GAN-Flow: A dimension-reduced variational framework for physics-based inverse problems

Agnimitra Dasgupta
University of Southern California
Los Angeles, CA 90089, USA
adasgupt@usc.edu

Dhruv V. Patel
Stanford University
Stanford, CA 94305, USA
dvpatel@stanford.edu

Deep Ray
University of Southern California
Los Angeles, CA 90089, USA
deepray@usc.edu

Erik A. Johnson
University of Southern California
Los Angeles, CA 90089, USA
johnsone@usc.edu

Assad A. Oberai
University of Southern California
Los Angeles, CA 90089, USA
aoberai@usc.edu

Abstract

We propose GAN-Flow – a modular inference approach that combines generative adversarial network (GAN) prior with a normalizing flow (NF) model to solve inverse problems in the lower-dimensional latent space of the GAN prior using variational inference. GAN-Flow leverages the intrinsic dimension reduction and superior sample generation capabilities of GANs, and the capability of NFs to efficiently approximate complicated posterior distributions. In this work, we apply GAN-Flow to solve two physics-based linear inverse problems. Results show that GAN-Flow can efficiently approximate the posterior distribution in such high-dimensional problems.

1 Introduction

Bayesian inference is attractive because the posterior distribution is able to capture and quantify uncertainties. However, Bayesian inference can be practically and computationally challenging. Practically, the selection of a well-informed prior density is difficult, but often crucial to the success of Bayesian inference. Computationally, closed-form solutions are seldom available and the posterior distribution must be approximated using appropriate techniques. Notably, high-dimensional posterior distributions are difficult to approximate. Mixing times of Markov chain Monte Carlo (MCMC) algorithms deteriorate as the dimensionality increases [4]. Similarly, designing high-dimensional approximating distribution for variational inference can be equally difficult [7].

We propose a hybrid inference framework for large-scale inverse problems, which we call GAN-Flow, that couples two types of generative models — generative adversarial networks (GANs) and normalizing flows (NFs). GAN-Flow uses a data-driven GAN-based prior and leverages the intrinsic dimensionality reduction offered by it [16, 17]. Next, GAN-Flow approximates the posterior distribution in the latent space of the GAN-based prior using NFs. One notable feature of GAN-Flow is as follows: after the GAN and NF components of the pipeline have been trained, additional samples from the high-dimensional posterior distribution can be generated with minimal computational effort.

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

This offers a significant advantage over sampling from the latent posterior using MCMC methods. Moreover, owing to the NF model in the framework, the overall framework is explicit (permits exact probability density evaluations in the latent space), which can be useful for many downstream tasks such as Bayesian rare-events simulations [20].

Related work Deep generative models have been at the forefront of deep learning-driven inference [13]. GANs have been used to learn prior models [16, 17, 3] or the posterior distribution directly [1]. Similarly, autoencoders [10] and normalizing flows [18, 21, 6] have been used to approximate the posterior distribution. In particular, this work was inspired by [17, 3], where the authors use adversarial priors and subsequently employ Hamiltonian Monte Carlo (HMC) and Metropolis-adjusted Langevin algorithm, respectively, to sample from the latent space's posterior. Bayesian inference approaches that comprise a deep generative prior coupled with a way of sampling from its posterior are known as *modular* Bayesian methods [3]. GAN-Flow is also modular in this sense, yet distinct because it uses a GAN-based prior but approximates the posterior using variational inference with the help of a NF.

2 Proposed method

Preliminaries Consider the random vectors $\mathbf{x} \in \Omega_{\mathcal{X}} \subseteq \mathbb{R}^{N_{\mathcal{X}}}$ and $\mathbf{y} \in \Omega_{\mathcal{Y}} \subseteq \mathbb{R}^{N_{\mathcal{Y}}}$ related by the forward model $\boldsymbol{f} : \Omega_{\mathcal{X}} \to \Omega_{\mathcal{Y}}$ such that $\mathbf{y} = \boldsymbol{f}(\mathbf{x})$. The inference of $\mathbf{x}$ from a noisy measurement vector $\hat{\mathbf{y}}$ constitutes an inverse problem. Given a likelihood function $\mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\mathbf{x})$, Bayes' rule is used to update prior belief about $\mathbf{x}$, characterized using the prior $\mathrm{p}_{\mathcal{X}}(\mathbf{x})$, as follows:

$$\mathrm{p}_{\mathcal{X}}(\mathbf{x}|\hat{\mathbf{y}}) \propto \mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\mathbf{x})\ \mathrm{p}_{\mathcal{X}}(\mathbf{x}). \tag{1}$$

Additionally, if the measurements are corrupted by an additive noise $\boldsymbol{\eta}$ distributed according to p_{η}, then $\hat{\mathbf{y}} = \mathbf{y} + \boldsymbol{\eta}$ and $\mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\mathbf{x}) = \mathrm{p}_{\eta}(\hat{\mathbf{y}} - \boldsymbol{f}(\mathbf{x}))$ in Eq. (1). Using the posterior distribution, the posterior statistics of any desired quantity of interest (denoted as $\boldsymbol{\ell}(\mathbf{x})$) can be computed as follows:

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

Herein, we will refer to $\mathbf{x}$ and $N_{\mathcal{X}}$ as the *ambient* variable and dimension, respectively.

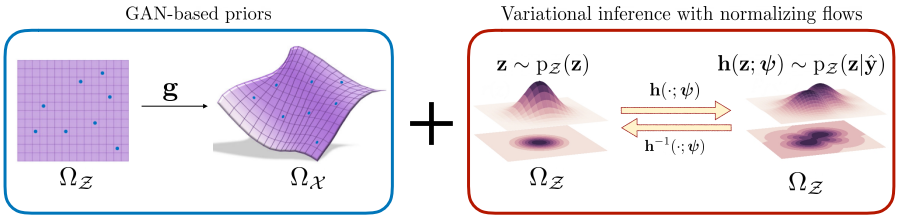

Figure 1. Components of GAN-Flows: GAN-based priors and NFs-based variational inference.

GAN prior GAN-Flow utilizes a Wasserstein GAN with Gradient Penalty (WGAN-GP) [2, 11] to model $\mathrm{p}_{\mathcal{X}}(\mathbf{x})$. It can be shown that WGAN-GP minimizes the Wasserstein-1 distance between $\mathrm{p}_{\mathcal{X}}(\mathbf{x})$ and the distribution of $\boldsymbol{g}(\mathbf{z})$ [16]. As a result, for a perfectly trained GAN with generator $\boldsymbol{g}^*$ [17]

$$\mathbb{E}_{\mathbf{x}\sim\mathrm{p}_{\mathcal{X}}(\mathbf{x})}\left[m(\mathbf{x})\right] = \mathbb{E}_{\mathbf{z}\sim\mathrm{p}_{\mathcal{Z}}(\mathbf{z})}\left[m \circ \boldsymbol{g}^*(\mathbf{z})\right] \quad \forall\ m \in C_b(\Omega_{\mathcal{X}}), \tag{3}$$

where $C_b(\cdot)$ is the space of continuously bounded functions and $\circ$ is used to denote the composition of functions. Eqs. (2) and (3) can be combined to compute

$$\mathbb{E}_{\mathbf{x}\sim\mathrm{p}_{\mathcal{X}}(\mathbf{x}|\hat{\mathbf{y}})}\left[\ell(\mathbf{x})\right] = \mathbb{E}_{\mathbf{z}\sim\mathrm{p}_{\mathcal{Z}}(\mathbf{z}|\hat{\mathbf{y}})}\left[\ell \circ \boldsymbol{g}^*(\mathbf{z})\right] \quad \forall\ \ell \in C_b(\Omega_{\mathcal{X}}), \tag{4}$$

by choosing $m(\cdot) = {}^{\ell(\cdot)\mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\cdot)}/_{\mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}})}$, where $\mathrm{p}_{\mathcal{Z}}(\mathbf{z}|\hat{\mathbf{y}}) \propto \mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\boldsymbol{g}^\star(\mathbf{z}))\mathrm{p}_{\mathcal{Z}}(\mathbf{z})$ is the posterior distribution of $\mathbf{z}$ conditioned on the measurements $\hat{\mathbf{y}}$ and $\mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\boldsymbol{g}^\star(\mathbf{z}))$ is computed as $\mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\mathbf{x})|_{\mathbf{x}=\boldsymbol{g}^*(\mathbf{z})}$ [17]. Eq. (4) implies that any statistics with respect to $\mathbf{x}$ (respectively, $\mathbf{x}|\hat{\mathbf{y}}$) can be computed using realizations of $\mathbf{z}$ (respectively, $\mathbf{z}|\hat{\mathbf{y}}$). Note, the trained generator $\boldsymbol{g}^*$ serves as a map between realizations of $\mathbf{z} \in \Omega_{\mathcal{Z}}$ and realizations of $\mathbf{x} \in \Omega_{\mathcal{X}}$. Moreover, it is conventional to choose $\mathrm{p}_{\mathcal{Z}}(\mathbf{z})$ as an $N_{\mathcal{Z}}$-variate standard normal distribution [17].

Posterior approximation Given a latent prior density $\mathrm{p}_{\mathcal{Z}}(\mathbf{z})$ and an optimally trained generator $\boldsymbol{g}^*$, GAN-Flow uses a normalizing flow [12] to sample from the conditional posterior $\mathrm{p}_{\mathcal{Z}}(\mathbf{z}|\hat{\mathbf{y}})$. This is done by learning a bijective pushforward map $\boldsymbol{h} : \Omega_{\mathcal{Z}} \to \Omega_{\mathcal{Z}}$, parameterized by $\boldsymbol{\psi}$, such that $\boldsymbol{h}(\mathbf{z}) \sim \mathrm{p}_{\mathcal{Z}}(\mathbf{z}|\hat{\mathbf{y}})$, where $\mathbf{z} \sim \mathrm{p}_{\mathcal{Z}}(\mathbf{z})$. The parameters $\boldsymbol{\psi}$ of $\boldsymbol{h}$ is chosen by minimizing the loss

$$\mathcal{L}_{\mathrm{NF}}(\boldsymbol{\psi}) = \mathbb{E}_{\mathbf{z}\sim\mathrm{p}_{\mathcal{Z}}(\mathbf{z})}\big[-\log \mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\boldsymbol{g}^* \circ \boldsymbol{h}(\mathbf{z};\boldsymbol{\psi})) - \log \mathrm{p}_{\mathcal{Z}}(\boldsymbol{h}(\mathbf{z};\boldsymbol{\psi})) - \log|\det \nabla_{\mathbf{z}}\boldsymbol{h}(\mathbf{z};\boldsymbol{\psi})|\big], \quad (5)$$

where $\log \mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\boldsymbol{g}^* \circ \boldsymbol{h}(\mathbf{z}))$ is the log-likelihood and $\nabla_{\mathbf{z}}\boldsymbol{h}(\mathbf{z})$ is the Jacobian of $\boldsymbol{h}$. The above loss function can be derived through the reverse Kulback-Liebler divergence between $\mathrm{p}_{\mathcal{Z}}(\mathbf{z}|\hat{\mathbf{y}})$ and the distribution induced by $\boldsymbol{h}(\mathbf{z})$; see [21] for a similar derivation. For additive noise models, the log-likelihood can also be written as $\mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\boldsymbol{g}^* \circ \boldsymbol{h}(\mathbf{z})) = \mathrm{p}_{\eta}(\hat{\mathbf{y}} - \boldsymbol{f} \circ \boldsymbol{g}^*(\boldsymbol{h}(\mathbf{z})))$. We construct $\boldsymbol{h}$ using NFs based on planar [18] and affine coupling [9] transformations. Flow-based models permit efficient computation of $\nabla_{\mathbf{z}}\boldsymbol{h}(\mathbf{z})$. Interested readers may refer to [12, 14] for reviews on NFs, popular flow-based model architectures used to construct them and their many applications.

3 Numerical examples

We apply GAN-Flow to two physics-based high-dimensional linear inverse problems.

Inferring initial conditions in unsteady heat conduction We apply GAN-Flow to the two-dimensional heat conduction problem where the initial condition of the temperature field (at $t = 0$) must be inferred from a noisy measurement of the temperature field taken after some time (at $t = 1$). For more details about the underlying physics model see Appendix A.1. The initial and final temperature fields ($\mathbf{x}$ and $\mathbf{y}$, respectively) are discretized over a 32×32 Cartesian grid and shown in Fig. 2. Thus, the ambient dimension of this inverse problem is 1024. The measurements (see Fig. 2) are generated by adding zero-mean Gaussian noise with unit variance ($\mathrm{p}_{\eta} \sim \mathcal{N}(0, 1)$) to the final temperature field.

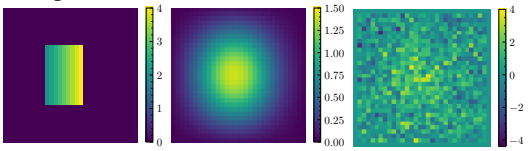

Figure 2. True initial (left), final (middle) and measured (right) temperature fields.

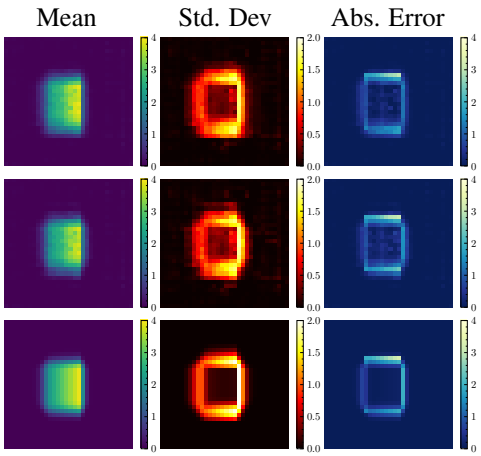

Figure 3. Comparison of Bayesian inference of initial condition of the temperature field using GAN-Flow (first row) and HMC with GAN prior (second row), and the true posterior statistics (third row).

To apply GAN-Flow to this problem, first, we train a WGAN-GP prior, with a 100-dimensional latent space, using a synthetic dataset $\mathcal{S}$ that we prepare by varying a parameterized model that can generate different rectangular initial temperature fields. More details about curating $\mathcal{S}$ can be found in Appendix A.1. For details about the WGAN-GP and the associated hyperparameters see Appendix B.1 and Table B2, respectively. Some training images and realizations from the trained WGAN-GP prior are also shown in Fig. B1. Note, that training the WGAN-GP does not require any evaluations of the forward model $\boldsymbol{f}$. Also, the chosen initial temperature field in Fig. 2 was not used to train the WGAN-GP model. Next, we train a normalizing flow model that uses planar flows with 20 flow layers to approximate the latent posterior. More details about the NF model and associated hyperaparameters are provided in Appendix B.2 and Table B3, respectively. Fig. 3 shows the results of the inference using GAN-Flow. Also shown in Fig. 3 are the results of inference when posterior samples are obtained using HMC (see Appendix C for details) and the true posterior mean and standard deviation that we compute using MC simulation and the parametric prior distribution described in Appendix A.1. The results show that inference using GAN-Flow is consistent with HMC sampling. For all the methods, the reconstruction error and posterior variance are largest around the edges of the initial temperature field where there are sharp transitions. We anticipate that a better WGAN-GP prior will help in resolving the edges of the initial temperature field.

Application to inverse Radon transform Now we apply GAN-Flow for the Bayesian inference of Shepp- Logan phantom [22] from noisy sinogram data. The forward model, described in Appendix A.2, involves the Radon transform [22], which is essentially a linear transformation of the input image $\mathbf{x} \in \mathbb{R}^{N \times N}$. The object represented by the input image is scanned at N uniformly spaced angles with N parallel detectors. Hence, the output sinogram image $\mathbf{y}$, which carries information about the density of the object, is also of size $N \times N$. In this work, we use the `torch-radon` package [19] to compute Radon transforms. For this example, the 'true' object, and the noise-free and noisy sinogram data are shown in Fig. 4. The noisy sinogram data is obtained by adding zero-mean Gaussian noise with variance σ_η^2 to the noise-free sinogram ($\mathrm{p}_\eta \sim \mathcal{N}(0, \sigma_\eta^2)$).

As before, the application of GAN-Flow to the inverse problem at hand starts with the training of a WGAN-GP model that has a 100-dimensional latent space. More details about the architecture of the WGAN-GP used in this example and associated hyperparameters can be found in Appendix B.1 and Table B2, respectively. To train the WGAN-GP model, we generate a dataset $\mathcal{S}$ of 8000 phantom images; each image is of dimension 128×128 (ambient dimensionality). The dataset generation process is adopted from [17] and described in Appendix A.2. In particular, new phantoms are generated by perturbing the Shepp-Logan phantom while respecting underlying spatial topologies. Some realizations drawn from the WGAN-GP prior are also shown in Fig. B1. The true sample shown in Fig. 4 was not a part of the training set $\mathcal{S}$ used to train the GANs.

Next, we perform variational inference in the latent space using a NF model composed of 32 affine coupling layers [9]. More details about the NF model used in this example and associated hyperparameters can be found in Appendix B.2 and Table B3, respectively. Fig. 4 shows the performance of the proposed approach for three different noise levels. GAN-Flow can recover first (mean) and second (variance) order statistics of the posterior distribution at different noise levels. We compare the statistics recovered using GAN-Flow against those computed using realizations of the latent posterior obtained using HMC sampling (see Appendix C for more details). From Fig. 4 we can see a good agreement between the posterior statistics computed using GAN-Flow and the latent posterior sampling. However, we would like to emphasize that GAN-Flow required a total of 8000 evaluations of the forward model $\boldsymbol{f}$ to train the NF model. Neither does the WGAN-GP training require any forward model evaluations, nor does obtaining newer posterior samples. Thus, once the NF model has been trained, it becomes possible to compute any posterior statistics without additional forward model evaluations. In contrast, every sample obtained using HMC will require additional forward model evaluations (more specifically, gradient computations of the output of the forward model with respect to the inputs are necessary). Moreover, in this case, HMC sampling requires more than 8000 samples to be discarded for burn-in [17]. Thus, these results evince the computational efficiency of GAN-Flow that it inherits from being a variational approach. On the flip side, we tried to perform variational inference in the ambient space with a normalizing flow model similar to what we use in GAN-Flow. The ambient space has dimensionality 128^2 in this case, and the memory footprint of the model was so large that we were unable to train it on our GPU with any meaningful batch sizes. This shows that GAN-Flow also benefits from the dimension reduction capabilities of adversarial priors. In Fig. 4 we also show the reconstruction using filtered back projection (FBP), which are expectedly poor.

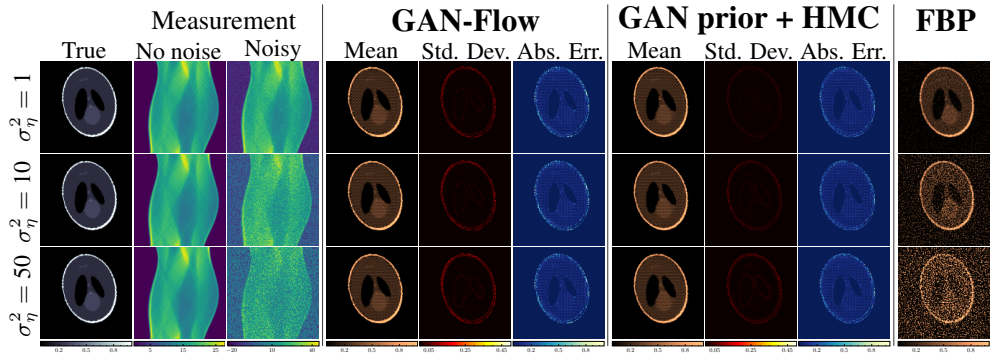

Figure 4. Comparison of Bayesian inference of phantom object from noisy Radon transforms using GAN-Flow and HMC with GAN prior. Also shown is the filtered back projection (FBP) reconstruction (10^{th} column).

4 Conclusions

In this work, we propose GAN-Flow — a modular Bayesian framework for inference and apply it to two physics-based linear inverse problems. GAN-Flow is composed of two generative models: a GAN that serves as the prior and provides a map from the low-dimensional latent space to the ambient space, and a NF model that is used for variational inference in the low-dimensional latent space. The GAN prior facilitates the solution of the inverse problem in a lower-dimensional latent space. NFs are then used to approximate the posterior of the latent variables using variational inference. Results indicate GAN-Flow can be used to efficiently solve high-dimensional inverse problems, and can often be more efficient than sampling from the latent posterior using MCMC-based algorithms. Future work will consider nonlinear inverse models and a methodology for the assessment of the posterior approximation obtained using GAN-Flow.

Broader Impact

In this work, we propose a modular Bayesian framework for inverse problems. Inverse problems are ubiquitous across many disciplines of science and engineering. Therefore, an efficient and robust inference framework that can solve high-dimensional inverse problems will be helpful. We envisage the potential impacts of our work to be beneficial and do not foresee any negative societal impacts of our work.

Acknowledgments and Disclosure of Funding

The authors gratefully acknowledge the support of this work by the National Science Foundation through award CMMI 16-63667 and by the Army Research Office through the grant W911NF2010050. The first author acknowledges the support of the Provost's Ph.D. Fellowship from the University of Southern California. The first author would also like to thank Rajrup Ghosh and Digbalay Bose for allowing the use of their jointly-owned GPU workstation. The second author acknowledges the support of the Timoshenko Distinguished Postdoctoral Fellowship from Stanford University. 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.

References

[1] J. Adler and O. Öktem. Deep Bayesian inversion. *arXiv preprint arXiv:1811.05910*, 2018.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. Proceedings of Machine Learning Research, 2017.

[3] P. Bohra, T.-A. Pham, J. Dong, and M. Unser. Bayesian inversion for nonlinear imaging models using deep generative priors. *arXiv preprint arXiv:2203.10078*, 2022.

[4] Y. Chen, R. Dwivedi, M. J. Wainwright, and B. Yu. Fast mixing of Metropolized Hamiltonian Monte Carlo: Benefits of multi-step gradients. *Journal of Machine Learning Research*, 21:92–1, 2020.

[5] A. D. Cobb and B. Jalaian. Scaling Hamiltonian Monte Carlo inference for Bayesian neural networks with symmetric splitting. *Uncertainty in Artificial Intelligence*, 2021.

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

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

[8] L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[9] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

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

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of Wasserstein GANs. *Advances in Neural Information Processing Systems*, 30, 2017.

[12] I. Kobyzev, S. J. Prince, and M. A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, 2020.

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

[14] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[15] D. Patel and A. A. Oberai. Bayesian inference with generative adversarial network priors. *arXiv preprint arXiv:1907.09987*, 2019.

[16] D. V. Patel and A. A. Oberai. Gan-based priors for quantifying uncertainty in supervised learning. *SIAM/ASA Journal on Uncertainty Quantification*, 9(3):1314–1343, 2021.

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

[18] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. Proceedings of Machine Learning Research, 2015.

[19] M. Ronchetti. Torchradon: Fast differentiable routines for computed tomography. *arXiv preprint arXiv:2009.14788*, 2020.

[20] D. Straub, I. Papaioannou, and W. Betz. Bayesian analysis of rare events. *Journal of Computational Physics*, 314:538–556, 2016.

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

[22] P. Toft. *The Radon transform – Theory and Implementation*. PhD thesis, Technical University of Denmark, 1996.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We have introduced a new framework called GAN-Flow for Bayesian inference of physics-based inverse problems. Through two examples, albeit synthetic and linear, we have demonstrated the efficacy of the framework.
   (b) Did you describe the limitations of your work? [Yes] In the heat conduction example, we noted that the quality of the prior will dictate the overall quality of the inference.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We do not foresee any potential negative impacts of our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We have stated any assumptions made in this work and cited references to most of the theoretical results we have used in this work.
   (b) Did you include complete proofs of all theoretical results? [No] Eq. (5) is the only theoretical result new in this paper and that can be derived following the presentation in [21] after using the appropriate definition of $\mathrm{p}_{\mathcal{Y}}(\hat{\mathbf{y}}|\boldsymbol{g}^{\star}(\mathbf{z}))$.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Codes and datasets will be shared upon request by contacting the corresponding author. But, we provide all pertinent details necessary to reproduce our work in the supplementary material and cite relevant resources/papers where more details can be found.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All training details can be found in Tables A1 and B2 and appendices B.1 and B.2
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Training GANs and NFs is computationally expensive. Hence, we were unable to study the effects of random seeds. If necessary, we will perform further ablation studies and report on the effects of random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We have cited relevant sources for any packages that we have used in this work. The only external asset we use is the `torch-radon` package [19].
   (b) Did you mention the license of the assets? [No] The `torch-radon` package is distributed under the GNU General Public License.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We have cited the relevant source for `torch-radon`
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Forward models and dataset generation

A.1 Initial condition inference

The two-dimensional time-dependent heat conduction partial differential equation over the bounded domain Ω is given as:

$$\begin{aligned} \frac{\partial u(\mathbf{s},t)}{\partial t} - \nabla \cdot (\kappa(\mathbf{s}) \nabla u(\mathbf{s},t)) = b(\mathbf{s}), \qquad \forall (\mathbf{s},t) \in \Omega \times (0,T) \\ u(\mathbf{s},0) = m(\mathbf{s}) \forall \mathbf{s} \in \Omega \\ u(\mathbf{s},t) = 0, \forall (\mathbf{s},t) \in \partial\Omega \times (0,T) \end{aligned} \tag{6}$$

The temperature field is discretized over a 32×32 size grid. Further, we choose $\Omega = [0, L] \times [0, L]$, where $L = 2\pi$, and $t = 1$. All quantities are assumed non-dimensional without a loss in generality. The measurement $\mathbf{y}$ include the temperature field at $t = T$. Added to the measurements is an additive Gaussian noise of zero mean and unit variance. We wish to infer the initial temperature field $\mathbf{x}$ at $t = 0$. It is possible to write $\mathbf{y} = \mathbf{A}\mathbf{x}$ [15], thus, the inverse problem at hand is linear.

Dataset generation We consider the dataset of rectangular inclusion for this experiment. Specifically, we consider a dataset of initial temperature field, where it is zero everywhere else except in a rectangular region, where it varies linearly from the value of 2 units on the left edge to 4 units on the right edge. This rectangular region is generated by sampling the coordinates of the top-left and lower-right corners uniformly between $[0.2L, 0.4L]$ and $[0.6L, 0.8L]$, respectively and L is set to 2π unit.

A.2 Radon Transform

Given the material density function $\rho \in \Omega \subset \mathbb{R}^2 \to \mathbb{R}$, the Radon transform is given as

$$\mathcal{R}(\rho; t, \phi) = \int_{\ell_{t,\phi}} \rho \, \mathrm{d}\ell, \tag{7}$$

where $\ell_{t,\phi}$ is the line that traverses through Ω at a distance of t from the center and an inclination of ϕ. Therefore, given an input phantom image $\mathbf{x} \in \mathbb{R}^{N \times N}$, the forward model

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \in \mathbb{R}^{N \times N} \quad \text{where } \mathbf{y}_{ij} = \mathcal{R}^h(\mathbf{x}; t_i, \phi_j). \tag{8}$$

$\mathcal{R}^h$ is the discrete Radon transform. The output $\mathbf{y}$ is commonly known as a sinogram. In this work, we choose $N = 128$, and each phantom is scanned at 128 uniformly spaced angles between $0°$ and $180°$ with 128 detectors.

Dataset generation We generate the dataset used to train the WGAN following Patel et al. [17]. The Shepp-Logan phantom is composed of the union of ten ellipses. Each ellipse has constant but different density. Let the k^{th} ellipse E_k be centered at (r_k, s_k), with semi-axis lengths a_k, b_k, angle of inclination α_k and density ρ_k. Now, the density of the phantom at any coordinate (r, s) is given by

$$\rho(r,s) = \Sigma_{k=1}^{10} C_k(r,s), \quad \text{where } C_k(r,s) = \begin{cases} \rho_k & \text{if } (r,s) \in E_k \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

We adopt the values of the base parameters from [22]; the values of the base parameters are tabulated in Table A1.

We generate new phantoms by perturbing the original base parameters in Table A1 to obtain slightly perturbed ellipses $\tilde{E}_k$ as follows:

$$\begin{aligned} \tilde{r}_k = r_k + 0.005\xi_{k,1}, \;\; \tilde{s}_k = s_k + 0.005\xi_{k,2}, \;\; \tilde{a}_k = a_k + 0.005\xi_{k,3}, \\ \tilde{b}_k = b_k + 0.005\xi_{k,4}, \;\; \tilde{\alpha}_k = \alpha_k + 2.5\xi_{k,5}, \;\; \tilde{\rho}_k = \rho_k + 0.0005\xi_{k,6} \end{aligned} \tag{10}$$

where $\left\{ \{\xi_{k,i}\}_{i=1}^{i=6} \right\}_{k=1}^{k=10}$ are random parameters with distribution $\mathcal{U}(-1, 1)$. Finally, we set the density ρ of the perturbed phantom following

$$\rho(r,s) = \max\left(0, \min\left(1, \Sigma_{k=1}^{10} \tilde{C}_k(r,s)\right)\right), \quad \text{where } \tilde{C}_k(r,s) = \begin{cases} \tilde{\rho}_k & \text{if } (r,s) \in \tilde{E}_k \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Table A1. Base parameters of the Shepp-Logan phantom. Note, α_k is in degrees.

k	r_k	s_k	a_k	b_k	α_k	ρ_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

which also ensures that the material density $\tilde{\rho}$ at any point is bounded within 0 (air cavity) and 1 (bone). We obtain discrete phantom images by evaluating Eq. (11) on a 128×128 grid. This image is further subject to a transformation that translates it by n and m pixels in the horizontal and vertical direction, respectively, and rotates it by an angle β. We assume

$$n, m \sim \mathcal{U}\{-8, -7, \ldots, 7, 8\}, \quad \beta \in \mathcal{U}(-20°, 20°) \tag{12}$$

B GAN-Flow architectures and hyperparameters

GAN-Flows consist of a WGAN-GP and a normalizing flow model. We provide details of the WGAN architectures in Appendix B.1 and associated hyperparameters in Table B2 . Similarly, we provide details of the affine coupling transform-based normalizing flow model in Appendix B.2. To describe the network architectures we introduce the following nomenclature:

- `FC(`n`,`m`)` denotes a fully connected layer with n input neurons and m output neurons.
- `Conv2D(`c`,`$k_1 \times k_2$`, `s`, `p`)` denotes a convolution layer with c filters of size $k_1 \times k_2$ with stride s and padding p
- `TransConv2D(`c`,`$k_1 \times k_2$`, `s`, `p`)` denotes a transpose convolution layer with c filters of size $k_1 \times k_2$ with stride s and padding p
- `BN`, `LN` and `ActNorm` denote batch norm, layer norm and activation normalization, respectively.
- `TanH`, `Sigmoid`, `LReLU`(α) denote hyperbolic tangent, sigmoid and leaky ReLU (with activation parameter α) activation functions, respectively.

Some realizations from the trained WGAN-GP priors of the initial temperature field and phantom object are shown in Fig. B1. All experiments in this work were carried out on an NVIDIA Quatro RTX 800 GPU that has a memory capacity of 48 GB.

Table B2. Hyperparameters for WGAN-GP.

Hyperparameter	Rectangular initial condition	Shepp-Logan phantom
Epochs	500	1000
Learning Rate	0.0001	0.0002
Bach Size	64	100
$n_{\text{critic}}/n_{\text{gen}}$	5	4
Optimizer	Adam	RMSprop
Optimizer parameters	$\beta_1 = 0.9$, $\beta_2 = 0.99$	0.9
Gradient penalty	10	10

B.1 GAN Architectures

Initial condition inference In this example we choose $N_{\mathcal{Z}} = 100$ and the generated images are of size 32×32. The generator architecture is as follows: `Input` (shape $[*, 100]$) $\rightarrow$ `FC(100, 512)`

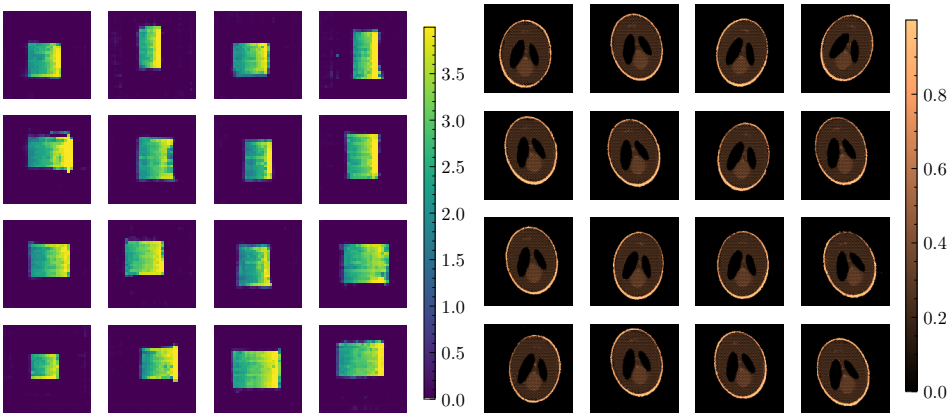

Figure B1. Realizations drawn from the trained WGAN-GP priors of the initial temperature field (left) and phantom object (right).

$\rightarrow$ `ReLU` $\rightarrow$ `Reshape` into $[*, 32, 4, 4]$ $\rightarrow$ `TransConv2D(16, 4 × 4, 2, 1)` $\rightarrow$ `ReLU` $\rightarrow$ `BN` $\rightarrow$ `TransConv2D(8, 4 × 4, 2, 1)` $\rightarrow$ `ReLU` $\rightarrow$ `BN` $\rightarrow$ `TransConv2D(1, 4 × 4, 2, 1)` $\rightarrow$ `TanH` $\rightarrow$ Rescale between $[0, 4]$ $\rightarrow$ `Output`.

The critic architecture is as follows: `Input` (shape $[*, 1, 32, 32]$) $\rightarrow$ `Conv2D(8, 4 × 4, 2, 2)` $\rightarrow$ `LReLU(0.2)` $\rightarrow$ `Conv2D(16, 4 × 4, 2, 2)` $\rightarrow$ `LReLU(0.2)` $\rightarrow$ `Conv2D(32, 4 × 4, 2, 2)` $\rightarrow$ `LReLU(0.2)` $\rightarrow$ `Reshape` into $[*, 512]$ $\rightarrow$ `FC(512, 8)` $\rightarrow$ `LReLU(0.2)` $\rightarrow$ `FC(8, 1)`

Inverse radon transform In this example we choose $N_{\mathcal{Z}} = 100$ and the generated phantoms are of size 128×128. The generator architecture is as follows: `Input` (shape $[*, 100]$) $\rightarrow$ `FC(100, 1024)` $\rightarrow$ `BN` $\rightarrow$ `ReLU` $\rightarrow$ `FC(1024, 8192)` $\rightarrow$ `BN` $\rightarrow$ `ReLU` $\rightarrow$ `Reshape` into $[*, 128, 8, 8]$ $\rightarrow$ `TransConv2D(128, 5 × 5, 2, 2)` $\rightarrow$ `BN` $\rightarrow$ `ReLU` $\rightarrow$ `TransConv2D(64, 5 × 5, 2, 1)` $\rightarrow$ `BN` $\rightarrow$ `ReLU` $\rightarrow$ `TransConv2D(32, 5 × 5, 2, 1)` $\rightarrow$ `BN` $\rightarrow$ `ReLU` $\rightarrow$ `TransConv2D(1, 5 × 5, 2, 1)` $\rightarrow$ `TanH` $\rightarrow$ Rescale between $[0, 1]$ $\rightarrow$ `Output`.

The critic architecture is as follows: `Input` (shape $[*, 1, 128, 128]$) $\rightarrow$ `Conv2D(1, 5 × 5, 2, 2)` $\rightarrow$ `LReLU(0.2)` $\rightarrow$ `Conv2D(32, 5 × 5, 2, 2)` $\rightarrow$ `LN` $\rightarrow$ `LReLU(0.2)` $\rightarrow$ `Conv2D(64, 5×5, 2, 2)` $\rightarrow$ `LN` $\rightarrow$ `LReLU(0.2)` $\rightarrow$ `Conv2D(128, 5×5, 2, 2)` $\rightarrow$ `LN` $\rightarrow$ `LReLU(0.2)` $\rightarrow$ `Reshape` into $[*, 8192]$ $\rightarrow$ `FC(8192, 1024)` $\rightarrow$ `FC(1024, 1)` $\rightarrow$ `Output`.

B.2 Normalizing flow architectures

In this section we provide details of the normalizing flow models that we use in the two numerical examples. Details of associated hyperparameters are provided in Table B3.

Initial condition inference For this example, we use planar flow layers [8, 9] to construct normalizing flows. The coupling layer subjects to its input $\mathbf{u}$ the following perturbation to obtain the output $\mathbf{v}$:

$$\mathbf{v} = \mathbf{u} + \mathbf{s} \odot h(\boldsymbol{w}^{\mathrm{T}}\mathbf{u} + b), \tag{13}$$

where $\boldsymbol{w}, \mathbf{s}, b$ are the parameters of the flow layer and $h(\cdot)$ is an element-wise nonlinearity that we choose to be the `TanH` activation.

Inverse radon transform For this example, we use affine coupling transform-based flow layers [8, 9] to construct normalizing flows. The coupling layer (see Fig. B2) works as follows: an input $\mathbf{u}$ is first subjected to a random permutation followed by `ActNorm`, and then divided into two parts $\mathbf{u}_1$ and $\mathbf{u}_2$ which are transformed into $\mathbf{v}_1$ and $\mathbf{v}_2$ as follows

$$\mathbf{v}_1 = \mathbf{u}_2 \odot \exp[s_1(\mathbf{u}_1)] + t_1(\mathbf{u}_1) \;\rightarrow\; \mathbf{v}_2 = \mathbf{u}_1 \odot \exp[s_2(\mathbf{v}_1)] + t_2(\mathbf{v}_1), \tag{14}$$

such that $\mathbf{v}^{\mathrm{T}} = [\mathbf{v}_1^{\mathrm{T}}, \mathbf{v}_2^{\mathrm{T}}]$. where s_1, s_2, t_1, t_2 are scale and shift operators modeled using standard neural networks. A normalizing flow is constructed by stacking K such coupling layers.

The use of coupling layer constrains the Jacobian to be upper triangular; meaning $\det \partial \mathbf{v}/\partial \mathbf{u}$ can be computed efficiently [8]. s_i, t_i is the output of a fully connected deep neural network as follows: `Input` (shape $[*, 50]$) $\rightarrow$ `FC(50, 12)` $\rightarrow$ `LReLU(0.01)` $\rightarrow$ `BN` $\rightarrow$ `FC(50, 12)` $\rightarrow$ `LReLU(0.01)` $\rightarrow$ `BN` $\rightarrow$ `FC(12, 100)` $\rightarrow$ `Output`. One half of this output is s_i and the other half is t_i.

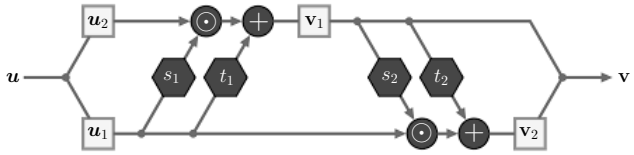

Figure B2. Affine coupling layer [9].

Table B3. Normalizing flow models and associated hyperparameters.

Model / Hyperparameter	Rectangular initial condition	Shepp-Logan phantom
Flow Layer	Planar [18]	Affine coupling [9]
No. of layers	20	32
Epochs	500	500
Learning Rate	0.005	0.005
Bach Size	32	16
Optimizer	Adam	Adam
Optimizer parameters β_1, β_2	0.9, 0.999	0.9, 0.999

C Inference using Hamiltonian Monte Carlo

In this section we provide details of posterior sampling from the latent space using Hamiltonian Monte Carlo (HMC). We use the `hamiltorch` package [5] to perform No U-turn sampling (NUTS). For both examples, we obtain a sample of size 60,000 and discard the first 30,000 realizations considering burn-in. We also set the number of leap frog steps to 10 and the step size as 0.01 with a desired acceptance rate of 0.75.