial\Omega \times (0,T) \end{aligned} \tag{26}$$

Table 1. Summary of inverse problems we consider in this work.

| | Inverse problem | | |
|---|---|---|---|
| | Heat conduction (Section 4.1) | Radon transform (Section 4.2) | Phase imaging (Section 4.3) |
| Type | Linear | Linear | Non-linear |
| Ambient dimension $n_{\mathcal{X}}$ | $32 \times 32$ | $128 \times 128$ | $256 \times 256$ |
| Prior dataset | Rectangular | Shepp-Logan phantom | fastMRI [38, 37] |
| Prior dataset size $n_{\text{data}}$ | 2000 | 8000 | 29877 |
| Latent dimension $n_{\mathcal{Z}}$ | 5 | 60 | 512 |
| Dimension reduction $n_{\mathcal{X}}/n_{\mathcal{Z}}$ | $\sim$200 | $\sim$273 | 128 |

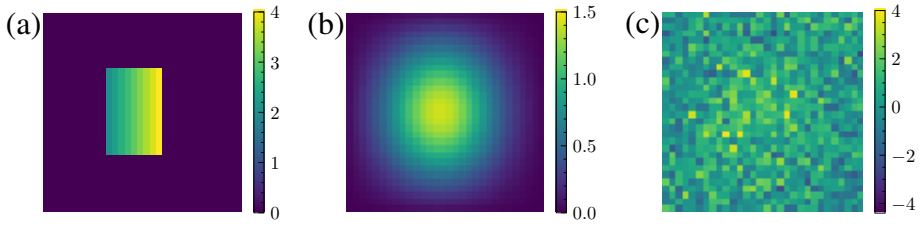


Figure 2. (a) Initial and (b) final temperature fields at $t = 0$ and 1, respectively. (c) Noisy synthetic measurements obtained after adding Gaussian white noise with unit variance to the temperature field shown in (b).

where $T$ is the final time at which measurements are made, i.e., $T = 1$, and the spatial domain $\Omega$ is a square region with the length of each side being $2\pi$ units, i.e., $\Omega = [0, L] \times [0, L]$ with $L = 2\pi$ units. We represent the solid on a $32 \times 32$ Cartesian grid over $\Omega$. We assume that the thermal conductivity $\kappa$ is homogeneous over $\Omega$ and equal to 0.64 units, and that $b(\boldsymbol{s}, t) = 0$. We use the central difference scheme to discretize temperature field on the same Cartesian grid as the solid body, thus, $n_{\mathcal{X}} = n_{\mathcal{Y}} = 1024$. The forward operator $F$ maps the initial temperature field $\boldsymbol{x}$ to the temperature field at time $T = 1$. We use backward-difference with a step size of 0.01 for the time-integration of Eq. (26). In this example, the inverse problem at hand is linear and it is possible to relate $\boldsymbol{x}$ and $\boldsymbol{y}$ using a linear operator, i.e., $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ [66]. The temperature fields at $t = 0$ (initial condition) and $t = T = 1$ are shown in Fig. 2(a) and (b), respectively. We add Gaussian white noise with unit variance to the temperature field at time $t = 1$ to generate the synthetic measurements; see Fig. 2(c). From these noisy measurements, we want to infer the initial condition shown in Fig. 2(a).

The prior dataset consists of $n_{\text{data}} = 2000$ initial temperature fields where the temperature is zero outside the rectangular inclusion and, within the rectangular inclusion, the temperature field varies linearly from a value of 2 units on the left edge to 4 units on the right edge. The rectangular region is generated by sampling uniformly the coordinates of the top-left and lower-right corners of the inclusion between $[0.2L, 0.4L]$ and $[0.6L, 0.8L]$, respectively. We show four realizations from the prior dataset in Fig. 3(a). The true temperature field in Fig. 2(a), which we want to infer, does not belong to the prior dataset. First, we train a WGAN-GP using the prior dataset. We choose the

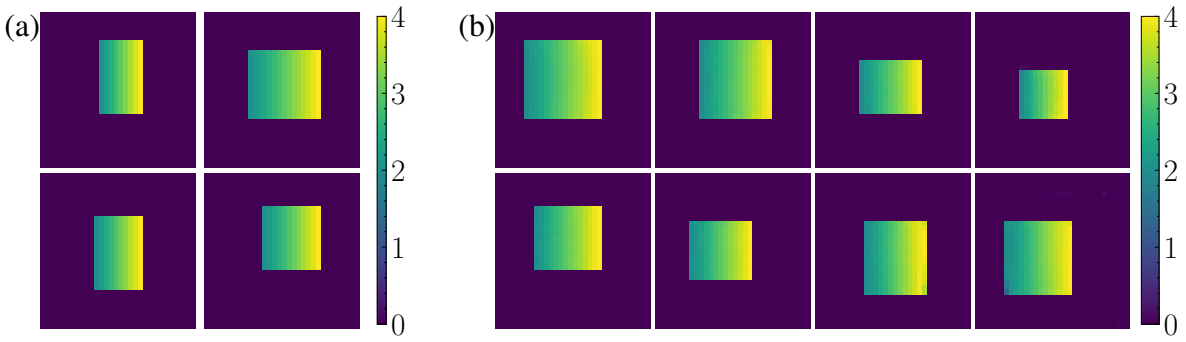


Figure 3. (a) Realizations from the rectangular prior dataset. (b) Realizations generated from the WGAN-GP prior.

latent space dimensionality to be 5, i.e., $n_{\mathcal{Z}} = 5$[1]. Therefore, we achieve a dimension reduction of approximately 200 times in this case. For details about the WGAN-GP model and the associated hyper-parameters, see Appendix B1 and Table B1, respectively. We show some realizations from the trained WGAN-GP prior in Fig. 3(b). The generated realizations are qualitatively similar to those in the prior dataset. We emphasize that training the WGAN-GP does not require any evaluation of the forward model $F$.

After we train the WGAN-GP model, we turn to training the normalizing flow model. In this example, the normalizing flow comprises 64 planar flow layers. See Table B1 for more details about the hyper-parameters associated with training the normalizing flow model. Significantly, the normalizing flow model is trained for 1000 epochs with a batch size of 32, meaning a total of 32,000 evaluations of the forward model $F$. After the normalizing flow model has been trained, we can use both the trained generator of the WGAN-GP model and the normalizing flow model to obtain as many samples from the posterior as necessary. We show the posterior mean and standard deviation of the initial temperature field estimated using GAN-Flow from a sample of size 15,000 in the left most column of Fig. 4. For the purposes of comparison, the posterior mean and standard deviation estimated using MCS (of sample size $10^6$) and GAN-HMC are also shown in Fig. 4. In contrast to GAN-Flow, GAN-HMC makes $3\times10^5$ evaluations of $F$ to yield 15,000 realizations from the posterior. Table 2 tabulates the root-mean-square error of the statistics esti-

[1]We vary the latent space dimensionality $n_{\mathcal{Z}} \in \{5, 10, 20, 40, 60, 80, 100\}$ keeping all other hyper-parameters fixed, and choose the smallest latent space dimension to yield the best estimates of the posterior mean and standard deviation; see Appendix C1 for the results from those experiments

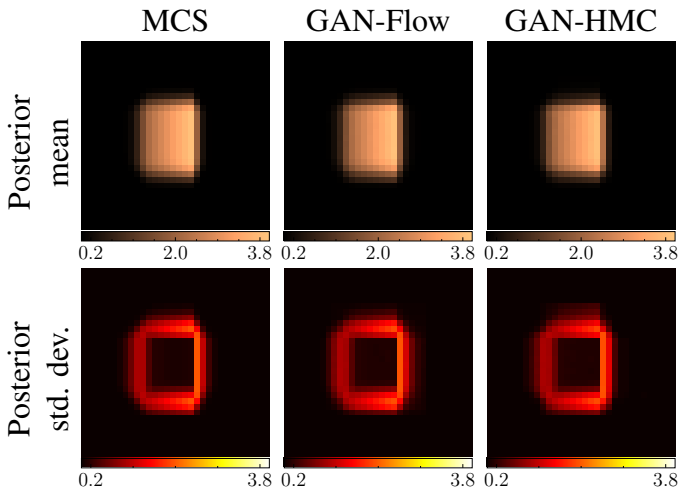


Figure 4. Estimated posterior mean (top row) and standard deviation (bottom row) obtained using MCS (left column), GAN-Flow (middle column) and GAN-HMC (right column) for the initial condition inference problem.

Table 2. RMSE in the posterior statistics obtained using different inference method compared to MCS.

| Method | Posterior statistic | |
|---|---|---|
| | Mean | Standard deviation |
| GAN-Flow | 0.034 | 0.048 |
| GAN-HMC | 0.071 | 0.049 |

mated using GAN-Flow and GAN-HMC, with the statistics estimated using MCS serving as the reference. From Fig. 4, we observe that the posterior statistics estimated using GAN-Flow and GAN-HMC compare very well with the '*true*' posterior statistics estimated using MCS. Quantitatively, the posterior mean estimated using GAN-Flow is marginally better than GAN-HMC, but this improvement is achieved with greater computational efficiency (about one order of magnitude fewer evaluations of the forward model $F$). These results are promising and suggest that GAN-Flow may even be more computationally efficient than GAN-HMC. While we consider Dirichlet boundary conditions in Eq. (26), investigations on a similar inverse problem with Neumann boundary conditions will be interesting; we leave this for a future work.

### *4.2. Inverse Radon transform*

Next, we consider the inverse problem of reconstructing an object from its noisy sinogram. Inverse problems of this type arise in computerized tomography (CT) wherein an object is scanned from different angles using X-ray beams, and subsequently reconstructed using information about the difference in intensity before and after the beam passes through the object [71]. The forward model is given by the Radon transform. Given the material density function $\rho \in \Omega \subset \mathbb{R}^2 \to \mathbb{R}$, the

Table 3. Base parameters of the Shepp-Logan phantom. Note, $\alpha_k$ is in degrees.

| $k$ | $r_k$ | $s_k$ | $a_k$ | $b_k$ | $\alpha_k$ | $\rho_k$ |
|---|---|---|---|---|---|---|
| 1 | 0.0 | 0.0 | 0.69 | 0.92 | 0 | 1.0 |
| 2 | 0.0 | −0.0184 | 0.6624 | 0.874 | 0 | −0.8 |
| 3 | 0.22 | 0.0 | 0.11 | 0.31 | −18 | −0.2 |
| 4 | −0.22 | 0.0 | 0.16 | 0.41 | −18 | −0.2 |
| 5 | 0.0 | 0.35 | 0.21 | 0.25 | 0 | 0.1 |
| 6 | 0.0 | 0.1 | 0.046 | 0.026 | 0 | 0.1 |
| 7 | 0.0 | −0.1 | 0.046 | 0.046 | 0 | 0.1 |
| 8 | −0.08 | −0.605 | 0.046 | 0.023 | 0 | 0.1 |
| 9 | 0.0 | −0.606 | 0.023 | 0.023 | 0 | 0.1 |
| 10 | 0.06 | −0.605 | 0.023 | 0.046 | 0 | 0.1 |

Radon transform is defined as [71]

$$\mathcal{R}(\rho; t, \phi) = \int_{l_{t,\phi}} \rho \, \mathrm{d}l, \tag{27}$$

where $l_{t,\phi}$ is the line that traverses through the object at a distance of $t$ from the center and an inclination of $\phi$. Therefore, given an input phantom image $\boldsymbol{x} \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{p}}$, the forward model is

$$\boldsymbol{y} = F(\boldsymbol{x}) \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{p}}, \tag{28}$$

which is a linear transformation of $\boldsymbol{x}$. In Eq. (28),

$$\boldsymbol{y}_{i,j} = \mathcal{R}^h(\boldsymbol{x}; t_i, \phi_j), \;\; t_i = \frac{i}{n_\mathrm{p}}, \phi_j = \frac{j}{\pi} \quad \forall i, j \in \{1, 2, \ldots, n_\mathrm{p}\}, \tag{29}$$

and $\mathcal{R}^h$ is the discrete Radon transform [71]. The output $\boldsymbol{y}$ is commonly known as a sinogram. In this example, we consider input images of size 128×128, i.e., $n_\mathrm{p} = 128$. Additionally, every input image is scanned at 128 uniformly spaced angles between $0^\circ$ and $180^\circ$ with 128 detectors; thus, $n_\mathcal{Y} = 128 \times 128$. We use the `torch-radon` package [72] to compute Radon transforms. The prior dataset for this example consists of Shepp-Logan head phantoms [36]. Every phantom consists of ten ellipses, where each ellipse has constant density. Let the $k^{\text{th}}$ ellipse $E_k$ be centered at $(r_k, s_k)$, with semi-axis lengths $a_k$ and $b_k$, angle of inclination $\alpha_k$ (in degrees) and density $\rho_k$.

The density of the phantom at any coordinate $(r, s)$ is

$$\rho(r,s) = \Sigma_{k=1}^{10} C_k(r,s), \quad \text{where } C_k(r,s) = \begin{cases} \rho_k & \text{if } (r,s) \in E_k \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

Table 3 provides details of the nominal values of the base parameters of the ellipses, which we adapt from Toft [36]. Following Patel et al. [19], we generate new phantoms by perturbing the base parameters of every ellipse $\tilde{E}_k$ as follows:

$$\begin{aligned} &\tilde{r}_k = r_k + 0.005\xi_{k,1}, \;\; \tilde{s}_k = s_k + 0.005\xi_{k,2}, \;\; \tilde{a}_k = a_k + 0.005\xi_{k,3}, \\ &\tilde{b}_k = b_k + 0.005\xi_{k,4}, \;\; \tilde{\alpha}_k = \alpha_k + 2.5\xi_{k,5}, \;\; \tilde{\rho}_k = \rho_k + 0.0005\xi_{k,6} \end{aligned} \tag{31}$$

where $\left\{ \{\xi_{k,i}\}_{i=1}^{i=6} \right\}_{k=1}^{k=10}$ are uniform random variables in $[-1, 1]$. Now, the density of the perturbed phantom $\tilde{\rho}$ is

$$\tilde{\rho}(r,s) = \max\left(0, \min\left(1, \Sigma_{k=1}^{10} \tilde{C}_k(r,s)\right)\right), \quad \text{where } \tilde{C}_k(r,s) = \begin{cases} \tilde{\rho}_k & \text{if } (r,s) \in \tilde{E}_k \\ 0 & \text{otherwise} \end{cases} \tag{32}$$

ensures that the material density $\tilde{\rho}$ at any point is bounded within 0 (air cavity) and 1 (bone). We obtain $n_{\text{data}} = 8000$ discrete phantom images by evaluating Eq. (32) on a grid of size $128 \times 128$. The resulting image is further subject to a transformation that translates it by $n_{\text{h}}$ and $n_{\text{v}}$ pixels in the horizontal and vertical direction, respectively, and rotates it by an angle $\beta$. We assume that $n_{\text{h}}$ and $n_{\text{v}}$ take integer values uniformly between $-8$ and 8, i.e., $n_{\text{h}}, n_{\text{v}} \sim \mathcal{U}\{-8, -7, \ldots, 7, 8\}$, while the random variable $\beta \in \mathcal{U}(-20°, 20°)$.

We show four realizations from the prior dataset in Fig. 5(a). We use another realization, not part of the prior dataset and shown in Fig. 6, to generate the synthetic measurements for this example. We simulate noisy sinogram data by adding zero-mean Gaussian noise with variance $\sigma_\eta^2$ to the noise-free sinogram. We vary the variance of the measurement noise $\sigma_\eta^2 \in \{1, 10, 50\}$ to test the robustness of GAN-Flow to varying levels of noise in the measurement. The noise characteristics of CT data is Gaussian when the photon counts are large [73]. A Gaussian noise

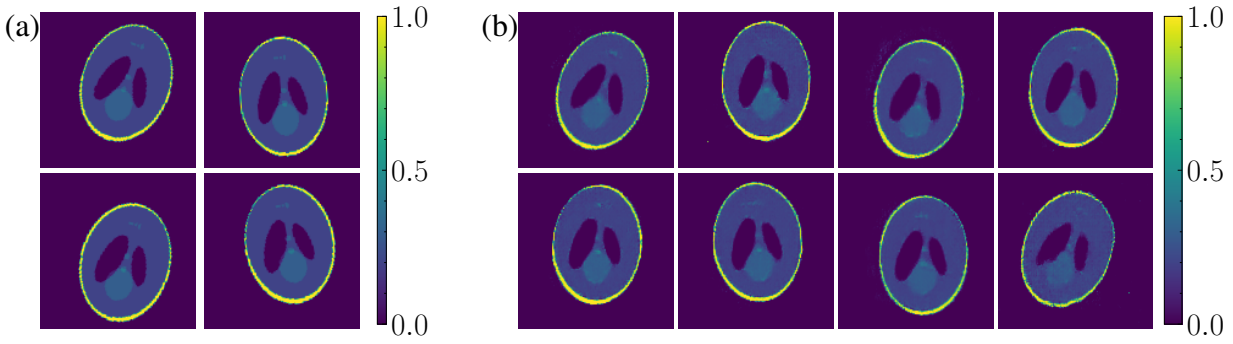


Figure 5. (a) Different phantoms from the prior dataset (b) Phantoms generated from the WGAN-GP prior.

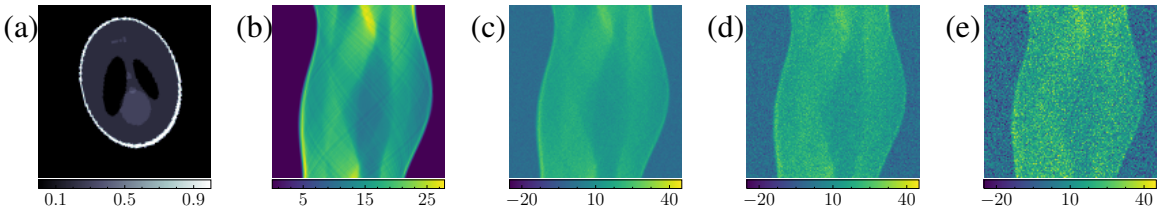


Figure 6. (a) Test phantom and (b) corresponding noise-free sinogram. Noisy sinograms after adding zero-mean Gaussian noise with variance (c) $\sigma_\eta^2 = 1$ (d) $\sigma_\eta^2 = 10$ and (e) $\sigma_\eta^2 = 50$.

model is useful even in the small photon count regime [73]. In this work, we limit our exposition to Gaussian noise. We remark that we can accommodate any noise model by suitably modifying the likelihood term in Eq. (21).

As in the previous example, we first train a WGAN-GP with latent dimensionality $n_{\mathcal{Z}} = 60$ to approximate the prior distribution[2]. We provide details of the generator and critic architectures used in this study in Appendix B1, while Table B1 provides details of other training hyper-parameters. Fig. 5(b) shows some realizations from the trained WGAN-GP prior. Next, we train a normalizing model that has 256 planar flow layers. We train the normalizing flow model for 15,000 epochs with a batch size of 32; we list other hyper-parameters associated with the training in Table B1. Thus, training the normalizing flow model requires $4.8\times10^5$ forward model evaluations. After this we obtain 30,000 realizations from the posterior distribution and use them to estimate the posterior mean and standard deviation. Fig. 7 shows the posterior statistics estimated

[2]Like the previous example, we vary $n_{\mathcal{Z}} \in \{5, 10, 20, 40, 60, 80, 100\}$ and choose $n_{\mathcal{Z}} = 60$ since the RMSE between the corresponding posterior mean and the test phantom is is either smallest or close to being the smallest; see Appendix C2

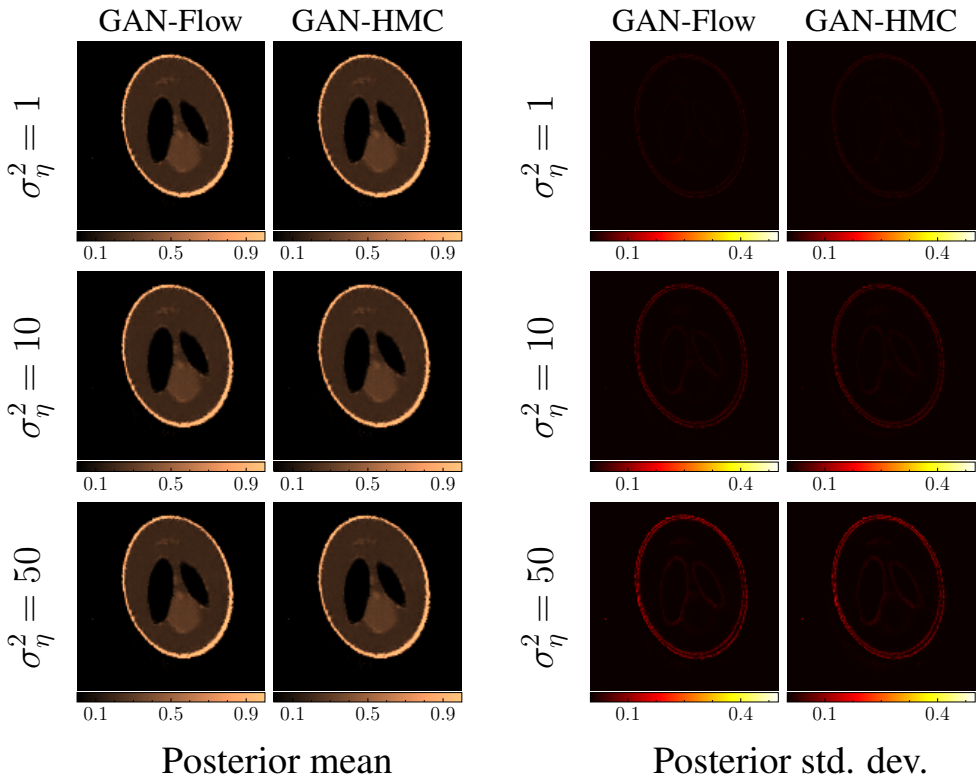


Figure 7. Estimated posterior mean (left) and standard deviation (right) obtained using GAN-Flow (left column) and GAN-HMC (right column) for the inverse Radon transform problem at various levels of measurement noise.

using GAN-Flow and GAN-HMC. For comparison, we also obtain a sample of size 60,000 from the posterior distribution using HMC, discard the first 30,000 realizations considering burn-in, and then estimate the posterior statistics. For this example, we run HMC with 10 steps and an initial step size of 0.01. With this setting, sampling from the latent posterior using HMC requires $6\times10^5$ forward model evaluations. The posterior statistics estimated using GAN-Flow and GAN-HMC are qualitatively similar and shows elevated uncertainty around the edges of the phantom. The uncertainty increases as the noise in the measurement increases, which is also expected. We compute the RMSE and structural similarity index metric (SSIM) [74] between the posterior mean and the test phantom for both GAN-Flow and GAN-HMC and report those values in Table 4. Quantitatively, both GAN-Flow and GAN-HMC provide similar reconstructions of the test phantom, which is consistent with Fig. 7. The results confirm that GAN-Flow is robust with respect to the level of measurement noise. We also perform additional experiments on another test phantom;

see Figs. C8 and C9 in Appendix C2 for those results. The performance of GAN-Flow is similar across both test phantoms.

Table 4. RMSE and SSIM of the posterior mean reconstruction of the test phantom at different levels of measurement noise.

| Method | RMSE | | | SSIM | | |
|---|---|---|---|---|---|---|
| | $\sigma_\eta^2 = 1$ | $\sigma_\eta^2 = 10$ | $\sigma_\eta^2 = 50$ | $\sigma_\eta^2 = 1$ | $\sigma_\eta^2 = 10$ | $\sigma_\eta^2 = 50$ |
| GAN-Flow | 0.041 | 0.042 | 0.045 | 0.968 | 0.964 | 0.963 |
| GAN-HMC | 0.041 | 0.043 | 0.045 | 0.968 | 0.964 | 0.962 |

### *4.3. Phase retrieval*

The final application we consider concerns phase retrieval, which involves the recovery of an object from the magnitude of its Fourier transform [75, 76]. Phase retrieval inverse problems are ubiquitous in many areas of science and engineering [77, 78, 79, 80, 81]. More specifically, we consider the phase retrieval problem of recovering an object from sparse measurements of the magnitude of its Fourier transform. We undersample the measurements to simulate accelerated measurement acquisition paradigms. The forward model for the phase retrieval problem we consider is given by:

$$\boldsymbol{y} = |\boldsymbol{M}\boldsymbol{F}\boldsymbol{x}| + \boldsymbol{\eta}, \tag{33}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{p}}$ is the object of interest discretized as an image of $n_\mathrm{p} \times n_\mathrm{p}$ pixels, $\boldsymbol{F}$ is the two-dimensional discrete Fourier transform (DFT), $|\cdot|$ computes the magnitude component wise, $\boldsymbol{M}$ is a binary mask that undersamples the Fourier magnitude measurements, and $\boldsymbol{\eta}$ is the measurement noise. In vector form, the effective dimensionality $n_\mathcal{Y}$ of $\boldsymbol{y}$ depends on the undersampling ratio $r$ (also known as acceleration factor [38]), i.e., $n_\mathcal{Y} = n_\mathcal{X}/r$.

For this example, the prior dataset comprises of a subsample of the single coil knee scans from the publicly available NYU fastMRI training dataset [38, 37]. Similar to Kelkar et al. [82], we prepare the prior dataset in the following way. The training dataset contains a total of 973 volumes and 34,742 slices. Each slice corresponds to an *emulated* single coil complex-valued Fourier space ($k$-space) MRI measurement; the single coil data is emulated by linearly combining multi-coil $k$-space data [38]. For every slice, the fastMRI initiative also provides a corresponding

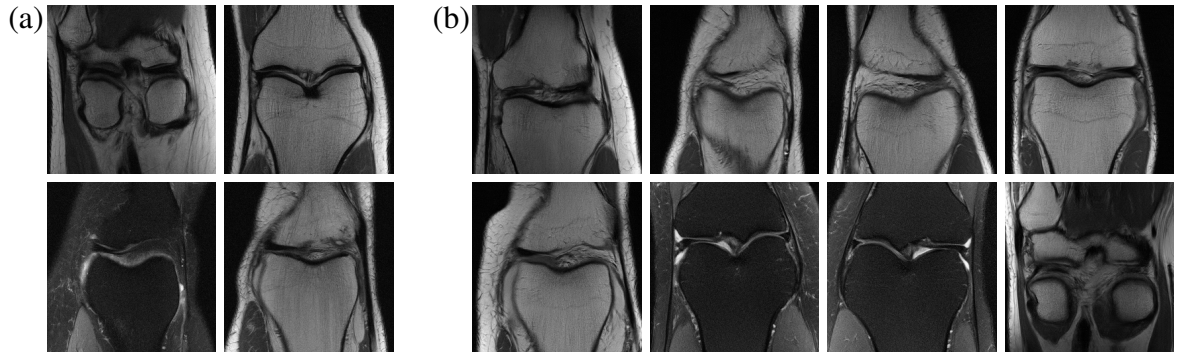


Figure 8. (a) Different knee slices from the prior dataset (b) Knee slices generated from the WGAN-GP prior.

slice of the knee for that volume computed from the emulated single coil measurement using the root-sum-of-squares method. We discard the first five reconstructed knee slices of every volume, center crop the rest into images of size $256 \times 256$ and then randomly divide them up into training and test sets. In total, the training and test set of the WGAN-GP contains 29,877 and 6,140 knee slices, respectively. The aforementioned training set is the prior dataset for this example and it contains $n_{\text{data}} = 29,877$ knee slices. Moreover, the ambient dimensionality $n_{\mathcal{X}} = 256 \times 256$ for this problem. Fig. 8(a) shows four typical knee slices from the prior dataset. We emphasize that, although we use knee slices from reconstructed MRI data, the forward model is nonlinear and given by Eq. (33).

We use three realizations from the test set as the test case for this problem. The test cases are shown in the top row of Fig. 9. Right below them, we plot the natural logarithm of the corresponding noise free $k$-space magnitude data to which we subsequently add zero-mean Gaussian noise with standard deviation equal to 0.04% of the zero-frequency amplitude of the two-dimensional DFT [28]. We also consider two types of Cartesian undersampling masks to reflect realistic scenarios where an object must be reconstructed from sparse measurements. Specifically, we consider two masks that yield four-fold and eight-fold accelerations. Following Zbontar et al. [38], the undersampling masks include 8% and 4% of the central region of the $k$-space when the acceleration factor $r = 4$ and 8, respectively. The remaining $k$-space lines are uniformly sampled with probability such that the desired acceleration can be achieved. As common in practice, we omit $k$-space magnitude measurements in the phase direction, i.e., the undersampling masks consist of vertical

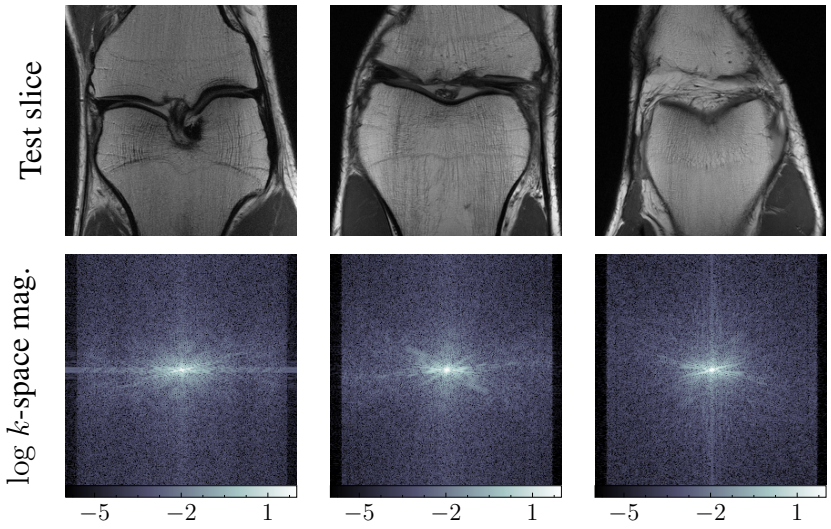


Figure 9. Test slices used for phase retrieval (top row) and corresponding noise free logarithm of Fourier ($k$-space) magnitudes (bottom row). We apply the masks, shown in Fig. 10, to the noisy Fourier magnitudes to generate the synthetic measurements for this example.

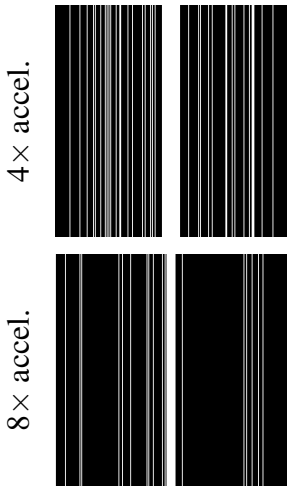


Figure 10. Cartesian undersampling masks with four- and eight-fold accelerations.

bands. Fig. 10 shows the two masks considered in this example.

In this example, the WGAN-GP prior is trained using the progressive growing of GAN (ProGAN) method [40]. Not only does the ProGAN training method stabilize the training of GANs designed to synthesize large images, but it also makes the training more efficient. In the ProGAN training methodology, learning commences from a coarse scale wherein the generator learns to synthesize, and the critic learns to discriminate, low resolution images, say of size $4 \times 4$. Over stages of increasing resolution, going from $4 \times 4$ to $8 \times 8$ and ultimately to $256 \times 256$, new layers are added to the generator and the critic, as the generator learns to synthesize, and the critic learns to discriminate, finer scale details. We adopt the implementation of ProGANs from [83] and choose the latent space dimensionality $n_{\mathcal{Z}} = 512$ following previous works [82, 40]. Further details about the WGAN-GP model, ProGAN training methodology, and associated training hyper-parameters may be found in Appendix B and Table B1. Samples from the trained WGAN-GP prior are shown in Fig. 8(b). We note that the WGAN-GP model is frozen for all subsequent steps.

The normalizing flow model for this problem consists of 16 affine-coupling flow layers with

activation normalization [32]. See Appendix B2 for more details about the scale and shift networks of the affine coupling layers. In this example, we train the normalizing flow model for 50,000 epochs and a batch size of 32. We train the normalizing flow model for every combination of test slice and undersampling mask. Subsequently, for each combination of test slice and mask, we obtain 10,000 samples from the latent posterior distribution to estimate the posterior pixel-wise mean and standard mean. Figs. 11 and 12 show the posterior mean, posterior standard-deviation, and the absolute error of the posterior mean reconstruction for the four- and eight-fold acceleration, respectively. We compute the RMSE and SSIM between the posterior mean reconstruction and the ground truth knee slices and report these values in Table 5. From Figs. 11 and 12 and Table 5, we observe that the reconstruction is satisfactory. However, the reconstruction of test slice 1 is comparatively better than those of test slices 2 and 3. This indicates that the reconstruction of some knee slices, like test slice 1, which probably lies in the typical set of the generator's latent space (range of $G^*$ [84]), can be better than atypical test slices. Moreover, for test slice 2, the posterior pixel-wise standard deviation when the acceleration factor $r = 4$ is marginally larger in comparison to the case when $r = 8$; see the second row third column of Figs. 11 and 12. The result above needs further investigation since a reduction in uncertainty as the number of measurements reduces is counter-intuitive.

Significantly, this example showcases the efficacy of GAN-Flow compared to methods involving MCMC simulations. In this study, we attempted to carry out inference using GAN-HMC. However, we were unsuccessful in obtaining convergence of the Markov chains after varying the number of leapfrog steps and burn-in time. We posit that this may be due to the relatively large latent dimensionality in this study.

Table 5. RMSE and SSIM of the of the posterior mean reconstruction of the test slices for different accelerations.

| Test slice | 4× acceleration | | 8× acceleration | |
|---|---|---|---|---|
| | RMSE | SSIM | RMSE | SSIM |
| Slice 1 | 0.0070 | 0.6185 | 0.0072 | 0.6142 |
| Slice 2 | 0.0238 | 0.4802 | 0.0246 | 0.4587 |
| Slice 3 | 0.0206 | 0.5556 | 0.0217 | 0.5702 |

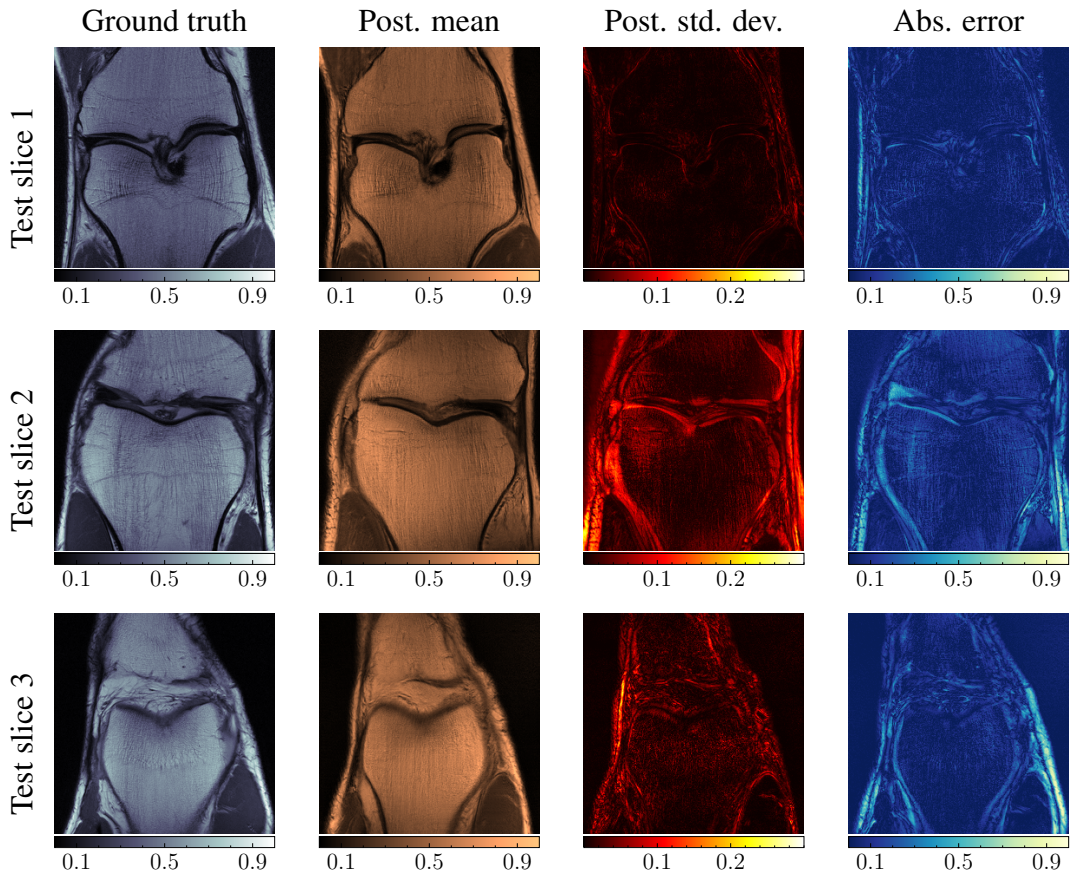


Figure 11. Posterior pixel-wise standard mean (second column) and posterior pixel-wise standard deviation (third column) for various test cases when acceleration factor $r = 4$. The first column shows the ground truth slices for reference. The last column shows the absolute error between the posterior mean and the ground truth.

## 5. Conclusions

Bayesian inference is widely applicable but its application is challenging, especially, in cases where the inverse problem is high-dimensional and prior information is qualitative in nature. In this work, we propose GAN-Flow, a dimension-reduced variational approach to solving large-scale inverse problems. GAN-Flow combines together two types of generative models — GANs and normalizing flows. The former is used to form an informative data-driven prior with the generator providing a map between a low-dimensional latent space and the high-dimensional ambient space. Normalizing flows are used to solve the inverse problem variationally in the low-dimensional latent space, made possible due to an invertible map that transforms the prior distribution in the latent

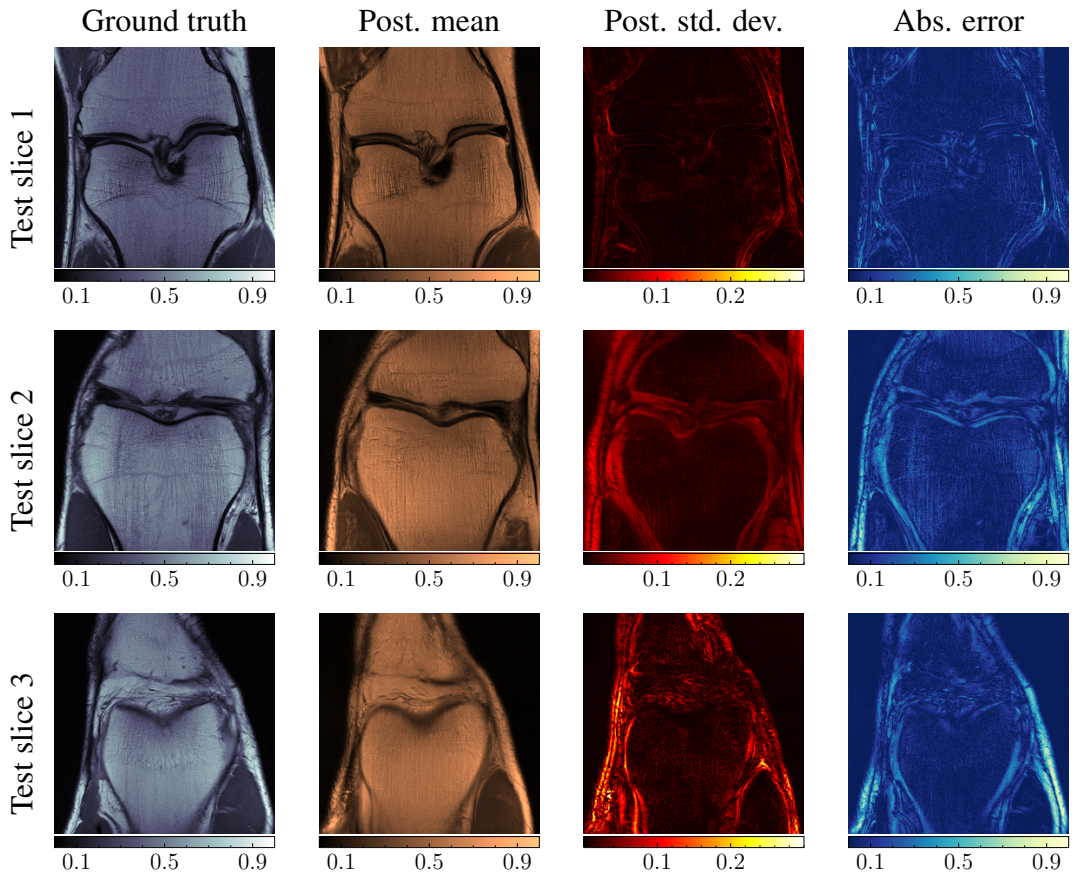


Figure 12. Posterior pixel-wise standard mean (second column) and posterior pixel-wise standard deviation (third column) for various test cases when acceleration factor $r = 8$. The first column shows the ground truth slices for reference. The last column shows the absolute error between the posterior mean and the ground truth.

space into the posterior distribution in the latent space. The low-dimensional latent posterior can be sampled, without evaluating the underlying forward model, to obtain samples from the high-dimensional ambient space. We have also shown how GAN-Flow can be used to estimate statistics of the ambient posterior distribution. We have used GAN-Flow to solve three physics-based inverse problems that include various challenging scenarios. In particular, the application to phase imaging shows that GAN-Flow can handle challenging prior information, nonlinear forward models, and very large-scale inverse problems. The extension of GAN-Flow to black-box forward models that are incompatible with automatic differentiation is a promising research direction that will make GAN-Flow more widely applicable. Another interesting research direction will be the

application of GAN-Flow to the inference of physics, i.e., terms in the governing equation of an observed phenomena.

## 6. Acknowledgments

This work was carried out while the first author was a graduate student in the Sonny Astani Department of Civil & Environmental Engineering at the University of Southern California. Agnimitra Dasgupta acknowledges support from the University of Southern California through a Provost's Ph.D. Fellowship. Agnimitra Dasgupta and Erik A Johnson also gratefully acknowledge the support of this work by the National Science Foundation through award CMMI 16-63667. Assad A Oberai gratefully acknowledges support from ARO, USA grant W911NF2010050. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF or USC. The authors also acknowledge the Center for Advanced Research Computing (CARC, carc.usc.edu) at the University of Southern California for providing computing resources that have contributed to the research results reported within this publication.

## References

[1] D. Calvetti, E. Somersalo, Inverse problems: From regularization to Bayesian inference, Wiley Interdisciplinary Reviews: Computational Statistics 10 (2018) e1427.

[2] C. Robert, G. Casella, A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data, Statistical Science 26 (2011) 102–115.

[3] T. Cui, J. Martin, Y. M. Marzouk, A. Solonen, A. Spantini, Likelihood-informed dimension reduction for nonlinear inverse problems, Inverse Problems 30 (2014) 114015.

[4] M. Betancourt, A conceptual introduction to Hamiltonian Monte Carlo, arXiv preprint arXiv:1701.02434 (2017).

[5] R. M. Neal, MCMC using Hamiltonian dynamics, Handbook of Markov chain Monte Carlo 2 (2011) 2.

[6] M. Girolami, B. Calderhead, Riemann manifold Langevin and Hamiltonian Monte Carlo methods, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73 (2011) 123–214.

[7] T. Cui, X. T. Tong, A unified performance analysis of likelihood-informed subspace methods, arXiv preprint arXiv:2101.02417 (2021).

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

[9] T. A. El Moselhy, Y. M. Marzouk, Bayesian inference with optimal maps, Journal of Computational Physics 231 (2012) 7815–7850.

[10] A. Andrle, N. Farchmin, P. Hagemann, S. Heidenreich, V. Soltwisch, G. Steidl, Invertible neural networks versus MCMC for posterior reconstruction in grazing incidence X-ray fluorescence, in: International Conference on Scale Space and Variational Methods in Computer Vision, Springer, 2021, pp. 528–539.

[11] M. Brennan, D. Bigoni, O. Zahm, A. Spantini, Y. Marzouk, Greedy inference with structure-exploiting lazy maps, Advances in Neural Information Processing Systems 33 (2020) 8330–8342.

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

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Advances in Neural Information Processing Systems 27 (2014).

[14] D. Rezende, S. Mohamed, Variational inference with normalizing flows, in: International Conference on Machine Learning, PMLR, 2015, pp. 1530–1538.

[15] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference., Journal of Machine Learning Research 22 (2021) 1–64.

[16] I. Kobyzev, S. J. Prince, M. A. Brubaker, Normalizing flows: An introduction and review of current methods, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (2020) 3964–3979.

[17] D. P. Kingma, M. Welling, Auto-encoding variational Bayes, arXiv preprint arXiv:1312.6114 (2013).

[18] S. Bond-Taylor, A. Leach, Y. Long, C. G. Willcocks, Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models, arXiv preprint arXiv:2103.04922 (2021).

[19] D. V. Patel, D. Ray, A. A. Oberai, Solution of physics-based bayesian inverse problems with deep generative priors, Computer Methods in Applied Mechanics and Engineering 400 (2022) 115428.

[20] P. Bohra, T.-a. Pham, J. Dong, M. Unser, Bayesian inversion for nonlinear imaging models using deep generative priors, arXiv preprint arXiv:2203.10078 (2022).

[21] J. Adler, O. Öktem, Deep bayesian inversion, arXiv preprint arXiv:1811.05910 (2018).

[22] D. Ray, H. Ramaswamy, D. V. Patel, A. A. Oberai, The efficacy and generalizability of conditional GANs for posterior inference in physics-based inverse problems, Numerical Algebra, Control and Optimization (2022).

[23] D. Ray, J. Murgoitio-Esandi, A. Dasgupta, A. A. Oberai, Solution of physics-based inverse problems using conditional generative adversarial networks with full gradient penalty, arXiv preprint arXiv:2306.04895 (2023).

[24] K. Sohn, H. Lee, X. Yan, Learning structured output representation using deep conditional generative models, Advances in Neural Information Processing Systems 28 (2015).

[25] H. Goh, S. Sheriffdeen, J. Wittmer, T. Bui-Thanh, Solving Bayesian inverse problems via variational autoencoders, arXiv preprint arXiv:1912.04212 (2019).

[26] T. Sahlström, T. Tarvainen, Utilizing variational autoencoders in the Bayesian inverse problem of photoacoustic tomography, SIAM Journal on Imaging Sciences 16 (2023) 89–110.

[27] A. Dasgupta, E. A. Johnson, REIN: Reliability estimation via importance sampling with normalizing flows, Reliability Engineering and System Safety (Under review).

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

[29] A. Dasgupta, Z. W. Di, Uncertainty quantification for ptychography using normalizing flows, in: Fourth Workshop on Machine Learning and the Physical Sciences, 2021.

[30] G. A. Padmanabha, N. Zabaras, Solving inverse problems using conditional invertible neural networks, Journal of Computational Physics 433 (2021) 110194.

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

[32] D. P. Kingma, P. Dhariwal, GLOW: Generative flow with invertible $1 \times 1$ convolutions, Advances in Neural Information Processing Systems 31 (2018).

[33] J. Lampinen, A. Vehtari, Bayesian approach for neural networks—review and case studies, Neural networks 14 (2001) 257–274.

[34] B.-H. Tran, S. Rossi, D. Milios, M. Filippone, All you need is a good functional prior for Bayesian deep learning, Journal of Machine Learning Research 23 (2022) 1–56.

[35] D. V. Patel, A. A. Oberai, GAN-based priors for quantifying uncertainty in supervised learning, SIAM/ASA Journal on Uncertainty Quantification 9 (2021) 1314–1343.

[36] P. Toft, The Radon transform, Ph.D. thesis, Technical University of Denmark, 1996.

[37] F. Knoll, J. Zbontar, A. Sriram, M. J. Muckley, M. Bruno, A. Defazio, M. Parente, K. J. Geras, J. Katsnelson, H. Chandarana, Z. Zhang, M. Drozdzalv, A. Romero, M. Rabbat, P. Vincent, J. Pinkerton, D. Wang, N. Yakubova, E. Owens, C. L. Zitnick, M. P. Recht, D. K. Sodickson, Y. W. Lui, fastMRI: A publicly available raw k-space and DICOM dataset of knee images for accelerated MR image reconstruction using machine learning, Radiology: Artificial Intelligence 2 (2020) e190007.

[38] 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, Y. W. Lui, fastmri: An open dataset and benchmarks for accelerated MRI, arXiv preprint arXiv:1811.08839 (2018).

[39] H. Zhang, I. Goodfellow, D. Metaxas, A. Odena, Self-attention generative adversarial networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 7354–7363.

[40] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of GANs for improved quality, stability, and variation, in: International Conference on Learning Representations, 2018.

[41] L. Dinh, D. Krueger, Y. Bengio, NICE: Non-linear independent components estimation, arXiv preprint arXiv:1410.8516 (2014).

[42] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using Real NVP, arXiv preprint arXiv:1605.08803 (2016).

[43] E. Cramer, A. Mitsos, R. Tempone, M. Dahmen, Principal component density estimation for scenario generation using normalizing flows, Data-Centric Engineering 3 (2022).

[44] E. Cramer, F. Rauh, A. Mitsos, R. Tempone, M. Dahmen, Nonlinear isometric manifold learning for injective normalizing flows, arXiv preprint arXiv:2203.03934 (2022).

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

[46] J. Brehmer, K. Cranmer, Flows for simultaneous manifold learning and density estimation, Advances in Neural Information Processing Systems 33 (2020) 442–453.

[47] Y. Hong, U. Hwang, J. Yoo, S. Yoon, How generative adversarial networks and their variants work: An overview, ACM Computing Surveys 52 (2019) 1–43.

[48] X. Yi, E. Walia, P. Babyn, Generative adversarial network in medical imaging: A review, Medical Image Analysis 58 (2019) 101552.

[49] A. Jabbar, X. Li, B. Omar, A survey on generative adversarial networks: Variants, applications, and training, ACM Computing Surveys 54 (2021) 1–49.

[50] A. K. Dhaka, A. Catalina, M. Welandawe, M. R. Andersen, J. Huggins, A. Vehtari, Challenges and opportunities in high dimensional variational inference, Advances in Neural Information Processing Systems 34 (2021) 7787–7798.

[51] Y. Marzouk, T. Moselhy, M. Parno, A. Spantini, An introduction to sampling via measure transport, arXiv preprint arXiv:1602.05023 (2016).

[52] A. Gelman, C. R. Shalizi, Philosophy and the practice of Bayesian statistics, British Journal of Mathematical and Statistical Psychology 66 (2013) 8–38.

[53] X. Meng, L. Yang, Z. Mao, J. del Águila Ferrandis, G. E. Karniadakis, Learning functional priors and posteriors from data and physics, Journal of Computational Physics 457 (2022) 111073.

[54] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 214–223.

[55] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. C. Courville, Improved training of Wasserstein GANs,

Advances in Neural Information Processing Systems 30 (2017).

[56] J. Thickstun, Kantorovich-Rubinstein Duality, 2019. URL: https://courses.cs.washington.edu/courses/cse599i/20au/resources/L12_duality.pdf.

[57] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, GANs trained by a two time-scale update rule converge to a local Nash equilibrium, Advances in Neural Information Processing Systems 30 (2017).

[58] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training GANs, Advances in Neural Information Processing Systems 29 (2016).

[59] A. Bhaduri, A. Gupta, A. Olivier, L. Graham-Brady, An efficient optimization based microstructure reconstruction approach with multiple loss functions, Computational Materials Science 199 (2021) 110709.

[60] D. P. Kingma, J. Ba, ADAM: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[61] Y. Wang, F. Liu, D. E. Schiavazzi, Variational inference with NoFAS: Normalizing flow with adaptive surrogate for computationally expensive models, Journal of Computational Physics 467 (2022) 111454.

[62] A. Mallasto, G. Montúfar, A. Gerolin, How well do WGANs estimate the Wasserstein metric?, arXiv preprint arXiv:1910.03875 (2019).

[63] J. Stanczuk, C. Etmann, L. M. Kreusser, C.-B. Schönlieb, Wasserstein GANs work because they fail (to approximate the Wasserstein distance), arXiv preprint arXiv:2103.01678 (2021).

[64] Y. Yao, A. Vehtari, D. Simpson, A. Gelman, Yes, but did it work?: Evaluating variational inference, in: International Conference on Machine Learning, PMLR, 2018, pp. 5581–5590.

[65] Y. Yao, A. Vehtari, A. Gelman, Stacking for non-mixing Bayesian computations: The curse and blessing of multimodal posteriors, Journal of Machine Learning Research 23 (2022).

[66] D. Patel, A. A. Oberai, Bayesian inference with generative adversarial network priors, arXiv preprint arXiv:1907.09987 (2019).

[67] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.

[68] A. D. Cobb, B. Jalaian, Scaling Hamiltonian Monte Carlo inference for Bayesian neural networks with symmetric splitting, in: Uncertainty in Artificial Intelligence, PMLR, 2021, pp. 675–685.

[69] J. Cannon, S. P. Esteva, An inverse problem for the heat equation, Inverse Problems 2 (1986) 395.

[70] J. R. Cannon, P. DuChateau, Structural identification of an unknown source term in a heat equation, Inverse Problems 14 (1998) 535.

[71] T. G. Feeman, The mathematics of medical imaging: A beginner's guide, Springer, 2015.

[72] M. Ronchetti, torchradon: Fast differentiable routines for computed tomography, arXiv preprint arXiv:2009.14788 (2020).

[73] J. Wang, H. Lu, Z. Liang, D. Eremina, G. Zhang, S. Wang, J. Chen, J. Manzione, An experimental study on the noise properties of X-ray CT sinogram data in radon space, Physics in Medicine & Biology 53 (2008) 3327.

[74] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Image Processing 13 (2004) 600–612.

[75] M. V. Klibanov, P. E. Sacks, A. V. Tikhonravov, The phase retrieval problem, Inverse Problems 11 (1995) 1.

[76] J. Rosenblatt, Phase retrieval, Communications in Mathematical Physics 95 (1984) 317–343.

[77] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, M. Segev, Phase retrieval with application to optical imaging: A contemporary overview, IEEE Signal Processing Magazine 32 (2015) 87–109.

[78] J. Miao, R. L. Sandberg, C. Song, Coherent x-ray diffraction imaging, IEEE Journal of Selected Topics in Quantum Electronics 18 (2011) 399–410.

[79] M. H. Maleki, A. J. Devaney, Phase-retrieval and intensity-only reconstruction algorithms for optical diffraction tomography, Journal of the Opical Society of America A 10 (1993) 1086–1092.

[80] R. P. Millane, Phase retrieval in crystallography and optics, Journal of the Optical Society of America A 7 (1990) 394–411.

[81] J. R. Fienup, J. C. Marron, T. J. Schulz, J. H. Seldin, Hubble Space Telescope characterized by using phase-retrieval algorithms, Applied Optics 32 (1993) 1747–1767.

[82] V. A. Kelkar, S. Bhadra, M. A. Anastasio, Compressible latent-space invertible networks for generative model-constrained image reconstruction, IEEE Transactions on Computational Imaging 7 (2021) 209–223.

[83] A. Karnewar, pro_gan_pytorch, https://github.com/akanimax/pro_gan_pytorch, 2018.

[84] A. Bora, A. Jalal, E. Price, A. G. Dimakis, Compressed sensing using generative models, in: International Conference on Machine Learning, PMLR, 2017, pp. 537–546.

## Appendix A. Derivation of the loss function $\mathcal{L}_{\text{NF}}$ for training the normalizing flow model in GAN-Flow

Let $\boldsymbol{z} \sim \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$ and $\tilde{\boldsymbol{z}}$ be two random vectors, both $\in \Omega_{\mathcal{Z}}$, and the diffeomorphism $H(\cdot;\boldsymbol{\psi}) : \Omega_{\mathcal{Z}} \to \Omega_{\mathcal{Z}}$ such that $\tilde{\boldsymbol{z}} = H(\boldsymbol{z};\boldsymbol{\psi})$. Let $\tilde{\boldsymbol{z}} \sim \mathrm{p}_{\tilde{\mathcal{Z}}}(\tilde{\boldsymbol{z}})$, which can be computed using Eq. (6) as follows:

$$\mathrm{p}_{\tilde{\mathcal{Z}}}(\tilde{\boldsymbol{z}}) = H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi}) = \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})|\det \nabla_{\boldsymbol{z}} H(\boldsymbol{z};\boldsymbol{\psi})|^{-1}. \tag{A.1}$$

For any function $a(\cdot) : \Omega_{\mathcal{Z}} \to \Omega_{\mathcal{Z}}$, the change-of-variables theorem of multivariable calculus states:

$$\begin{aligned}\int_{\Omega_{\mathcal{Z}}} a(\tilde{\boldsymbol{z}})\, \mathrm{p}_{\tilde{\mathcal{Z}}}(\tilde{\boldsymbol{z}})\, \mathrm{d}\tilde{\boldsymbol{z}} &= \int_{\Omega_{\mathcal{Z}}} a(\tilde{\boldsymbol{z}})\, \underbrace{\left[\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})|\det \nabla_{\boldsymbol{z}} H(\boldsymbol{z};\boldsymbol{\psi})|^{-1}\right]}_{\mathrm{p}_{\tilde{\mathcal{Z}}}(\tilde{\boldsymbol{z}}) \text{ from Eq. (A.1)}}\, \mathrm{d}\tilde{\boldsymbol{z}} \\ &= \int_{\Omega_{\mathcal{Z}}} a(\tilde{\boldsymbol{z}})\, \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})\, \underbrace{\left[|\det \nabla_{\boldsymbol{z}} H(\boldsymbol{z};\boldsymbol{\psi})|^{-1}\mathrm{d}\tilde{\boldsymbol{z}}\right]}_{\mathrm{d}\boldsymbol{z}} = \int_{\Omega_{\mathcal{Z}}} a\big(H(\boldsymbol{z};\boldsymbol{\psi})\big)\, \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z}.\end{aligned} \tag{A.2}$$

On recognizing that $\int_{\Omega_{\mathcal{Z}}} a(\tilde{\boldsymbol{z}})\, \mathrm{p}_{\tilde{\mathcal{Z}}}(\tilde{\boldsymbol{z}})\, \mathrm{d}\tilde{\boldsymbol{z}}$ is nothing but $\mathbb{E}_{\boldsymbol{z}\sim H_{\#}\mathrm{p}_{\mathcal{Z}}}\left[a(\boldsymbol{z})\right]$, we can write

$$\mathbb{E}_{\boldsymbol{z}\sim H_{\#}\mathrm{p}_{\mathcal{Z}}}\left[a(\boldsymbol{z})\right] = \mathbb{E}_{\boldsymbol{z}\sim \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})}\left[a(H(\boldsymbol{z};\boldsymbol{\psi}))\right]. \tag{A.3}$$

Now, the reverse KL divergence between the pushforward distribution $H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi})$ and the target distribution $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}})$ is:

$$\begin{aligned}&d_{\text{KL}}\big(H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi})\|\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}})\big) \equiv \mathbb{E}_{\boldsymbol{z}\sim H_{\#}\mathrm{p}_{\mathcal{Z}}}\left[\log H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi}) - \log \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}})\right] \\ &= \int_{\Omega_{\mathcal{Z}}} \Big[\log H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi}) - \underbrace{\big(\log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{z}) + \log \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}) - \log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})\big)}_{\log \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}})}\Big] H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi})\, \mathrm{d}\boldsymbol{z} \\ &= \int_{\Omega_{\mathcal{Z}}} \Big[\log H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi}) - \log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|\boldsymbol{z}) - \log \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}) + \log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})\Big] H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi})\, \mathrm{d}\boldsymbol{z} \\ &= \int_{\Omega_{\mathcal{Z}}} \Big[\log \underbrace{\big(\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})|\det \nabla_{\boldsymbol{z}} H(\boldsymbol{z};\boldsymbol{\psi})|^{-1}\big)}_{H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi})} - \log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|H(\boldsymbol{z};\boldsymbol{\psi})) - \log \mathrm{p}_{\mathcal{Z}}(H(\boldsymbol{z};\boldsymbol{\psi})) + \log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})\Big] \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z} \\ &= \int_{\Omega_{\mathcal{Z}}} \Big[\log \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}) - \log|\det \nabla_{\boldsymbol{z}} H(\boldsymbol{z};\boldsymbol{\psi})| - \log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|H(\boldsymbol{z};\boldsymbol{\psi})) - \log \mathrm{p}_{\mathcal{Z}}(H(\boldsymbol{z};\boldsymbol{\psi})) + \log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})\Big] \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z} \\ &= \mathbb{E}_{\boldsymbol{z}\sim \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})}\Big[\log \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}) - \log|\det \nabla_{\boldsymbol{z}} H(\boldsymbol{z};\boldsymbol{\psi})| - \log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}}|H(\boldsymbol{z};\boldsymbol{\psi})) - \log \mathrm{p}_{\mathcal{Z}}(H(\boldsymbol{z};\boldsymbol{\psi})) + \log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})\Big],\end{aligned} \tag{A.4}$$

where we substitute the expression for $\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z}|\hat{\boldsymbol{y}})$ from Eq. (18) in the second line, and then use Eq. (A.1) to replace $H_{\#}\mathrm{p}_{\mathcal{Z}}(\boldsymbol{z};\boldsymbol{\psi})$ in the fourth line while changing the variables of integration as per Eqs. (A.2) and (A.3). To obtain the proposed loss function $\mathcal{L}_{\text{NF}}$ in Eq. (21) for training the normalizing flow, we simply ignore the terms $\log \mathrm{p}_{\mathcal{Z}}(\boldsymbol{z})$ and $\log \mathrm{p}_{\mathcal{Y}}(\hat{\boldsymbol{y}})$ in Eq. (A.4) since they do not depend on $\boldsymbol{\psi}$. This completes the derivation of Eq. (21) starting from Eq. (20).

## Appendix B. Details of the GAN and normalizing flow architectures of GAN-Flow and training hyper-parameters

In this section we describe the WGAN-GP and normalizing flow models used in the GAN-Flow pipeline for various inverse problems. Some of the nomenclature we use are as follows:

1. FC($n$) — Fully connected layer of width $n$.
2. Tr. Conv2D ($c_{\text{out}}$, $k$, $s$, $p$, $p_{\text{out}}$) — 2D transpose convolution layer with $c_{\text{out}}$ output channels, kernel size $(k, k)$, stride $s$, padding $p$ and output padding $p_{\text{out}}$.
3. Conv2D ($c_{\text{out}}$, $k$, $s$, $p$) — 2D convolution layer $c_{\text{out}}$ output channels, kernel size $(k, k)$, stride $s$ and padding $p$.
4. Self Attention — self-attention module [39].
5. BN, LN, PixelNorm, Mini-batch std. dev. normalization — batch, layer, pixel [40] and mini-batch standard deviation [40] normalization, respectively.
6. LReLU($\alpha$), ELU, and TanH — Leaky rectified linear unit (with parameter $\alpha$), exponential linear unit and hyperbolic tangent activation functions, respectively.
7. Up-sample $2\times$ – Up-scaling by a factor of 2 using bi-linear interpolation.
8. Down-sample $2\times$ – Average pooling over $2\times2$ patches with stride 2.

### *B1. WGAN-GP model architectures*

Table B1 lists the training hyper-parameters. Fig. B1 shows the generator and critic architecture used in the initial condition inference problem. Fig. B2 shows the generator and architecture used in the inverse Radon transform problem. Fig. B3(a) and (c) shows the generator and critic

Table B1. Training hyper-parameters for WGAN-GP and normalizing flow models of the GAN-Flow pipeline.

| Model | hyper-parameter | Inverse problem | | |
|---|---|---|---|---|
| | | Heat conduction (Section 4.1) | Radon transform (Section 4.2) | Phase imaging (Section 4.3) |
| Wasserstein GAN | Latent dimension $n_{\mathcal{Z}}$ | 5 | 60 | 512 |
| | Architecture | Fig. B1 | Fig. B2 | Fig. B3 |
| | Training epochs | 500 | 1000 | 294 |
| | Learning rate | 0.0002 | 0.001 | 0.003 |
| | Gradient penalty $\lambda$ | 10 | 10 | 10 |
| | Batch size | 64 | 100 | $128 \rightarrow 64$ |
| | $n_{\text{critic}}/n_{\text{gen}}$ | 5 | 4 | 1 |
| | Optimizer | Adam $\beta_1 = 0, \beta_2 = 0.99$ | Adam $\beta_1 = 0.5, \beta_2 = 0.99$ | Adam $\beta_1 = 0, \beta_2 = 0.99$ |
| Normalizing flow | Type of flow model layer | Planar | Planar | Affine coupling |
| | Number of flow layers $n_{\text{f}}$ | 64 | 256 | 16 |
| | Training epochs | 5000 | 15000 | 50000 |
| | Learning rate | 0.002 | 0.002 | 0.001 |
| | Batch size | 32 | 32 | 32 |
| | Optimizer | Adam $\beta_1 = 0.9, \beta_2 = 0.999$ | Adam $\beta_1 = 0.9, \beta_2 = 0.999$ | Adam $\beta_1 = 0.9, \beta_2 = 0.999$ |

architectures, respectively, we use for the phase retrieval problem. The generator comprises convolution blocks that are shown in Fig. B3(b). Similarly, the critic is made of a convolution block that is denoted as 'Dis. Convolution Block' in Fig. B3(c) and shown in Fig. B3(b). In this study, we use the Progressive growing of GAN methodology to train the generator and critic networks. We briefly summarize the ProGAN method here, and refer [40] to interested readers for more details.

### *B1.1. Progressive growing of GANs*

In the ProGAN methodology, both generator and critic are trained synchronously to synthesize images starting from size 4×4 up until 256×256. For instance, at the first stage, when the GAN is learning to synthesize images of size 4×4, only the first three layers (after the inputs are reshaped and including the first convolution block) of the generator is trained. Similarly, at this stage, the discriminator only consists of first two layers (including the first Dis. Convolution block) and the final four layers (starting from the mini-batch standard deviation normalization). After the first

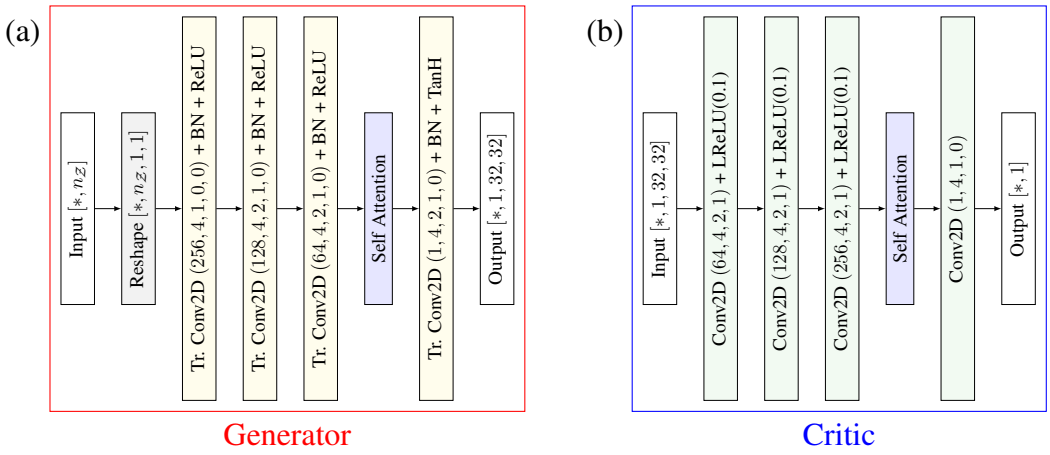


Figure B1. (a) Generator and (b) critic architectures of the WGAN-GP model for the initial condition inference inverse problem.

stage of training is complete, a new convolution block is now appended to the previously trained convolution block in the generator. Similarly, another 'Dis. Convolution Block' is appended to the previously trained block of similar type to the discriminator. All weights of the generator and critic networks are updated at this stage. This process continues for a further four stages, thus a total six stages, up until the GAN learns to synthesize images of the required size. The requisite number of stages $n_{\text{stage}}$ should satisfy $2^{n_{\text{stage}}+1} = 128$. In this study, the desired size of the knee slices was 256×256, which necessitates $n_{\text{stage}} = 7$. We train the GAN for 42 epochs at each stage, which makes a total of 294 epochs across all stages. We use a batch size of 128 for the first four stages, and reduce it by half for the penultimate three stages. We also randomly flip the images horizontally to augment the training dataset. At every stage lower-resolution images from the prior dataset are down-sampled using average pooling to obtain the necessary '*real*' images for training. Moreover, during training at the $k^{\text{th}}$ stage ($k$ starts from 2 going to 8 corresponding to resolutions of 4×4 to 128×128) beyond $k = 2$, the synthesized images are formed by a linear combination of the up-sampled images from the previous generator up to the previous stage and the current stage using residual connections. Similarly, the critic also blends together image features at the $(k-1)^{\text{th}}$ resolution level using residual connections. This linear superposition factor, say $\alpha$, linearly increases between 0 and 1 through the training epochs to ultimately only consider the images entirely synthesized at the $k^{\text{th}}$ resolution level, i.e., the contribution from the residual connections

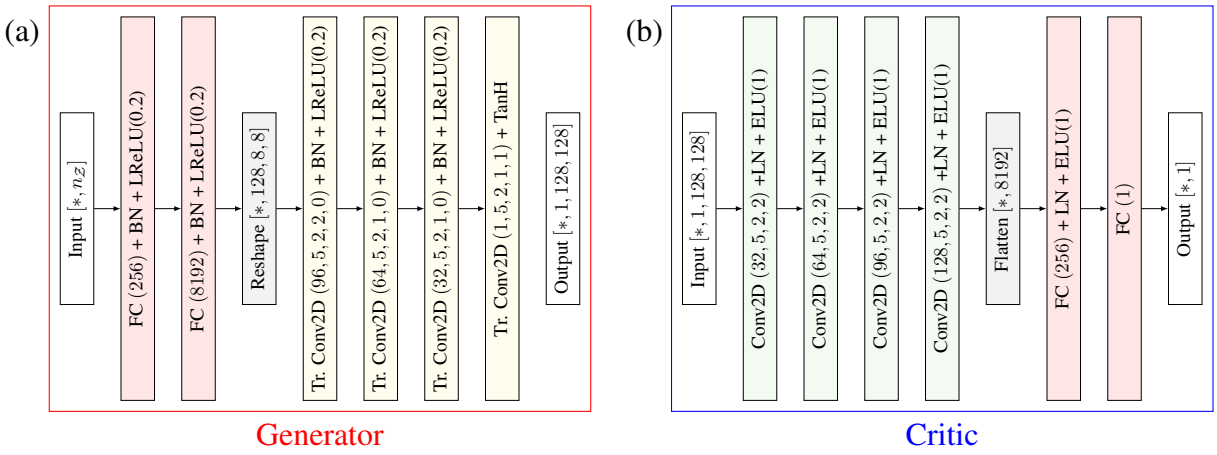


Figure B2. (a) Generator and (b) critic architectures of the WGAN-GP model for the inverse Radon transform problem.

gradually fades away as training progresses for every stage. Figure 2 in [40] is instructive of this multi-scale blending. Fig. B4 shows knee slices of various resolutions generated at the end of $k^{\text{th}}$ stage of training using the WGAN-GP model from Fig. B3.

### *B2. Normalizing flow model architectures*

We use planar flow layers with hyperbolic tangent non-linearity for both the initial condition inference and inverse Radon transform problem. For the phase retrieval problem, we use affine coupling flow layers, as shown in Fig. B5(a), and the scale and shift networks are shown in Fig. B5(b). Every affine coupling block permutes its input such that the partition, described above Eq. (11), is random for every layer; this promotes better mixing among every latent dimension. Subsequently, activation normalization is applied, which scales the inputs to have zero mean and unit variance; this transformation is also updated during training. The re-scaling layer in Fig. B5(b) has a single learnable parameter that simply scales and multiplies itself with the output from the previous layer. This parameter is initially set to zero such that the whole layer starts out as an identity transform. After the re-scaling layer, TanH operation operates on one-half of the data and serves as the scale operator, while the other half acts as the shift operator.

## Appendix C. Additional results

### *C1. Initial condition inference*

For the initial condition inference inverse problem, we vary the latent dimensionality $n_{\mathcal{Z}}$, while keeping all other hyper-parameters fixed, to study its effect on the overall performance of GAN-Flow. Fig. C6 reveals that the performance of both GAN-Flow and GAN+HMC deteriorates as the latent dimensionality $n_{\mathcal{Z}}$ increases. One reason for this may be the following: the latent space dimensionality controls the expressivity of the generator, and a larger than necessary latent space dimensionality may be introducing spurious uncertainty in the prior. Additionally, the deteriorating performance can also be attributed to the curse of dimensionality. As the latent space dimensionality increases, both HMC and normalizing flows find it increasingly harder to sample the latent posterior distribution. In a nutshell, Fig. C6 is empirical evidence of the fact that dimension reduction is beneficial for Bayesian inference. From Fig. C6, $n_{\mathcal{Z}} = 5$ leads to the lowest RMSE in the estimated posterior statistics. This is expected since the underlying prior distribution has only four random variables. While it may appear from Fig. C6(a) that GAN-Flow is not able to estimate the posterior mean as well as GAN+HMC, recall that we train the normalizing flows with the same hyper-parameter setting and a total of $3.2\times10^4$ forward model evaluations irrespective of the value of $n_{\mathcal{Z}}$. Increasing the computational effort as $n_{\mathcal{Z}}$ increases should help improve the performance of GAN-Flow. In Fig. C6(b), GAN-Flow does at least as well as GAN+HMC in capturing the posterior variance at modest dimensions ($n_{\mathcal{Z}} < 80$).

### *C2. Inverse Radon transform*

Similar to the previous example, we vary the latent dimensionality $n_{\mathcal{Z}}$, while keeping all other hyper-parameters fixed, to study its effect on the overall performance of GAN-Flow across different levels of measurement noise. We plot the reconstruction error of the posterior mean in Fig. C7, which shows that the optimal latent dimension $n_{\mathcal{Z}}$ lies between 40 and 80 for the three levels of measurement noise we consider. Note that, in this example, the underlying prior distribution is parameterized by 13 variables. Hence, the reconstructions from GAN-Flow and GAN-HMC are inadequate when $n_{\mathcal{Z}} \leq 20$. At such low latent dimensions, the prior distribution from the WGAN-GP is not sufficiently expressive. Beyond that, the reconstruction error first reduces and then

increases again. The comparatively larger reconstruction errors when $n_{\mathcal{Z}} > 80$ may be due to insufficient training of the normalizing flow or the inefficacy of HMC in sampling high-dimensional distributions. Therefore, for this example, we choose $n_{\mathcal{Z}} = 60$ to obtain a balanced performance from GAN-Flow across all levels of measurement noise.

Further, we test GAN-Flow on another phantom separate from the one in Section 4.2 while keeping all training hyperparameters fixed as before. Fig. C8 shows the phantom, corresponding sinogram and noisy measurements for different levels of noise. Fig. C9 shows the pixel-wise posterior mean and standard deviation estimation estimated using GAN-Flow. With respect to the ground truth Fig. C8(a), the RMSE of the posterior mean reconstruction is 0.044, 0.047 and 0.047 for $\sigma_\eta^2 = 1, 10$ and $50$, respectively. The SSIM is 0.965, 0.963 and 0.962 at $\sigma_\eta^2 = 1, 10$ and $50$, respectively. These metrics are similar to those reported in Table 4. Thus, the performance of GAN-Flow is consistent across both test phantom.

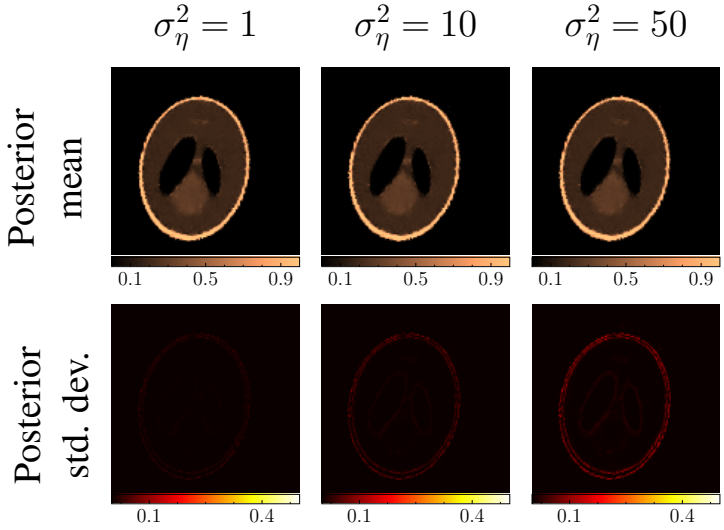


Figure C9. Estimated posterior mean (top row) and standard deviation (bottom row) obtained using GAN-Flow for the alternate test phantom on inverse Radon transform problem at various levels of measurement noise.

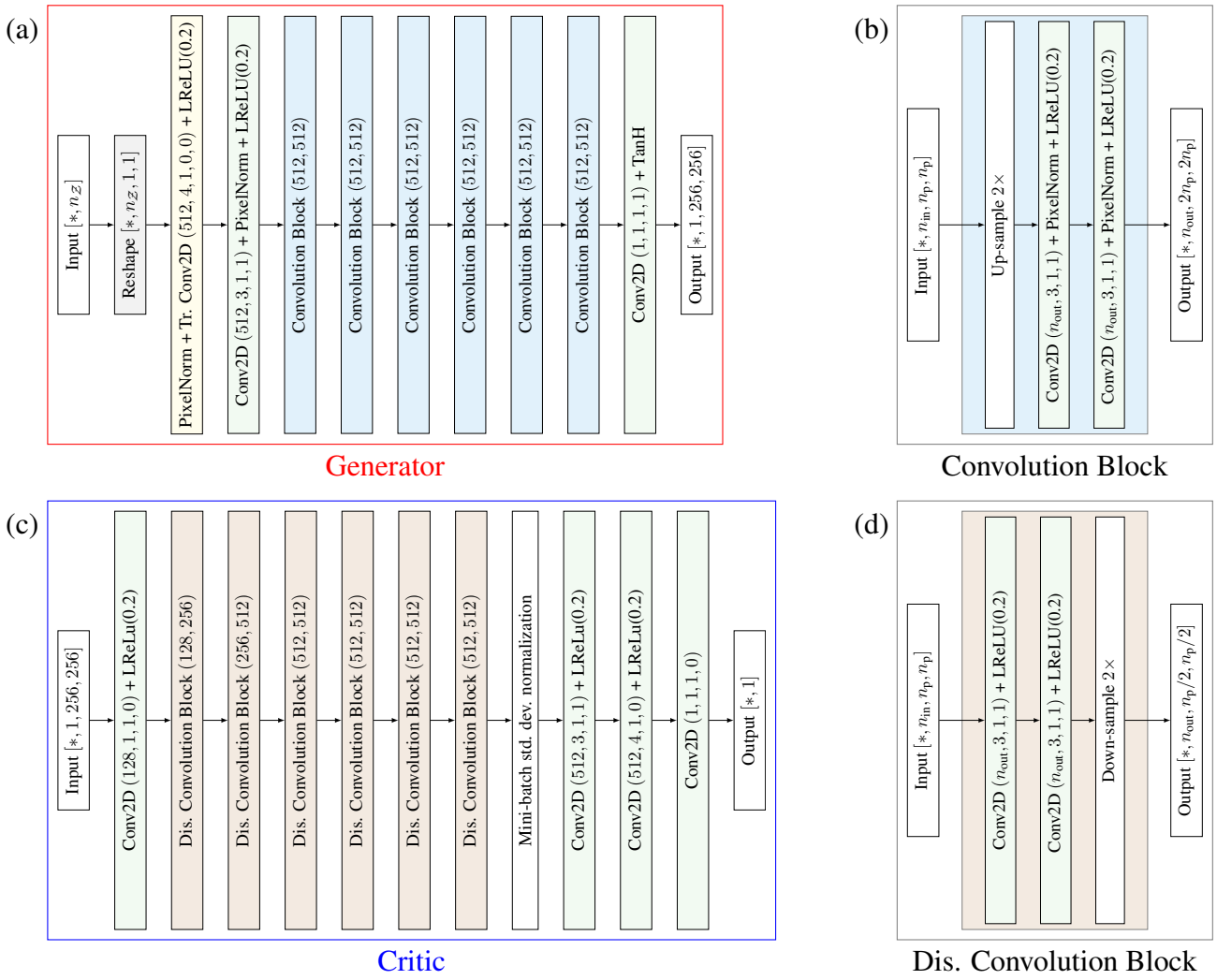


Figure B3. (a) Generator and (c) critic architectures of the WGAN-GP model for the phase retrieval problem. (b) and (d) shows the convolution blocks within the generator and critic, respectively.

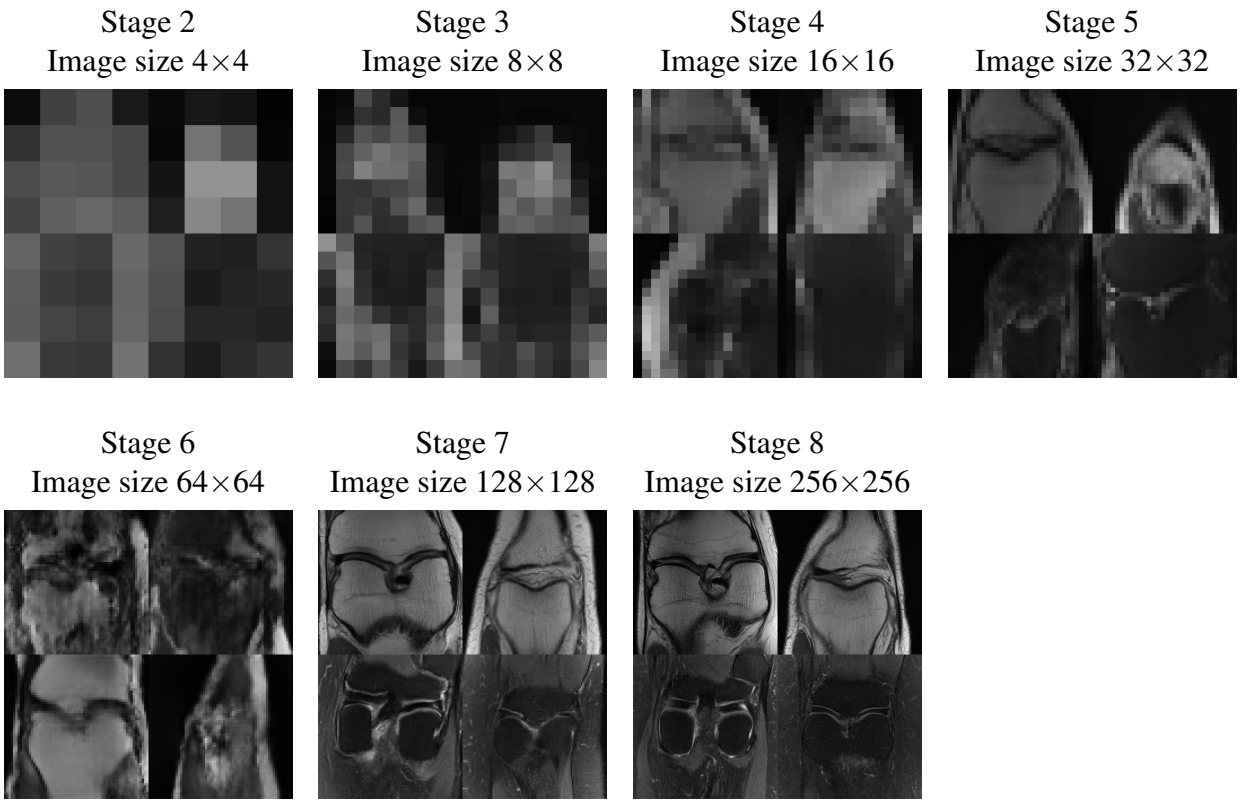


Figure B4. Knee slices of resolution $2^k \times 2^k$ generated at the end of the $k^{\text{th}}$ training stage using the WGAN-GP model for the phase retrieval problem.

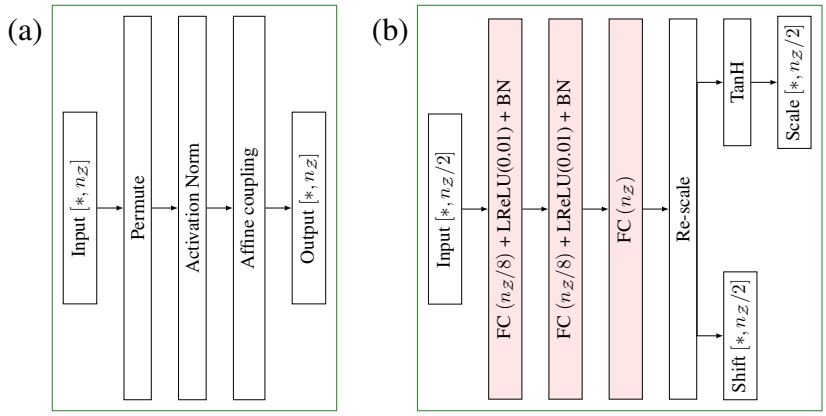


Figure B5. (a) Typical flow layer with affine coupling transform, and (b) scale and shift networks used for the affine coupling transform in the phase retrieval problem.

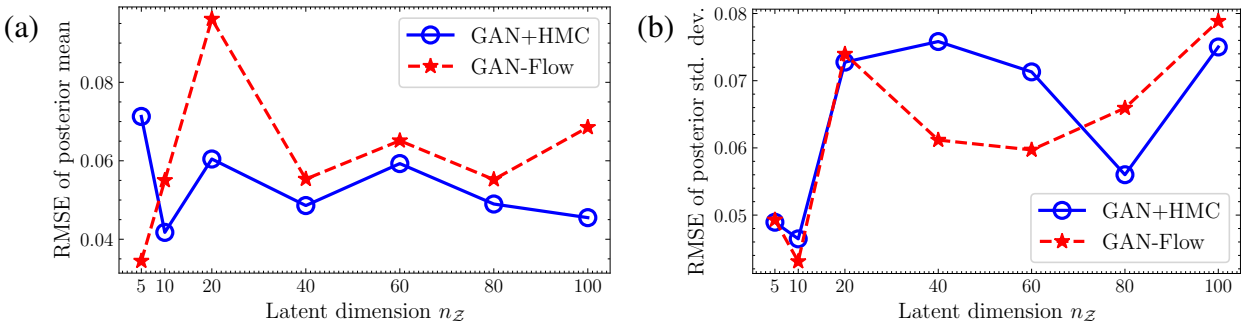


Figure C6. RMSE of the posterior pixel-wise (a) mean and (b) standard-deviation estimated using various methods with respect to the corresponding statistics estimated using MCS for the initial condition inference problem.

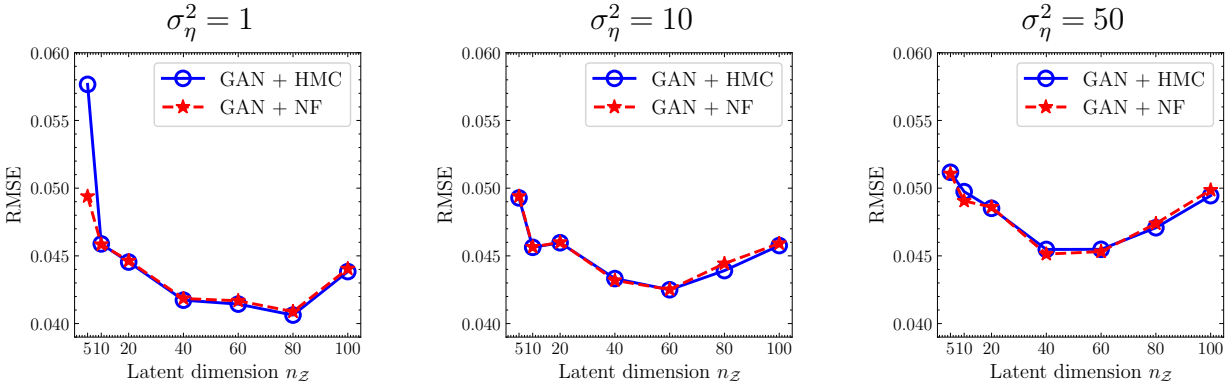


Figure C7. RMSE of the posterior pixel-wise mean with respect to the '*true*' phantom for varying levels of variance $\sigma_\eta^2$ of the measurement noise in the inverse Radon transform problem.

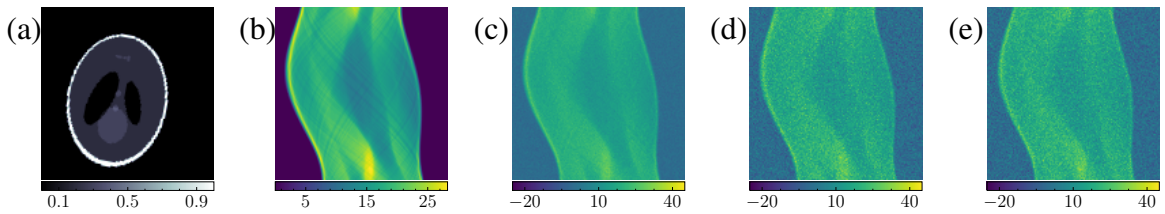


Figure C8. (a) Alternate test phantom and (b) corresponding noise-free sinogram. Noisy sinograms after adding zero-mean Gaussian noise with variance (c) $\sigma_\eta^2 = 1$ (d) $\sigma_\eta^2 = 10$ and (e) $\sigma_\eta^2 = 50$.