Computers & Geosciences 171 (2023) 105263

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Review article

Variational Bayesian inference with complex geostatistical priors using inverse autoregressive flows

Shiran Levy [a,*], Eric Laloy [b], Niklas Linde [a]

[a] Institute of Earth Sciences, University of Lausanne, Lausanne, Switzerland
[b] Institute for Environment, Health and Safety, Belgian Nuclear Research Centre, Mol, Belgium

ARTICLE INFO

Keywords:
Geophysical inversion
Normalizing flows
Variational inference
Generative adversarial network
Variational autoencoder

ABSTRACT

We combine inverse autoregressive flows (IAF) and variational Bayesian inference (variational Bayes) in the context of geophysical inversion parameterized with deep generative models encoding complex priors. Variational Bayes approximates the unnormalized posterior distribution parametrically within a given family of distributions by solving an optimization problem. Although prone to bias if the chosen family of distributions is too limited, it provides a computationally-efficient approach that scales well to high-dimensional inverse problems. To enhance the expressiveness of the variational distribution, we explore its combination with IAFs that allow samples from a simple base distribution to be pushed forward through a series of invertible transformations onto an approximate posterior. The IAF is learned by maximizing the lower bound of the evidence (marginal likelihood), which is equivalent to minimizing the Kullback–Leibler divergence between the approximation and the target posterior distribution. In our examples, we use either a deep generative adversarial network (GAN) or a variational autoencoder (VAE) to parameterize complex geostatistical priors. Although previous attempts to perform Gauss–Newton inversion in combination with GANs of the same architecture were proven unsuccessful, the trained IAF provides a good reconstruction of channelized subsurface models for both GAN- and VAE-based inversions using synthetic crosshole ground-penetrating-radar data. For the considered examples, the computational cost of our approach is seven times lower than for Markov chain Monte Carlo (MCMC) inversion. Furthermore, the VAE-based approximations in the latent space are in good agreement. The VAE-based inversion requires only one sample to estimate gradients with respect to the IAF parameters at each iteration, while the GAN-based inversions need more samples and the corresponding posterior approximation is less accurate.

1. Introduction

Probabilistic inverse modeling is often based on Bayes' theorem:

$$p(\mathbf{m}|\mathbf{d}) = \frac{p(\mathbf{d}|\mathbf{m})p(\mathbf{m})}{p(\mathbf{d})}, \quad (1)$$

where $\mathbf{m}$ are unobserved model parameters, $\mathbf{d}$ are the measured data, $p(\mathbf{m}|\mathbf{d})$ is the posterior probability density function (PDF) of interest, $p(\mathbf{m})$ is the prior PDF, $p(\mathbf{d}|\mathbf{m})$ is the likelihood and $p(\mathbf{d}) = \int p(\mathbf{d}|\mathbf{m})p(\mathbf{m})d\mathbf{m}$ is the marginal likelihood that is often referred to as the evidence. The latter is very challenging to estimate, especially for problems of large dimensionality, due to the requirement of integrating the likelihood over the prior of all possible model parameters $\mathbf{m}$. Markov chain Monte Carlo (MCMC) methods circumvent this problem of evidence estimation by making model proposals using formalized rules and comparing posterior probability ratios, thereby, enabling unbiased sampling from $p(\mathbf{m}|\mathbf{d})$ provided that the MCMC chain(s) are long enough (Robert et al., 1999). In practice, MCMC methods can incur prohibitive computational costs for many problems encountered in the geosciences.

Variational inference (VI; Blei et al., 2017) or its Bayesian variant termed variational Bayes (VB; Kingma and Welling, 2014) provides an attractive alternative to MCMC methods as it replaces a sampling problem with an optimization problem. It proceeds by approximating the posterior PDF of interest using a surrogate distribution referred to as the variational density, which is adjusted such that the evidence lower bound (ELBO, see Section 2.2) is maximized. The variational density belongs to a family of distributions from which it inherits its parameterization. The approximation resulting from VI is limited by the chosen parametric family of distributions. For instance, a classical choice is to use a Gaussian distribution with unknown hyperparameters, which often offers a poor approximation.

* Corresponding author.
E-mail address: shiran.levy@unil.ch (S. Levy).

https://doi.org/10.1016/j.cageo.2022.105263
Received 21 June 2022; Received in revised form 28 September 2022; Accepted 29 October 2022
Available online 14 November 2022

Various variational techniques involving intermediate invertible transformations have been developed to allow for more expressive variational densities. Automatic differential variational inference (ADVI; Kucukelbir et al., 2017), for instance, attempts to accommodate different probabilistic models by transforming the original latent space of the model into an unconstrained real-valued space, serving as a "common space". VI is then performed on the common space and differentiation is performed with respect to the original latent space. This approach offers an automatic, comfortable and efficient way to perform VI for a variety of models. Nevertheless, there are several limitations to ADVI: (1) the approximation might suffer from bias due to implicit Gaussian approximations, (2) the approximation is sensitive to the choice of the invertible transformation connecting the variational density in the real-valued space and the original space and (3) it might not be suitable when the posterior is multi-modal (Kucukelbir et al., 2017; Zhang and Curtis, 2020a; Zhao et al., 2021). Another approach that has seen multiple applications in geophysics (Zhang and Curtis, 2020b; Ramgraber et al., 2021; Zhang and Curtis, 2021) is Stein variational gradient descent (SVGD; Liu and Wang, 2016). It uses an ensemble of particles, initialized from a base analytical distribution, that are iteratively updated to approximate the posterior using a smooth transformation describing an incremental perturbation. In each step, particles are updated via perturbations in the direction of the steepest descent, where the magnitude and direction of perturbations are determined based on the Stein operator (more specifically Stein's identity and kernelized Stein discrepancy), to minimize the Kullback–Leibler divergence (D_{KL}, Kullback and Leibler, 1951) between the current distribution of the particles and a target distribution. An advantage of SVGD is that it does not require explicit parameterization. However, SVGD underestimates the variance as the dimensionality of the problem increases and, therefore, performs poorly on high-dimensional problems (Ba et al., 2019). Ba et al. (2019) argue that accurate estimates using SVGD could be obtained by either increasing the number of particles, but this comes at a high computational cost and might not always be practical for high-dimensional problems, or by introducing re-sampling to avoid deterministic bias.

In this study, we consider the increasingly popular family of transformations referred to as normalizing flows (Rezende and Mohamed, 2015; Papamakarios et al., 2021; Kobyzev et al., 2021). Normalizing flows transform an initial density of random variables into a target density of richer form through a series of invertible, differentiable and volume conserving maps. Their combination with VI enables a more flexible and scalable approach allowing for approximate posterior distributions of high complexity (Rezende and Mohamed, 2015; Kingma et al., 2016). Some example applications are flow-based generative models (Dinh et al., 2016, 2014; Kingma and Dhariwal, 2018), inference, reparameterization and representation learning (Papamakarios et al., 2021 and references therein). In a geophysical context, Zhao et al. (2021) assessed normalizing flows expressed by neural networks on two tomographic problems and found that it can significantly reduce the number of forward evaluations needed to reach a solution compared to SVGD and MCMC, while at the same time being less biased than ADVI. However, the authors indicate a possible drawback when training the neural network for high-dimensional problems, for example, in 3D problems. This motivates our work which seeks to combine such approaches with dimensionality reduction.

One of the most popular techniques to reduce dimensionality is principal component analysis (PCA; Wold et al., 1987), although a plethora of other methods exist (e.g. Kernel-PCA, linear discriminant analysis and deep neural networks; Dejtrakulwong et al., 2012; Konaté et al., 2015; Hinton and Salakhutdinov, 2006). For example, Urozayev et al. (2021) used VB to infer the low-dimensional latent variables describing the coefficients of a discrete cosine transform (DCT) in a seismic imaging problem. By reducing the dimensionality and using VB, they could reduce the computational complexity and ensure that geologically-meaningful solutions were obtained. Laloy et al. (2017, 2018) showed that deep generative neural networks, such as variational autoencoders (VAEs) or generative adversarial networks (GANs), are well-suited for dimensionality reduction when working on inverse problems with complex prior models. Such methods allow for fast sampling from the prior and the reduction in dimensionality makes MCMC inversions more efficient compared to alternative approaches relying on a training image (TI) such as sequential geostatistical resampling (Mariethoz et al., 2010; Hansen et al., 2012; Tahmasebi, 2018).

Generally speaking, there are two ways in which generative neural networks can be used in inverse modeling. In the first approach, a pre-trained generative network is combined with an inference framework (e.g. MCMC). In the second approach, the generative network serves as the inference network that is trained to generate realizations that honor the data (Dupont et al., 2018; Mosser et al., 2018; Song et al., 2021b,a; Laloy et al., 2021). The first approach can be further split into two sub-approaches: (1) The distribution conditional on the data is explored in the latent space of the generative network by sampling, minimization or optimization methods (Laloy et al., 2017, 2019; Mosser et al., 2020; Levy et al., 2022) and (2) a mapping is learned between an initial simple distribution and a distribution on the latent space of the generative network which is conditioned on data and from which we can sample conditional realizations (Chan and Elsheikh, 2019). Here we study this latter sub-approach for inversion and build on previous works on normalizing flows and VI (Rezende and Mohamed, 2015; Kingma et al., 2016; Hoffman et al., 2019). We train inverse autoregressive flows (IAF; Kingma et al., 2016), a type of normalizing flows, using stochastic variational inference (SVI; Hoffman et al., 2013) to invert synthetic, noise contaminated (indirect) geophysical data in presence of a complex geostatistical prior. We refer to this approach as neural-transport (Hoffman et al., 2019). Our model parameters are parameterized within the latent space of a deep generative model (DGM): either a GAN or a VAE. Training of the IAF proceeds by randomly drawing samples from a standard normal distribution and pushing them through the IAF transform into a space in which VI is performed. The parameters of the IAF are updated at each training iteration using stochastic gradient-based optimization with the objective to maximize the ELBO.

For the same type of subsurface models as considered herein, Laloy et al. (2019) attempted to infer the latent parameters of a GAN using two different deterministic gradient-based inversion approaches. They found that even when a linear forward solver was used, both approaches performed poorly given the high non-linearity of the GAN. Their conclusion was later reinforced by Lopez-Alvis et al. (2021) who suggested to replace the GAN with a VAE, for which they obtained better inversion results. This is because the VAE generator was found to be less nonlinear and to better preserve topology compared to the GAN generator (see Lopez-Alvis et al., 2021, for details). As neural-transport is a stochastic approach that relies on gradient-based optimization, we expect it to perform better than deterministic gradient-based approaches and, thereby, at least partly avoid pitfalls due to the non-linearity and complex manifold topology of the GAN. Additionally, neural-transport may offer a potentially-significant speedup compared to MCMC given that (1) it allows parallelization of the problem making it well suited to high-dimensional problems and (2) it solves an optimization problem using gradient-based information. The objective of this study is to assess the performance of the neural-transport approach with respect to using either a GAN or a VAE and compare its performance against MCMC results.

The remainder of the paper is structured as follows. Section 2 briefly describes the theory behind each component of the methodology namely, the used DGMs, IAF, VI and the combined neural-transport routine. Section 3 presents inversion results obtained from neural-transport and a comparison of neural-transport against MCMC. Section 4 discusses the results, advantages and limitations of neural-transport and outlines possible future developments. Section 5 concludes the study.

2. Methods

We build upon the work of Hoffman et al. (2019) who coined the term neural-transport (NT) to describe the trained IAF transformation. Compared to previous work with NT, here we consider an intermediate latent space of a DGM: either a SGAN or a VAE. The IAF (Section 2.1) serves as an inference network in which samples from a standard normal distribution are mapped into a target distribution. The IAF is trained through variational Bayesian inference (Section 2.2), in which the parameters of the transformation are iteratively updated through gradient-based optimization. The inferred model parameters are those within the latent space of the DGM (Section 2.3) while the physical forward response (Section 2.4) is computed on high-dimensional model realizations following the DGM transformation. Finally, the resulting approximate posterior distribution, that is conditioned on indirect (noise-contaminated) geophysical data, can be sampled and estimated. We describe the implementation of this approach, combining NT and DGMs, in Section 2.5. To assess the quality of the IAF approximation we use several metrics (Section 2.6) such as the root-mean-squared error and structural similarity index and compare with results obtained by MCMC inversion.

2.1. Inverse autoregressive flows

IAF is a class of normalizing flows, in which a random variable $\mathbf{z}^{(0)}$ drawn from a known probability density function (base distribution) $\mathbf{z}^{(0)} \sim q(\mathbf{z}^{(0)})$ is mapped into a random variable $\mathbf{m}$ from the target distribution $\mathbf{m} \sim q(\mathbf{m})$. Given a transformation $\mathbf{m} = f(\mathbf{z}^{(0)})$ where $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is an invertible, continuous and differentiable mapping between two random variables, one can sample from the target distribution by applying the transformation and evaluating the target distribution using the change of variables theorem

$$q(\mathbf{m}) = q(\mathbf{z}^{(0)}) \left| \det \frac{df(\mathbf{z}^{(0)})}{d\mathbf{z}^{(0)}} \right|^{-1}, \tag{2}$$

where $\frac{df(\mathbf{z}^{(0)})}{d\mathbf{z}^{(0)}}$ is the Jacobian matrix $\mathbf{J}$ and $\det \frac{df(\mathbf{z}^{(0)})}{d\mathbf{z}^{(0)}}$ its determinant, representing the change in volume as a result of the transformation f. If the mapping consists of several transformations, the logarithmic form of $q(\mathbf{m})$ can be evaluated by:

$$\log q(\mathbf{m}) = \log q(\mathbf{z}^{(0)}) - \sum_{k=1}^{K} \log \left| \det \frac{df_k(\mathbf{z}^{(k-1)})}{d\mathbf{z}^{(k-1)}} \right|, \tag{3}$$

where $k = 1, \dots, K$ is the number of sequential transformations and $\mathbf{m} = \mathbf{z}^{(K)}$. In IAF, the transformation f_k applied on the random variable $\mathbf{z}_i^{(k-1)}$ $(i = 1, 2, \dots, n)$ is conditional on previous instances and can be formulated as:

$$z_i^{(k)} \sim q(z_i^{(k)} | \mathbf{z}_{1:i-1}^{(k-1)}) = f_k(z_i^{(k-1)}) = z_i^{(k-1)} \odot \sigma_{\phi,i}(\mathbf{z}_{1:i-1}^{(k-1)}) + \mu_{\phi,i}(\mathbf{z}_{1:i-1}^{(k-1)}), \tag{4}$$

where ϕ are the trainable parameters of the IAF and σ and μ are the scale and shift functions, respectively, conditional on previous instances. For this type of transformation $|\det \mathbf{J}_k| = \prod_i^n \sigma_i$, making the determinant of the Jacobian easy to compute and the target distribution easier to evaluate. Since $z_i^{(k)}$ only depends on known variables $\mathbf{z}_{1:i}^{(k-1)}$, the mapping can be computed in parallel.

2.2. Variational Bayesian inference

Variational Bayes is an approach to approximate an intractable posterior distribution by optimization. The approximation of $p(\mathbf{m}|\mathbf{d})$ is made with a surrogate distribution $q^*(\mathbf{m})$ defined within a family $\mathcal{Q}$, for which:

$$q^*(\mathbf{m}) = \arg \min_{q(\mathbf{m}) \in \mathcal{Q}} D_{KL}(q(\mathbf{m}) \parallel p(\mathbf{m}|\mathbf{d})). \tag{5}$$

The notation D_{KL} in Eq. (5) indicates the Kullback–Leibler divergence (KL; Kullback and Leibler, 1951), a statistical measure of the distance between two distributions defined as $D_{KL}(f(x) \parallel g(x)) = \int f(x) \log(\frac{f(x)}{g(x)}) dx$ for a given random variable x. We define ϕ as the parameterization of the variational density $q(\mathbf{m})$ and it depends on our choice of the distribution family $\mathcal{Q}$. Since the posterior distribution $p(\mathbf{m}|\mathbf{d})$ is intractable in most cases and the evidence $p(\mathbf{d})$ is a constant, a common approach is to instead maximize the evidence lower bound (ELBO; see Blei et al., 2017)

$$\log p(\mathbf{d}) = \underbrace{\mathbb{E}_{\mathbf{m}\sim q} \log p(\mathbf{m}, \mathbf{d}) - \mathbb{E}_{\mathbf{m}\sim q} \log q_\phi(\mathbf{m})}_{ELBO} + D_{KL}(q_\phi(\mathbf{m}) \parallel p(\mathbf{m}|\mathbf{d})). \tag{6}$$

The name "evidence lower bound" comes from the fact that $\log \mathbb{E}_{q(x)} p(x) \geq \mathbb{E}_{q(x)} \log p(x)$ and that $D_{KL}(q(\mathbf{m}) \parallel p(\mathbf{m}|\mathbf{d})) \geq 0$, resulting in the following inequality (Jordan et al., 1999):

$$\log p(\mathbf{d}) \geq \mathbb{E}_{\mathbf{m}\sim q} \log p(\mathbf{m}, \mathbf{d}) - \mathbb{E}_{\mathbf{m}\sim q} \log q(\mathbf{m}) = ELBO. \tag{7}$$

As we maximize the ELBO in Eq. (6), it approaches $\log p(\mathbf{d})$. We define a corresponding loss function $\mathcal{L}(\phi) = ELBO$ which depends on the parameterization ϕ of the variational density

$$\begin{aligned} \mathcal{L}(\phi) &= \int q_\phi(\mathbf{m}) \log p(\mathbf{m}, \mathbf{d}) d\mathbf{m} - \int q_\phi(\mathbf{m}) \log q_\phi(\mathbf{m}) d\mathbf{m} \\ &= \int q_\phi(\mathbf{m}) \log \frac{p(\mathbf{m}, \mathbf{d})}{q_\phi(\mathbf{m})} d\mathbf{m} = \mathbb{E}_{\mathbf{m}\sim q} \left[\log \frac{p(\mathbf{m}, \mathbf{d})}{q_\phi(\mathbf{m})} \right]. \end{aligned} \tag{8}$$

Then, ϕ is optimized to maximize $\mathcal{L}(\phi)$ (and as a consequence it also minimizes $D_{KL}(q_\phi(\mathbf{m}) \parallel p(\mathbf{m}|\mathbf{d}))$ via gradient-based optimization in which gradients of $\mathcal{L}(\phi)$ are computed with respect to ϕ using samples from $q_\phi(\mathbf{m})$

$$\nabla_\phi \mathcal{L}(\phi) = \mathbb{E}_{\mathbf{m}\sim q} \left[\nabla_\phi \log \frac{p(\mathbf{m}, \mathbf{d})}{q_\phi(\mathbf{m})} \right]. \tag{9}$$

An unbiased Monte Carlo estimation of the ELBO (and its derivatives) can be computed by evaluating the logarithmic ratios in Eqs. (8) and (9) at N_s samples from $q_\phi(\mathbf{m})$.

2.3. Deep generative models

DGMs are artificial neural networks that are trained to generate data according to an underlying distribution of a dataset of interest. A network is composed of input, hidden and output layers, where the input is our input features, the output is our generated data and hidden layers are intermediate layers connecting the input to the output. The hidden layers are composed of small units (nodes) referred to as neurons. Mathematically, a hidden layer can be formulated as:

$$h(\mathbf{X}) = \varphi(\mathbf{X}^{\mathrm{T}}\mathbf{W} + \mathbf{b}), \tag{10}$$

where $\mathbf{X}$ is the input vector to the layer, $\mathbf{W}$ contains the weights connecting input features to individual neurons, $\mathbf{b}$ is a vector of biases and φ is an activation function (sigmoid, tanh, ReLU etc.) introducing non-linearity. In convolutional neural networks (CNNs), such as those used herein, weights in each layer are organized in a series of matrices (kernels) that are convolved with the input to form a series of feature maps. Convolutional networks reduces the number of weights required, especially for large inputs and they are advantageous in tasks where the input exhibits local interactions between features (see Goodfellow et al., 2016 for more information). In this study, we consider two types of DGMs: spatial generative adversarial network (SGAN) and a variational autoencoder (VAE). These DGMs are introduced to reduce the dimensionality of the inverse problem by learning an encoding of a complex prior, thereby, aiming at reducing the computational cost and improving inversion performance. Both DGMs are trained using the binary channelized image of size 2500 × 2500 pixels introduced by Zahner et al. (2016) and later used by Laloy et al. (2018) (Fig. 2a in their paper) and Lopez-Alvis et al. (2021).

2.3.1. Spatial generative adversarial networks

The SGAN (Jetchev et al., 2016; Laloy et al., 2018) is a type of generative adversarial network (GAN; Goodfellow et al., 2014), that is, a CNN consisting of a discriminator D and a generator G. Adversarial training consists of optimization with the generator and discriminator competing against each other. The input to the generator in a SGAN is a noise tensor $\mathbf{Z}$ of 2D or 3D shape, which is typically drawn from a standard normal or uniform distribution. For convenience, in this paper we represent $\mathbf{Z}$ in its vector form $\mathbf{z}$, however, in practice the input to the generator of the SGAN is a tensor of rank that is higher than one. For a 2D model domain, the output is an image $\tilde{\mathbf{X}}$ of size $m \times n \times q$, where q represent the number of RGB channels. The size of $\tilde{\mathbf{X}}$ is determined by the depth of the network as well as the number of spatial parameters (m and n). The significance of having an input tensor in a SGAN as opposed to a 1D vector in a standard GAN, is the way perturbations in the latent space are translated into changes in the image space. As opposed to a global update, perturbing one of the SGAN's latent parameters leads to a localized change in $\tilde{\mathbf{X}}$. The input to the discriminator is either the generated image $\tilde{\mathbf{X}}$ or an image $\mathbf{X}$ from a training set, containing the patterns we would like to learn. The output of the discriminator is a score of either 0 or 1, representing the belief that the input is either generated by the generator or is a part of the training set (i.e., training image), respectively. The network is trained using the following minimization–maximization loss function:

$$\min_{G(\cdot)} \max_{D(\cdot)} \mathop{\mathbb{E}}_{\mathbf{X}\sim P_r} [\log D(\mathbf{X})] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p_g} [\log(1 - D(G(\mathbf{z})))]. \tag{11}$$

The discriminator D will try to maximize the function in Eq. (11) by correctly labeling $\mathbf{X}$ as 1 and $G(\mathbf{z})$ as 0, while the generator G will try to minimize it through fooling the discriminator. For numerical stability, an l_2-norm regularization $\alpha_{GAN}\|\Omega\|_2^2$ is applied to both the generator and discriminator, where Ω contains the network weights and regularization increases as we increase α_{GAN}, the weighting factor. This type of regularization encourages the individual weights to be small, thus, preventing large weights on a few layer units (neurons). The loss from Eq. (11) is then used to update the parameters of the discriminator and generator, where the weights of the discriminator are updated first and the weights of the generator are updated in a second stage. The update to the parameters of the network is performed by back-propagating the error computed in the forward pass (going from input to output) through the respective network. The update to each network parameter is proportional to a specified learning rate and the gradients of the error with respect to that parameter (see Laloy et al. (2018) for more details).

We adopt the SGAN architecture of Laloy et al. (2019) who used a generator with seven convolutional layers, instance normalization and ReLU activation function except for the last layer which is only followed by a tanh activation function. We train the network with a square latent space $\mathbf{z}$ of 12×12 out of which we use only 5×3 (15) latent variables to generate images of size 65×129. The training images are normalized into a range of $[-1, 1]$ before they are fed into the discriminator. Consequently, when using the trained generator, images are also re-scaled into the $[0, 1]$ range. We train the SGAN with the ADAM optimizer (Kingma and Ba, 2014) using a batch of 32 images at each training iteration and the following hyperparameters: $\beta_{GAN,1} = 0.5$, $\beta_{GAN,2} = 0.999$, learning rate of $5e^{-4}$ and $\alpha_{GAN} = 1e^{-7}$.

2.3.2. Variational autoencoders

Variational autoencoders are a type of generative models proposed by Kingma and Welling (2014) for various deep learning tasks (e.g. recognition, denoising, representation and visualization) involving intractable posteriors. VAEs include two neural networks: a probabilistic encoder described by $q_\vartheta(\mathbf{z}|\mathbf{X})$ and a probabilistic decoder described by $p_\theta(\mathbf{X}|\mathbf{z})$, where ϑ and θ are the parameters of the encoder and decoder, respectively. The former transforms an input $\mathbf{X}^{(i)}$ from the training set $\{\mathbf{X}_{i=1}^N\}$ into a probabilistic n-dimensional representation $\mathbf{z}$ and the latter samples $\mathbf{z}$ and transform it into $\tilde{\mathbf{X}}^{(i)}$, that is, a reconstruction of $\mathbf{X}^{(i)}$. The training objective is to maximize the ELBO $\log p(\mathbf{X})$ (Kingma and Welling, 2014):

$$\mathcal{L}(\vartheta, \theta) = \mathbb{E}_{q_\vartheta(\mathbf{z}|\mathbf{X})} \left[\log(p_\theta(\mathbf{X}|\mathbf{z}))\right] - D_{KL}(q_\vartheta(\mathbf{z}|\mathbf{X}) \parallel p(\mathbf{z})). \tag{12}$$

The first term represents the reconstruction error of the decoder when transforming samples from $\mathbf{z}$ into $\tilde{\mathbf{X}}$ while the second term encourages the variational density $q_\vartheta(\mathbf{z}|\mathbf{X})$ to be close to $p(\mathbf{z}) \equiv \mathcal{N}(\mathbf{0}_n, \mathbf{I}_n)$. The model becomes non-differentiable if we sample $\mathbf{z}$ directly from a distribution parameterized by the output of the encoder as we would need to compute the gradients with respect to a random sample. This is problematic for gradient-based optimization during which gradients are back-propagated through the network. In order to solve this problem, $\mathbf{z}$ is re-parameterized using a random auxiliary noise $\boldsymbol{\epsilon}$ such that (Kingma and Welling, 2014):

$$\tilde{\mathbf{z}} = \boldsymbol{\mu}_\vartheta(\mathbf{X}) + \boldsymbol{\sigma}_\vartheta(\mathbf{X}) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{13}$$

where $\odot$ denotes an element-wise product and $\boldsymbol{\mu}_\vartheta$ and $\boldsymbol{\sigma}_\vartheta$ are mean and standard deviation vectors provided by the encoder. After reparameterization, $\mathbf{z}$ becomes deterministic and gradients can be back-propagated through it. Here we use the VAE proposed by Lopez-Alvis et al. (2021) which has the same layer architecture as the SGAN and was trained on the same training images. Although VAE can be fully probabilistic, Lopez-Alvis et al. (2021) considered only the mean of the decoder, therefore, making it a deterministic generator $G_\theta(\mathbf{z})$. After training, θ is constant and to generate $\tilde{\mathbf{X}}$ samples, we simply draw samples from $\tilde{\mathbf{z}} \approx \mathcal{N}(\mathbf{0}_n, \mathbf{I}_n)$ and push it through the generator $G_\theta(\mathbf{z})$. Lopez-Alvis et al. (2021) discuss the importance of two hyper-parameters that needs to be specified when training the VAE: β_{VAE} which is a weighting factor multiplying the second term in Eq. (12) and α_{VAE} which controls the distribution from which the auxiliary noise is drawn from: $p(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0}_n, \alpha_{VAE} \cdot \mathbf{I}_n)$. They illustrate how well-chosen α_{VAE}- and β_{VAE}-values leads to a well-behaved generator and better inversion performance relative to other choices. The hyper-parameters with which the VAE is trained are as follows: $\beta_{VAE} = 1000$, $\alpha_{VAE} = 0.1$ and a learning rate of $1e^{-3}$ (for more details, see Lopez-Alvis et al., 2021). The VAE decoder contains two fully connected layers followed by four convolutional layers that are all followed by instance normalization and ReLU activation function except for the last layer which is only followed by a sigmoid activation function. The latent space $\mathbf{z}$ of the VAE is composed of a vector of 20 parameters corresponding to output images of 65×129 pixels.

2.4. Crosshole traveltime tomography

In our test examples, we consider a crosshole ground penetrating radar (GPR) setup in which source and receiver antennas are distributed within two vertically-oriented boreholes. The forward response can be formulated as follows:

$$\mathbf{d} = g(\mathbf{s}) + \boldsymbol{\epsilon}, \tag{14}$$

where g is the forward operator, $\mathbf{s}$ is the slowness field (inverse of velocity $\mathbf{v}$) of the modeled domain, $\boldsymbol{\epsilon}$ is the observational noise and $\mathbf{d} = (d_1, \ldots, d_N) \in \mathcal{R}^N$ with $N \geq 1$ is the measured first-arrival travel times between source–receiver pairs. We assign velocities to individual model parameters using $v = 0.06 + 0.02 \cdot (1 - \tilde{x})$, resulting in a continuous range of velocities between $[0.06, 0.08]$ m ns^{-1}.

We consider a non-linear forward solver implemented in the pyGIMLi geophysical modeling library (Rücker et al., 2017). In this implementation, the travel times are calculated on a mesh of nodes based on the Dijkstra method, giving the shortest path between source–receiver positions for a given slowness model. We use the Jacobian provided by pyGIMLi for a given source–receiver geometry and slowness model and calculate the travel times according to:

$$\mathbf{d}_{sim} = \mathbf{J}_g(\mathbf{s})\mathbf{s}, \tag{15}$$

where $\mathbf{J}_g$ is the Jacobian matrix (also known as the sensitivity matrix) containing the length of the ray segment at each model cell for each travel time. Note that this Jacobian refers to the physical forward solver and not to the Jacobian of the IAF (Eq. (2)). The accuracy of the simulated travel times can be improved by adding secondary nodes. In our examples, the Jacobian is re-computed for each slowness field using two secondary nodes. With the Jacobian acting as the forward operator (Eq. (15)) we are able to comply with the automatic differentiation (auto-differentiation) requirements of machine-learning supported Python libraries (e.g., PyTorch and TensorFlow). Unfortunately, pyGIMLi objects are not supporting data storage using pickling which is a requirement when using most parallel computing Python libraries. Given this limitation, in this work the forward simulations are performed in a sequential manner.

2.5. Inversion in the latent space of a deep generative model with neural-transport

Our inversion framework combining NT with a DGM is composed of the methods describe in previous subsections: IAF, VI and DGMs, where the forward response is required in order to compute the joint probability $p(\mathbf{m}, \mathbf{d})$. Both DGMs define a low-dimensional latent space involving uncorrelated variables with a well-defined prior (standard normal) that we choose to be in agreement with the base distribution of the IAF. As both DGMs are implemented in PyTorch, we use Pyro (Bingham et al., 2019), a library for probabilistic programming built on Python and PyTorch, to train the IAF. In the following, we define $\mathbf{z}^{(0)}$ as a random variable within the latent space of the IAF that is drawn from a standard normal base distribution, we further define our target distribution q_ϕ on the latent space of the SGAN (or VAE) such that $\mathbf{m} = \mathbf{z}^{GAN}$ (or $\mathbf{z}^{VAE}$) and $\tilde{\mathbf{X}}$ as the high-dimensional, image-space parameters before slowness $\mathbf{s}$ is assigned. For the remainder of this paper, we will refer to random samples drawn from the base distribution (and pushed forward to the variational distribution space) as particles. Each particle represents one model realization and has the same size as that of the latent space of the DGM in use. For the sake of conciseness, we will refer to variables from the target distribution $\mathbf{z}^{GAN}$ (or $\mathbf{z}^{VAE}$) simply as $\mathbf{z}$ and specify in the appropriate places to which generative model they belong.

To train the IAF, N_s particles are drawn from the base distribution $\mathbf{z}^{(0)} \sim q(\mathbf{z}^{(0)})$ and mapped through the invertible transformation of the IAF into the variational space $q_\phi(\mathbf{z})$ in which we approximate the posterior on the DGM's latent space $\mathbf{z}$. The N_s particles $\mathbf{z}$ are then transformed into high-dimensional $\tilde{\mathbf{X}}$-realizations through the generator $G(\cdot)$. Slowness $\mathbf{s}$ is assigned to each pixel and the likelihood is computed for each of the N_s particles using the geophysical forward solver. We compute the logarithm of the joint distribution $\log p(\mathbf{z}, \mathbf{d}) = \log p(\mathbf{d}|\mathbf{z}) + \log p(\mathbf{z})$ (also referred to as the logarithmic form of the unnormalized posterior $p(\mathbf{z}|\mathbf{d})$). Since we have a standard-normal prior (mean $\mu_z = 0$ and standard deviation $\sigma_z = 1$) on both the SGAN and VAE latent spaces, and further assume independent, identical and normally-distributed observational noise with zero mean and standard deviation of σ_d, we have

$$\log p(\mathbf{z}, \mathbf{d}) = -\frac{1}{2}\left(N_d \log(2\pi) + 2N_d \log(\sigma_d) + \sigma_d^{-2} \sum_{i=1}^{N_d} \left[d_i - g_i(G(\mathbf{z}))\right]^2 + N_z \log(2\pi) + 2N_z \log(\sigma_z) + \sigma_z^{-2} \sum_{i=1}^{N_z} z_i^2 \right), \tag{16}$$

where N_d is the number of data observations, N_z is the number of latent $\mathbf{z}$ parameters and z_i is the ith parameter in $\mathbf{z}$. Note that the log-likelihood is evaluated on forward simulations based on the high-dimensional $\tilde{\mathbf{X}}$-space while the log-prior is evaluated on the low-dimensional SGAN (or VAE) latent parameters. The loss function $\mathcal{L}(\phi)$ can be calculated by using Eq. (3) to evaluate $\log q_\phi(\mathbf{z})$:

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{z}, \mathbf{d})}{q_\phi(\mathbf{z})}\right] = \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p(\mathbf{z}, \mathbf{d})}{q(\mathbf{z}^{(0)}) \prod_{k=1}^{K} \left|\det \frac{d f_k(\mathbf{z}^{(k-1)})}{d\mathbf{z}^{(k-1)}}\right|^{-1}}\right]. \tag{17}$$

The gradient of $\mathcal{L}(\phi)$ is computed through auto-differentiation. We consider $-\mathcal{L}(\phi)$ and perform stochastic gradient descent to update ϕ. A brief summary of the above routine appears in Fig. 1 and Algorithm 1.

The architecture of the IAF can be adjusted in response to the level of complexity of the target distribution. Hoffman et al. (2019) used three stacked flows with two hidden layers each. We found that two sequential flows, each containing one hidden layer and a hidden dimensionality that is twice as large as the target distribution to be sufficient for our considered examples. Each flow is followed by a non-linear ReLU activation function providing the network with further flexibility (For detailed information about the architecture of the IAF see Appendix A). During training the network parameters are optimized using ADAM with $\beta_{IAF,1} = 0.9$ and $\beta_{IAF,2} = 0.999$ and a learning rate of 0.01. In order to enable gradient calculation of the model parameters with respect to the DGM as part of the NT routine, we do not threshold the generated images to $[0, 1]$ (Laloy et al., 2019).

Algorithm 1: Bayesian inference using neural-transport and a deep generative model

set T (maximum number of iterations) and $t = 0$
while $(t < T)$ **do**
  Draw N_s particles (realizations of $\mathbf{z}^{(0)}$) from the base distribution $q(\mathbf{z}^{(0)})$
  $\mathbf{z} \leftarrow \text{IAF}_\phi(\mathbf{z}^{(0)})$
  $\tilde{\mathbf{X}} \leftarrow \text{G}(\mathbf{z})$
  Assign slowness values to $\tilde{\mathbf{X}}$ and compute the forward simulation $g(\mathbf{s})$ to get simulated data $\mathbf{d}$
  Compute $\mathcal{L}(\phi)$ and $\nabla_\phi \mathcal{L}(\phi)$ using Eq. (9), (16) and (17) and update ϕ using stochastic gradient descent
  $t = t + 1$
end
begin G($\mathbf{z}$)
  Perform a series of transposed convolution layers with pre-trained weights
  return $\tilde{\mathbf{X}}$
end
begin $\text{IAF}_\phi(\mathbf{z}^{(0)})$
  $\mathbf{z} = f_k \circ f_{k-1} \circ \ldots \circ f_1(\mathbf{z}^{(0)})$
  return $\mathbf{z}$
end

2.6. Performance assessment

We test NT in combination with each of the two considered DGMs using five different test models (Fig. 2) generated by the respective generator. Following Lopez-Alvis et al. (2021), we refer to models generated by the SGAN with the abbreviation 'mg' and models generated by the VAE with 'mv'. It is seen that the SGAN provides images that are less blurry than those produced by the VAE. To assess the performance and quality of the approximate posterior $q_\phi(\mathbf{z})$ obtained from NT, we consider different statistical metrics. For each test model, we plot the mean and standard deviation image of the approximate posterior. The root-mean-squared error (RMSE) is computed on mean values of the latent parameters ($\text{RMSE}_\mathbf{z}$), model parameters ($\text{RMSE}_\mathbf{X}$) and data misfit ($\text{RMSE}_\mathbf{d}$) at the last iteration. We rely on $\text{RMSE}_\mathbf{d}$ to determine if the approximate posterior has converged. The loss function for the IAF is used here as a complementary metric as we observed that the data fit can decrease even after $\mathcal{L}$ becomes stable. We define two criteria

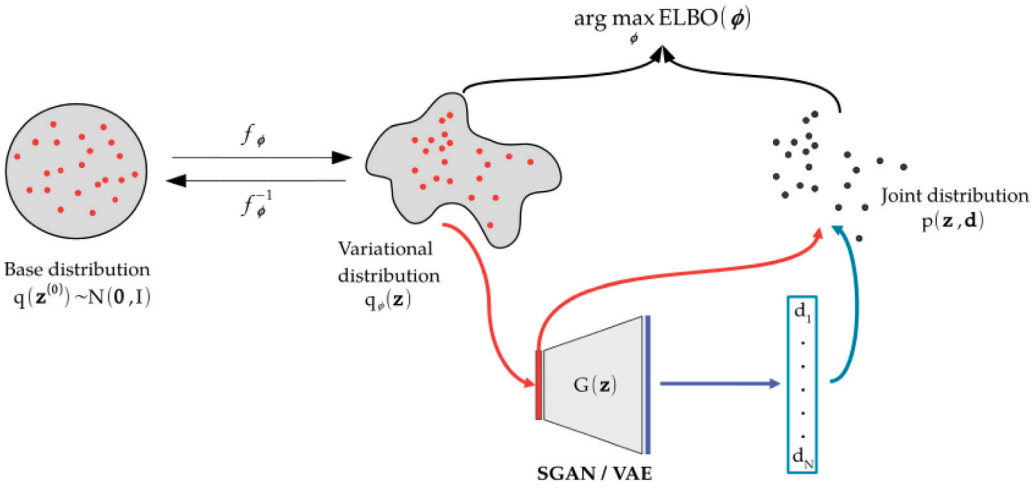

Fig. 1. Illustration of one training iteration of neural-transport combined with deep generative models. The inferred model parameters (here represented in 2D as dots) are the low-dimensional latent parameters **z**. The pre-trained generator G transform the latent parameters into their corresponding high-dimensional parameters in the image-space on which forward simulations are carried out to obtain the data vector **d**. The IAF (represented by transformation f) parameters ϕ, are tuned during training to maximize the ELBO.

that both need to be met to declare convergence: (1) for an iteration after which the $\text{RMSE}_\mathbf{d}$ value averaged over all particles equals that in the last 10% iterations of the algorithm; (2) the average WRMSE $= \sqrt{\frac{1}{N_d}\sum\left[\frac{d_i - g_i(G(\mathbf{z}))}{\sigma_i}\right]^2}$ (data misfit weighted by the standard deviation of the data noise) is less than 1.1. The former criterion ensures that the approximate posterior reached a stable solution while the latter prevents declaring convergence for models that are stuck in a local minimum with a data misfit that is poor. The similarity of the NT solution (mean value of the approximate posterior) to the true model in the high-dimensional space $\tilde{\mathbf{X}}$ is further assessed using the structural similarity index (SSIM; Wang et al., 2004). It measures the similarity between two images with respect to their structural information:

$$SSIM(\mathbf{u}, \mathbf{v}) = \frac{(2\mu_\mathbf{u}\mu_\mathbf{v} + C_1)(2\sigma_{\mathbf{uv}} + C_2)}{(2\mu_\mathbf{u}^2 + \mu_\mathbf{v}^2 + C_1)(2\sigma_\mathbf{u}^2 + \sigma_\mathbf{v}^2 + C_2)}, \tag{18}$$

where **u** and **v** are sliding windows of size $M \times M$, each sub-samples its respective $[0, 1]$ normalized image. The values of the SSIM range between -1 and 1, where 1 indicates perfectly matching images. Here we use $M = 7$, $C_1 = 0.01$ and $C_2 = 0.03$ as those values are commonly used (Wang et al., 2004; Laloy et al., 2021 and references therein).

Additionally, we assess the performance of the NT approach against the results obtained by MCMC. We use the differential evolution adaptive Metropolis ($\text{DREAM}_{(ZS)}$) algorithm (ter Braak and Vrugt, 2008; Vrugt et al., 2009; Laloy and Vrugt, 2012) to sample the posterior in the latent space of each considered DGM. In this MCMC algorithm, several chains evolve in parallel and jumps are proposed based on candidate points from an archive of past states. At each MCMC step and for each individual chain, a sample $\mathbf{z}'$ proposed according to a symmetric proposal distribution is either accepted or rejected according to a Metropolis acceptance probability

$$p_{acc}(\mathbf{z}^{t-1}, \mathbf{z}') = \min\left(1, \frac{p(\mathbf{d}|\mathbf{z}')p(\mathbf{z}')}{p(\mathbf{d}|\mathbf{z}^{t-1})p(\mathbf{z}^{t-1})}\right). \tag{19}$$

If accepted, the chain moves to the proposed state, if rejected it remains at the current state. Convergence of each latent parameter is declared based on the Gelman–Rubin diagnostic (Gelman and Rubin, 1992) when $\hat{R} \leq 1.2$. We compare the approximate posterior PDFs of **z** obtained from NT to those obtained from MCMC inversion. Furthermore, both posterior distributions are also compared to the prior of the latent space. The distance between two PDFs is computed using the KL-divergence while their predictive power is assessed using the logarithmic scoring rule (LogS; Good, 1952). The LogS statistic is defined as $\text{logS}(\hat{p}, \mathbf{y}) = -\log \hat{p}(\mathbf{y})$ where **y** are the true values of the parameters of interest and $\hat{p}$ is the PDF used to predict the probability of **y**. A lower LogS indicates a more accurate prediction. Both the KL-divergence and LogS values are reported as mean values over all parameters.

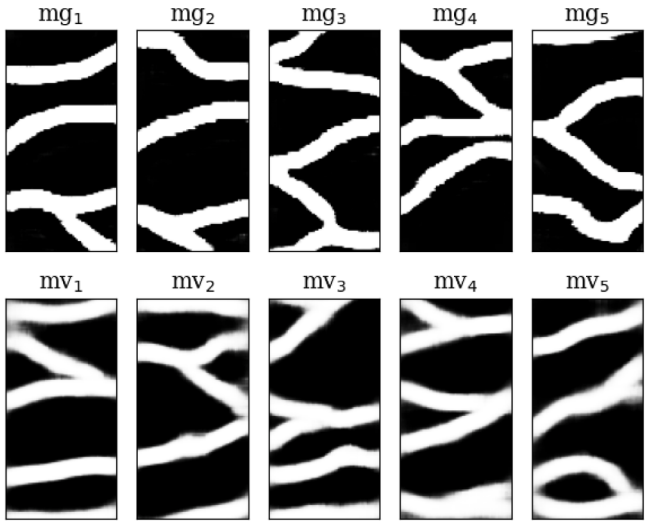

Fig. 2. Reference models used for inversion. Models labeled with: 'mg' are test models generated by the SGAN and 'mv' are test models generated by the VAE.

3. Inversion results

We consider the inversion of synthetic data created using the forward solver described in Section 2.4 that are contaminated with normally-distributed noise $\mathcal{N}(0, 1)$. We use 25 sources and 25 receivers resulting in 625 data points. Given results from the hyperparameter search described in Appendix B, we use 20 particles to perform inversion with the SGAN and only a single particle when using the VAE. After 250 iterations with 20 particles, the $\text{RMSE}_\mathbf{d}$ of NT with SGAN is still decreasing towards the target value (Fig. B.2a). Consequently, we increase the number of training iterations to 300 (6000 forward simulations) for the SGAN inversions. On the other hand, NT with VAE converges towards the target value in less than 1000 iterations with a single particle (Fig. B.2d). Since we perform only one forward simulation per iteration we allow a maximum of 2000 training iterations for the VAE inversions. The learning rate in both types of inversions is

set to 0.01. We run each NT-inversion scheme on a single CPU (AMD EPYC™ 7402) in a sequential manner due to the inability to distribute the forward response function on multiple CPUs (see Section 2.4). It takes around 13 h to run the VAE-based NT inversion considering 2000 training iterations and a single particle and around 40 hours to run the SGAN-based NT for 300 training iterations and 20 particles. The computational effort is completely dominated by the calculation of the Jacobian of the physical forward solver (Eq. (15)) at each iteration as it makes up 99% of the total computational time.

Figs. 3 and 4 for SGAN and VAE, respectively, show the mean and standard deviation of the approximate posterior compared to the true model as well as the ELBO loss and $\text{RMSE}_{\mathbf{d}}$ during inversion. These figures are complemented by quantitative metrics in Table 1. For all of the mg models in Fig. 3 that were obtained using SGAN as the DGM, the main features were reconstructed with the right number and location of the channels (Fig. 3b) and the largest uncertainty is located at the boundaries of the channels (Fig. 3c) as expected given results of previous studies (e.g., Zahner et al., 2016). The inferred models mg_2 and mg_3 are of lower quality compared to the other ones with SSIM values of 0.71 and 0.76, while it is ≥ 0.85 for the other mg models. The inferred model for mg_2 has large uncertainty (around 0.3) on the upper left 2 m of the model and exhibits the largest data misfit (1.42 ns while it is $\leq$ 1.10 ns for the other models). It is possible that these estimates would have improved further with more training iterations.

The results obtained when using the VAE as DGM are both qualitatively (Fig. 4) and quantitatively (Table 1) much better. Table 1 suggests that all mv models are well reconstructed with all $\text{RMSE}_{\mathbf{d}} \leq$ 1.05 ns and all SSIM $\geq$ 0.9. The $\text{RMSE}_{\mathbf{z}}$ values are as low as 0.08 and none is higher than 0.37, indicating a good match between the inferred latent parameters and their true counterparts. These values are at least one order lower than the values obtained for the SGAN-based inversion ($\geq$ 1.08). The $\text{RMSE}_{\mathbf{X}}$ is as low as 0.04 for the VAE models (mv_4) whereas the lowest value for the SGAN models is 0.10 (mg_1). Moreover, the mean standard deviation of $\mathbf{X}$, while highly dependent on the number of channel elements in the reference model, is consistently lower when the VAE is used as DGM (on average by $\sim 60\%$). Although there are noticeable differences in the bottom part of the inferred mv_1 and mv_2 models (last 2 m) compared to their references, these do not translate into large data misfits as these regions are not well constrained by the GPR rays. Furthermore, when we train the IAF with ten particles on the mv_1 model the reconstruction improves (see Appendix C).

Three out of the five mg models and all mv models converged according to the criteria in Section 2.6 (see Table 1). For the mg models, convergence occurs after 260 training iterations on average (with 20 particles), while for the mv models the stage at which convergence can be declared ranges between 477 and 1767 training iterations (with a single particle). Nonetheless, it can be seen in Fig. 4 that by the 800th iteration the $\text{RMSE}_{\mathbf{d}}$ for all the models is below 1.1 ns. This is in agreement with Fig. 5 where we take three exemplary latent parameters of mv_5 and plot their approximate PDF at different stages of the training process. We observe that after 500 training iterations the approximate density is close to the final density (2000 iterations) and after 1000 iterations the density becomes very similar to the final one for the first (Fig. 5a) and tenth (Fig. 5b) latent parameters.

After demonstrating that the SGAN- and particularly the VAE-based NT produce high-quality reconstructions of the true model, we assess now the corresponding approximate posteriors with respect to MCMC inversion ($\text{DREAM}_{(ZS)}$) and the standard normal prior PDF. We run eight parallel MCMC chains for one test model of each DGM: mg_5 and mv_5 (these are also the models resulting in the lowest $\text{RMSE}_{\mathbf{z}}$). We limit the number of samples per chain to 20 000 which represent a computation time of ~6 days on a 8-core workstation. For the SGAN-based MCMC inversion, 12 of the 15 parameters satisfy the $\hat{R}$ criterion within this computational budget. For the VAE-based MCMC inversion, all 20 parameters converged within 8900 samples per chain (total of 71 200 samples). Given the two different convergence criteria for NT and MCMC, we have that the computational time required for the VAE-based MCMC inversion to converge to the posterior target is 7 times larger than that required by the VAE-based NT (56 times if the MCMC algorithm would not have been running in parallel). After 20 000 MCMC samples per chain, the $\text{RMSE}_{\mathbf{z}}$s of the posterior means are 0.31 and 0.09 for the SGAN- and VAE-based MCMC inversions, respectively. The mean SSIM values in the model space that correspond to those posterior latent parameters are 0.91 (mg_5) and 0.95 (mv_5). In addition, the final $\text{RMSE}_{\mathbf{d}}$ averaged over all chains is about 1.03 ns for both DGM-based MCMC inversions. Compared to NT the MCMC achieves lower RMSE values and higher SSIM values when the SGAN is the DGM, but the performance is comparable for the VAE-based inversion.

Table 1
Summary of the results obtained from inversion with neural-transport for various reference models and using either the SGAN (20 particles) or the VAE (one particle) as DGM. $\overline{S}(\mathbf{X})$ is the average standard deviation of the posterior $p(\mathbf{X}|\mathbf{d},\mathbf{z})$ corresponding to samples from $q_\phi(\mathbf{z})$. $\text{RMSE}_{\mathbf{z}}$ is calculated on the mean latent space parameters of the resulting posterior while the $\text{RMSE}_{\mathbf{X}}$ is calculated in the model image space. $\text{RMSE}_{\mathbf{d}}$ is the data misfit RMSE value at the last iteration. The SSIM is calculated in the model image space on $[0, 1]$ models.

DGM	Model	Converged (iteration #)	$\overline{S}(\mathbf{X})$	RMSE			SSIM
				$\mathbf{z}$	$\mathbf{X}$	$\mathbf{d}$ [ns]	
SGAN	mg_1	269	0.024	1.15	0.10	1.07	0.91
	mg_2	–	0.110	1.35	0.20	1.42	0.71
	mg_3	–	0.054	1.39	0.19	1.10	0.76
	mg_4	243	0.027	1.59	0.12	1.08	0.86
	mg_5	271	0.040	1.08	0.12	1.06	0.85
VAE	mv_1	1767	0.020	0.37	0.12	1.05	0.94
	mv_2	1467	0.017	0.18	0.06	1.04	0.96
	mv_3	477	0.024	0.15	0.09	1.04	0.90
	mv_4	496	0.020	0.08	0.04	1.04	0.97
	mv_5	1256	0.020	0.08	0.06	1.04	0.94

For comparison, we plot the marginal prior and posterior latent distributions obtained by performing both NT and MCMC sampling for the mg_5 (SGAN) and mv_5 (VAE) true models (Figs. 6 and 7). The LogS of each PDF and the KL-divergence values between the various PDFs are provided in Table 2. The marginal posteriors obtained for the mg_5 model (Fig. 6) are considerably wider than those obtained for mv_5 (Fig. 7). This can be also observed in the range of the posterior standard deviation displayed in Figs. 3c and 4c. The posterior derived by the SGAN-based NT is often not centered around the true value and receives the highest LogS (7.66; see Table 2). Moreover, when the SGAN is used as DGM the KL-divergence value between NT and MCMC posteriors goes to infinity due to a minimal overlap. The estimates are more consistent when using the VAE. Here the latent posterior PDF (Fig. 7) is either centered around or contains the true value for both the NT inversion and the MCMC inversion and posterior uncertainty derived by the NT is similar to that of the MCMC. Furthermore, the KL-divergence of the NT posterior from the MCMC posterior is relatively small (0.19), which indicates strong similarity between the two. The two VAE-based approximate posteriors also provide relatively similar LogS value with the NT posterior being slightly more accurate than the MCMC one (−1.29 for NT versus −1.19 for MCMC).

4. Discussion

Our results demonstrate that the presented NT approach works well with both SGAN and VAE in terms of reconstructing the true models in the image space. Moreover, the approximate posterior from the VAE-based inference provides a slightly better prediction (low LogS value) than MCMC as well as reliable uncertainty estimates with respect to the true latent space values (Fig. 7 and Table 2) at a much lower computational cost. Due to the invertible transformations of the IAF, in NT we can evaluate the approximate posterior analytically as well as draw random samples from it.

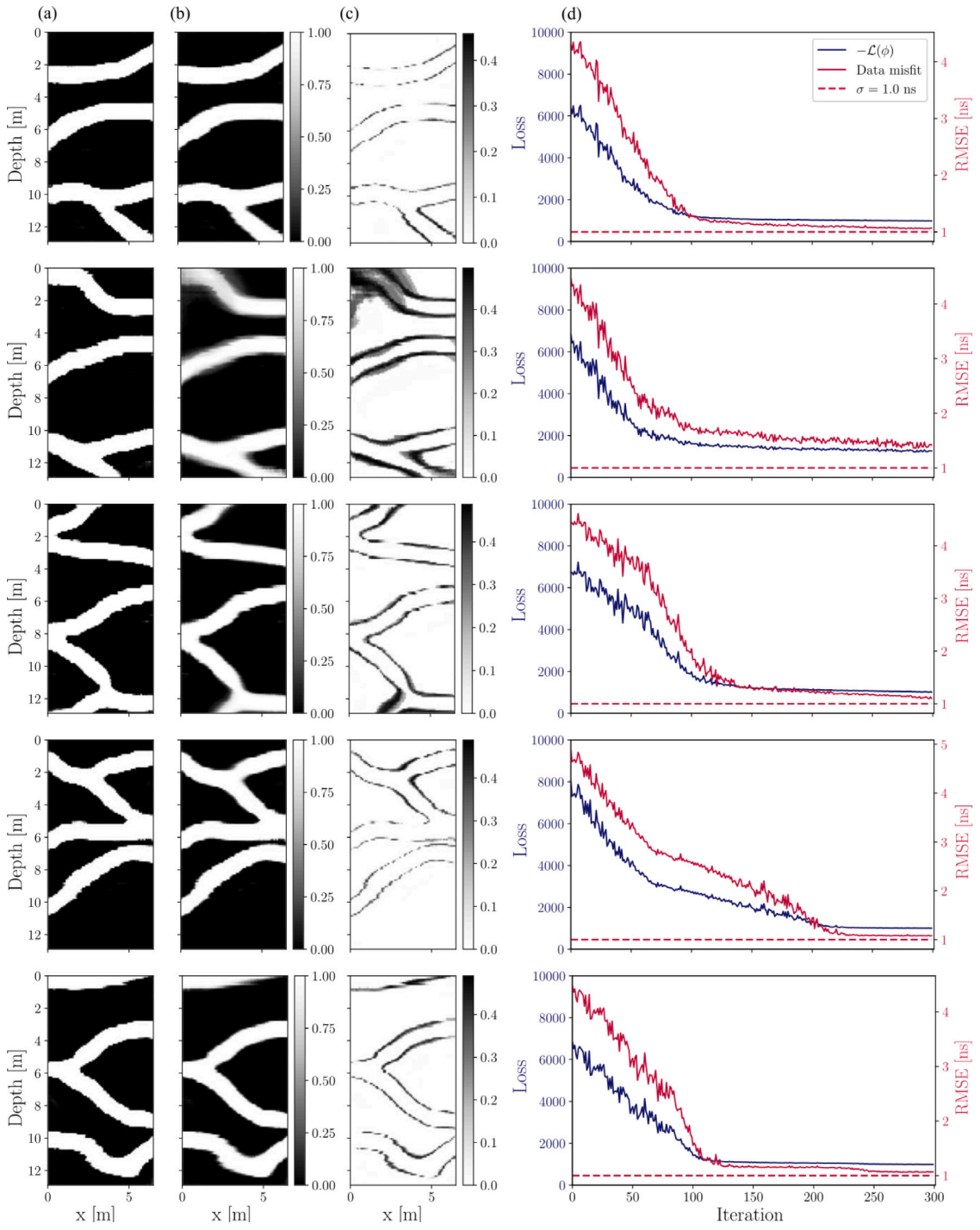

Fig. 3. Inferred posterior distributions for different reference models generated by the SGAN using 20 particles and 300 training iterations. (a) True mg_1-mg_5 models (b) mean posterior models obtained from NT and (c) posterior standard deviation in the model image space. (d) The ELBO loss and $RMSE_d$ in ns during training. The $RMSE_d$ curves represent the average values over all particles.

The differences in training (adversarial versus variational) and more so the differences in architecture (fully connected and convolutional layers in the VAE versus fully convolutional spatial GAN) between the two DGMs lead to transformations that vary in their degree of non-linearity. The SGAN transformation provides approximate latent posterior distributions that are both wider and less accurate than those obtained by the VAE-based inversions, thereby, indicating stronger dependencies between the SGAN parameters. The stronger correlation between the SGAN latent parameters can be explained by its spatial architecture, that is, its 2D latent space and fully convolutional layers. Although the approximate posterior in the latent space of the SGAN does not provide a good prediction of the true value, as seen from the $RMSE_z$ and LogS values, the reconstruction of the actual model (in the original high-dimensional image space) is reasonable with SSIM values in the 0.71–0.91 range. This suggests that two latent vectors that lie far from each other may correspond to similar realizations in the high-dimensional model space. This can be explained as an attempt of the generator to accommodate the change in topology between the latent space and the real manifold in the high-dimensional space (Lopez-Alvis et al., 2021). Moreover, Lopez-Alvis et al. (2021)

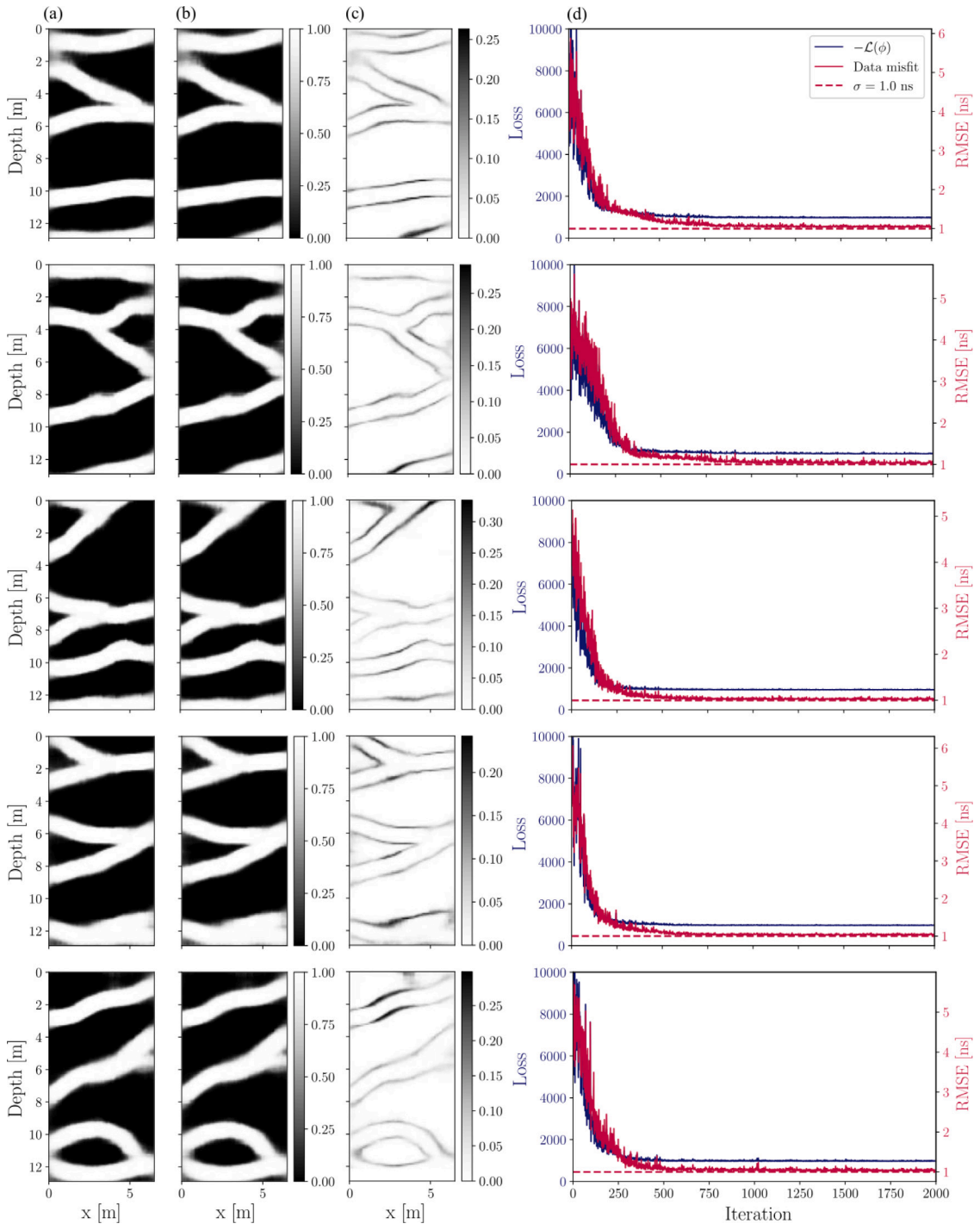

Fig. 4. Inferred posterior distributions for different reference models generated by the VAE using one particle and 2000 training iterations. (a) True mv_1-mv_5 models (b) mean posterior models obtained from NT and (c) posterior standard deviation in the model image space. (d) The ELBO loss and $RMSE_d$ in ns during training.

showed that the convexity of the misfit function in the latent space of the VAE can be controlled by choosing the right hyper-parameters during training. This is an important advantage, as Laloy et al. (2019) showed that the misfit function in the latent space of the SGAN is in fact a rough surface containing many local minima. Instead, the VAE is trained in such a way that both the changes induced in topology and the convexity of the misfit function are controlled. However, one must also keep in mind that this comes at the expense of generation accuracy and that realizations generated by the VAE are more lossy and, consequently, less sharp than those obtained from a GAN (Hou et al., 2017; Bao et al., 2022; see also our Fig. 2).

We stress that variational inference is limited by the parameterization of the approximate distribution, hence, in some cases the solution might not converge to an appropriate approximation of the posterior if the parameterized distribution is not expressive enough. A further improvement, for example in the case of the SGAN-based inversion, can be achieved by running traditional MCMC or Hamiltonian Monte Carlo (HMC; Duane et al., 1987; Neal, 2011) within the latent space of the IAF (Hoffman et al., 2019; Papamakarios et al., 2021). In this setting, the normalizing flow is used to reparameterize the posterior distribution and the starting point for the MCMC sampler is the approximation resulting from training the IAF, thereby, providing a much shorter

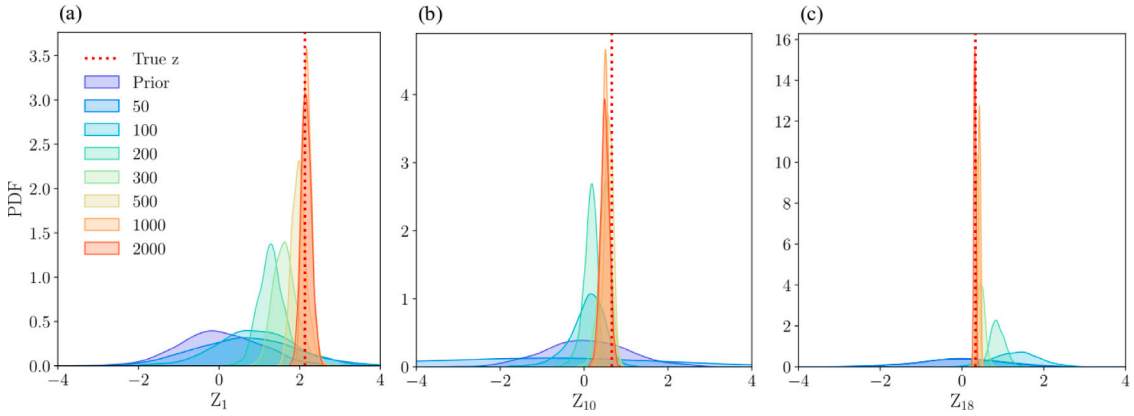

Fig. 5. Estimation of the variational PDF describing the marginal posterior of the 1st, 10th and 18th latent parameters of the mv_5 model at various training iterations.

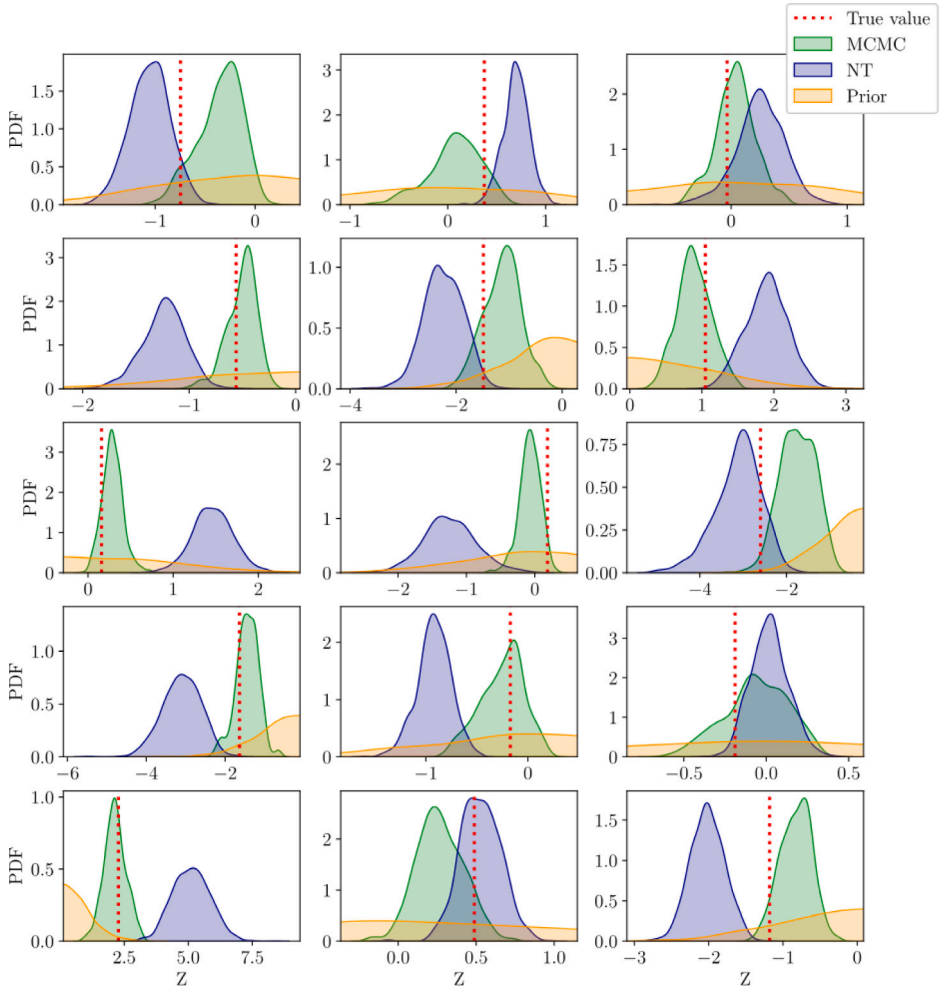

Fig. 6. Approximate marginal posteriors on the latent space **z** obtained with neural-transport (NT) and MCMC for model mg_5 in Fig. 2 as well as the prior on the latent space of the SGAN.

burn-in. Additionally, it provides a favorable sampling geometry from a standard normal which may improve MCMC mixing in multi-modal problems (Nijkamp et al., 2020).

The results for the VAE-based inversion (Fig. 4) demonstrate that NT can be performed with a single particle. The SGAN-based inversion on the other hand requires more than a single particle and starts

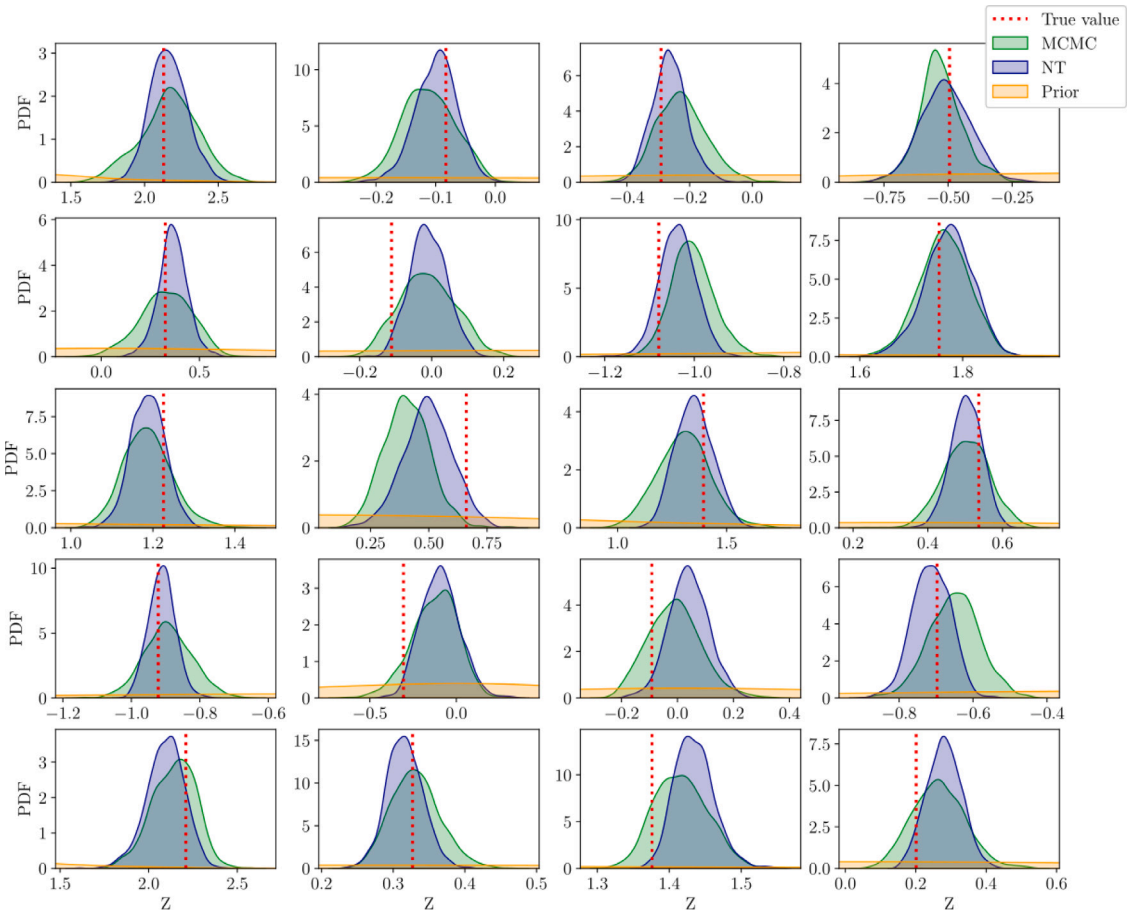

Fig. 7. Approximate marginal posteriors on the latent space **z** obtained with neural-transport (NT) and MCMC for model mv_5 in Fig. 2 as well as the prior on the latent space of the VAE.

Table 2
Statistical summary of the posterior PDF in the latent space **z** of mg_5 and mv_5 models obtained by neural-transport (NT) and MCMC with $DREAM_{(ZS)}$. Both are compared against the prior PDF of **z**. The logS and KL-divergence are reported as the mean value over the **z** parameters.

	PDF (Q)	logS	$D_{KL}(Q\|\|P)$	
			MCMC (P)	Prior (P)
mg_5	NT	7.66	inf	3.63
	MCMC	0.11	0	1.48
	Prior	1.60	–	0
mv_5	NT	−1.29	0.19	2.84
	MCMC	−1.19	0	2.55
	Prior	1.43	–	0

to perform well (measured in terms of data misfit only) when the number of particles is increased to 20 for the considered case studies (Fig. B.2). This finding is consistent with those by Laloy et al. (2019) and Lopez-Alvis et al. (2021), where deterministic gradient-based inversions within the low-dimensional space of the SGAN was found to perform poorly due to the highly non-linear SGAN transformation and small-scale irregularities in the objective function. Increasing the number of particles allows for more regions in the latent/model space to be explored at each iteration, thereby providing a more robust gradient estimation. This is perhaps particularly important at the initial phase of inversion where a vast region of the prior is explored. As opposed to many other gradient-based methods, NT involves random sampling at each iteration, which makes it more robust and reduces the risk of getting stuck in a local minimum. Increasing the number of particles is also shown to result in earlier convergence, however, it comes at the cost of an increased computational expense as evolving one particle for one iteration involves a forward simulation and the calculation of its Jacobian. For instance, for the SGAN-based inversions with 20 particles it takes an average of 260 training iterations (among those who have reached convergence) to converge, which translates into 5200 forward simulations. In contrast, the maximum number of iterations needed for the single-particle VAE-based inversion to converge is 1767 forward simulations only (see Table 1). Those SGAN-based inversion cases which have not converged possibly require either more training iterations or more particles. Nevertheless, among the converged cases using either the VAE or the SGAN, the total number of forward simulations needed in the NT approach is always much lower than in MCMC. When compared based on their individual convergence criteria, the computational times required by MCMC and NT differ by a factor of 7 in favor of NT. This factor would be 56 if the eight MCMC chains were not evolved in parallel.

In our NT applications, the forward simulations of the particles are computed sequentially. However, the computational time when considering multiple particles can be significantly reduced by distributing the computations associated with individual particles over several processing units (preferably using one unit per particle). This option is available in NT-based inversions: transformations and forward simulations can be performed in parallel when using more than a single particle, given that the forward simulation is parallelizable. Note that auto-differentiation as performed by machine learning libraries such as PyTorch and TensorFlow requires the forward solver to be implemented in the library in use, or alternatively, that the gradients of the forward response are provided by the user (Richardson, 2018; Laloy et al., 2019). As mentioned in Section 2.5, to maintain a differentiable operation we do not threshold the generated images to a binary value

of 0 or 1 in neither the SGAN nor the VAE generations. This limitation might affect inversion performance when the inverted model is either binary or categorical.

Using DGMs results in a drastic reduction in the number of inferred parameters (here from 8385 to only 15 and 20 SGAN and VAE latent parameters, respectively) as well as realizations which honor the higher-order statistical features of the model as represented by a training image (Laloy et al., 2017, 2018). The NT mechanism on the other hand, leverages on gradient information, random drawing of particles and flexible parameterization of the approximate posterior distribution. Consequently, NT combined with a DGM forms an efficient and scalable approach for solving high-dimensional inverse problems. As discussed in Section 3, most of the computational cost of the NT approach comes from the forward simulation and the largest updates to the IAF parameters occur at early training stages. Therefore, further improvement of NT efficiency could probably be gained by updating the Jacobian of the forward solver in Eq. (15) less frequently as the inversion advances. Another option could be to set a large number of particles at the beginning of the inversion and gradually decrease it as updates to the IAF parameters are becoming smaller. We leave these two options for future studies. Additionally, our study was limited to channelized subsurface models and a weakly non-linear forward solver operating in a crosshole setting. Further research is required to assess the performance of this approach for different geomodels, physical models (e.g. fluid flow, wave-based reflection data) and 3D problems.

5. Conclusions

Neural-transport refers to the application of variational Bayes to train an IAF transformation, which maps samples from a simple base distribution into samples from an approximate posterior over the latent space of a DGM. We demonstrate that inferring a model in the latent space of either a SGAN or a VAE using neural-transport significantly reduces the number of forward simulations required compared to MCMC sampling. In this respect, DGMs play an important role in improving the efficiency and scalability of the NT approach when dealing with geophysical inverse problems that are generally high-dimensional. Our results are in agreement with previous works concerning deterministic inversions performed in the latent space of a SGAN or a VAE. Indeed, the VAE is found to provide a better reconstruction of the true model in both the low-dimensional latent and high-dimensional model image spaces; the agreement with the MCMC results were also excellent. NT combined with SGAN provides a reasonable estimation in the high-dimensional model space and much better results overall compared to previous results based on deterministic inversions. In contrast to MCMC, where the posterior is estimated based on an ensemble of samples, NT provides a closed-form solution of the approximate posterior such that it can be efficiently evaluated and sampled from. Performance of NT-based inversion could be further improved by combining it with MCMC sampling within the latent space of the NT, starting from the solution of the NT-based inversion.

Computer code availability

Neural-transport and DGMs scripts as well as test examples are available at the following GitHub repository: https://github.com/ShiLevy/Neural_transport_DGM.

CRediT authorship contribution statement

Shiran Levy: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft. **Eric Laloy:** Software, Writing – review & editing. **Niklas Linde:** Conceptualization, Methodology, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Code is available at the provided GitHub repository under the "computer code availability" section.

Acknowledgments

This work was supported by the Swiss National Science Foundation (project number: 184574).

Appendix A. IAF design

The IAF is constructed as sequential flows with each of them producing a different distribution (see Fig. A.1a). Each flow involves an autoregressive network which takes as input either variables from the base distribution (if it is the first flow) or variables that are a result of the preceding flow. The output of the autoregressive network is a mean μ and logarithm of the standard deviation $\log(\sigma)$ (to prevent negative standard deviation as output, it is therefore exponentiated to get σ). To achieve the autoregressive property, the connections

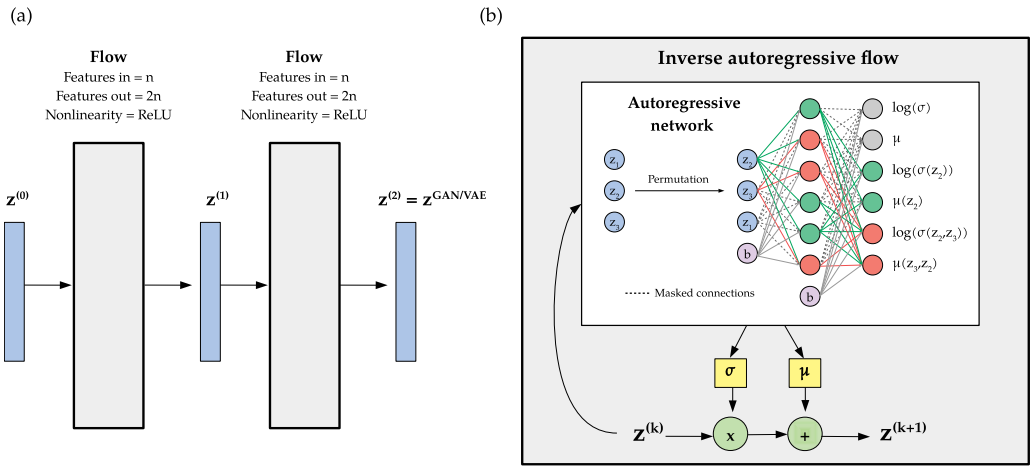

Fig. A.1. Schematic drawing of the IAF architecture. (a) General workflow of the IAF with two flows corresponding to one intermediate distribution. The number of neurons depends on the number of input features n. (b) An illustration of an autoregressive network in which connections are masked to honor the autoregressive property.

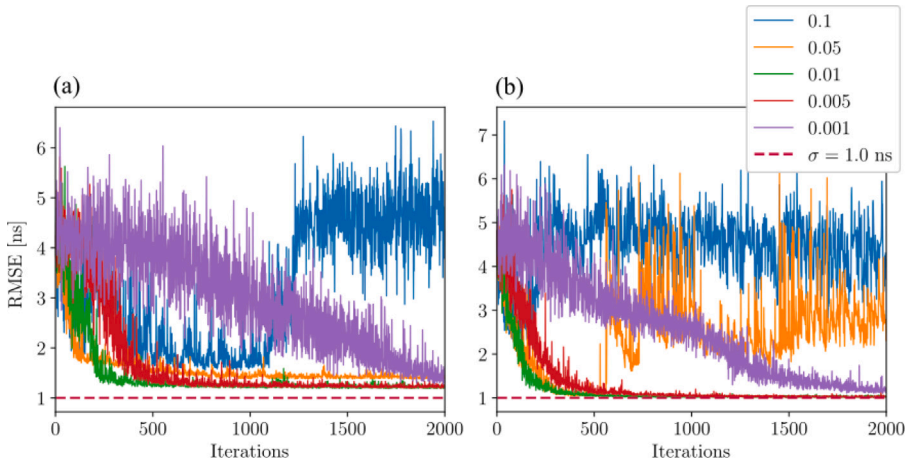

Fig. B.1. Average $RMSE_d$ value over model particles during inference as a function of the learning rate using (a) SGAN and (b) VAE as DGM.

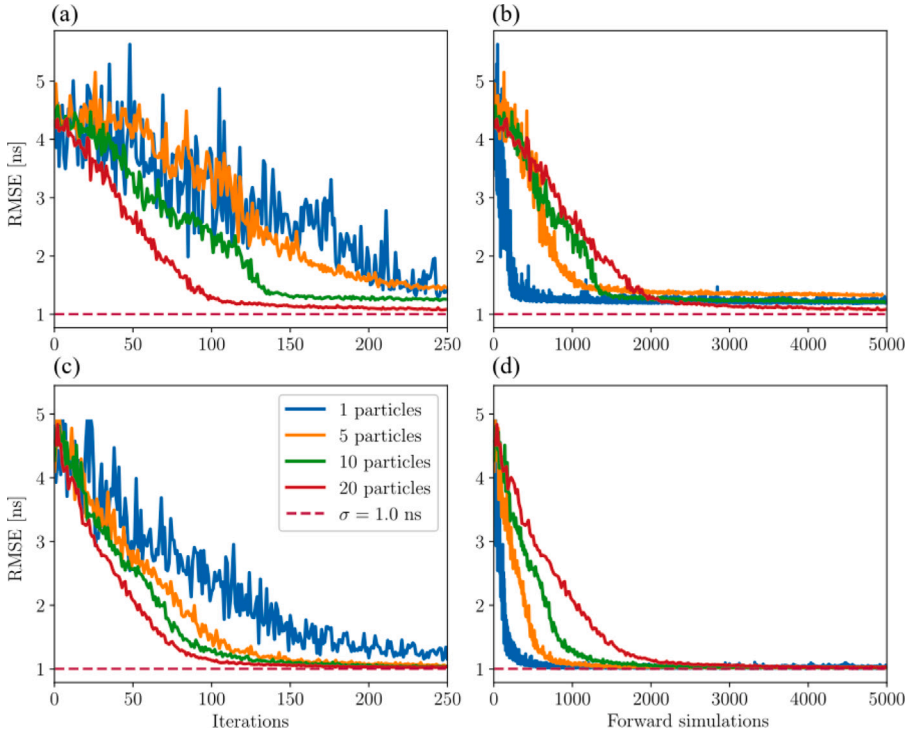

Fig. B.2. Average $RMSE_d$ values during the NT inversion as a function of the number of (a) and (c) iterations and (b) and (d) forward simulations with SGAN (a, b) and VAE (c, d) as DGM. The different curves correspond to different number of particles used to estimate the ELBO and its gradients at each NT iteration.

between layers are masked to ensure conditioning of variables only on those preceding them (see illustration in Fig. A.1b; Germain et al., 2015; Papamakarios et al., 2017). Before each flow the input is re-ordered (permutation) which has been shown to improve the training of such models (Germain et al., 2015; Kingma et al., 2016). The autoregressive network includes one hidden layer with $2n$ hidden units (neurons). To be able to represent all degrees of conditioning, the number of units in a hidden layer should be at least $n - 1$. Here we found that $2n$ units in the hidden layer performs slightly better than using n units (a default choice). We introduce a non-linearity to the transformation by applying a ReLU activation function to each flow. Results did not change significantly when other activation functions such as LeakyReLU or ELU were used and we found ReLU to work well for our purposes. Each additional flow in the current architecture introduces $6n^2 + 2$ trainable parameters therefore, the number of flows chosen was based on a consideration of complexity/performance versus computational effort that might vary for different types of models.

Appendix B. Hyperparameter calibration

Once the architecture (i.e. number of flows, number of layers etc.) of the IAF is fixed, there are two main algorithmic variables that may affect the final results: (1) number of particles N_s and (2) learning rate. To determine proper values for these variables and test the robustness of the approach to different choices, we perform a hyperparameter search and show the results on models mg_1 (for the SGAN) and mv_3 (for the VAE) in Fig. 2. We first test the NT routine with the SGAN and VAE using learning rates of: 0.1, 0.05, 0.01, 0.005, 0.001 and one particle. The curves in Fig. B.1 represent the RMSE of the data misfit $RMSE_d$ during the NT-training for the different learning rates. For both the SGAN and VAE it is found that a learning rate of 0.01 gives the fastest and most stable convergence towards the target misfit of 1 ns corresponding to the standard deviation of the noise added to the synthetic data. A higher learning rate results in either instability

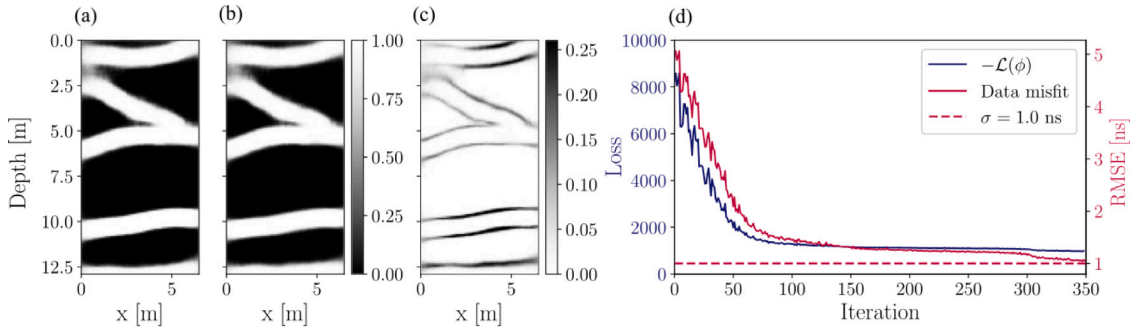

Fig. C.1. (a) True mv_1 model (b) mean posterior models obtained from NT using ten particles and (c) posterior standard deviation in the model image space. (d) The ELBO loss and $RMSE_d$ in ns during training.

or convergence to a sub-optimal solution, while a lower learning rate results in a slower convergence.

To evaluate how many particles N_s to use, we fix the learning rate at the optimal value of 0.01. We then test the NT using 1, 5, 10 and 20 particles. We compare the $RMSE_d$ averaged over the particles during NT-training and plot them as a function of the number of training (gradient-descent) iterations (Fig. B.2a and c) and the number of forward simulations (Fig. B.2b and d). Increasing the number of particles leads to more stable and earlier convergence with respect to the number of iterations. However, increasing the number of particles also induces a higher computational demand. When considering the $RMSE_d$ with respect to the number of forward simulations as in Fig. B.2d, it becomes clear that the VAE optimization performed using one particle only provides the best trade-off (at least if not considering parallelization), as the target misfit is then reached at the lowest computational cost. In contrast, the SGAN benefits from a larger number of particles as we found that using 20 particles reduces the risk of getting stuck in a local minima (Fig. B.2b) and in most cases it brings the $RMSE_d$ closer to the target misfit despite the fewer gradient-decent iterations assigned to it. This behavior is likely due to the higher degree of non-linearity of the SGAN, for which a higher number of particles provides more robustness with respect to local features in the misfit function when updating the parameters of the autoregressive network.

Appendix C. Supplementary results

The NT-inversion with the mv_1 model performed relatively poorly at the lower boundary when using one particle only (Fig. 4a–d). Indeed, the mean model (Fig. 4b) and the standard deviation (Fig. 4c) suggest that the true model is not part of the posterior. By extending the number of particles to ten, we find that the posterior mean is much closer to the true model (compare Fig. C.1a–b) and the standard deviation is higher implying a better exploration of the posterior. The lower region of high standard deviation maps well to the interface between the lower channel and the background matrix. This suggests that adding more particles can also be beneficial when using VAE as DGM.

References

Ba, J., Erdogdu, M.A., Ghassemi, M., Suzuki, T., Sun, S., Wu, D., Zhang, T., 2019. Towards characterizing the high-dimensional bias of kernel-based particle inference algorithms. In: 2nd Symposium on Advances in Approximate Bayesian Inference. Vancouver, Canada.

Bao, J., Li, L., Davis, A., 2022. Variational autoencoder or generative adversarial networks? A comparison of two deep learning methods for flow and transport data assimilation. Math. Geosci. 1–26.

Bingham, E., Chen, J.P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P.A., Horsfall, P., Goodman, N.D., 2019. Pyro: Deep universal probabilistic programming. J. Mach. Learn. Res. 20, 28:1–28:6.

Blei, D.M., Kucukelbir, A., McAuliffe, J.D., 2017. Variational inference: A review for statisticians. J. Amer. Statist. Assoc. 112 (518), 859–877.

Chan, S., Elsheikh, A.H., 2019. Parametric generation of conditional geological realizations using generative neural networks. Comput. Geosci. 23 (5), 925–952.

Dejtrakulwong, P., Mukerji, T., Mavko, G., 2012. Using kernel principal component analysis to interpret seismic signatures of thin shaly-sand reservoirs. In: SEG Technical Program Expanded Abstracts 2012. Society of Exploration Geophysicists, pp. 1–5.

Dinh, L., Krueger, D., Bengio, Y., 2014. Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516.

Dinh, L., Sohl-Dickstein, J., Bengio, S., 2016. Density estimation using real nvp. arXiv preprint arXiv:1605.08803.

Duane, S., Kennedy, A.D., Pendleton, B.J., Roweth, D., 1987. Hybrid Monte Carlo. Phys. Lett. B 195 (2), 216–222.

Dupont, E., Zhang, T., Tilke, P., Liang, L., Bailey, W., 2018. Generating realistic geology conditioned on physical measurements with generative adversarial networks. arXiv preprint arXiv:1802.03065.

Gelman, A., Rubin, D.B., 1992. Inference from iterative simulation using multiple sequences. Statist. Sci. 7 (4), 457–472.

Germain, M., Gregor, K., Murray, I., Larochelle, H., 2015. Made: Masked autoencoder for distribution estimation. In: International Conference on Machine Learning. PMLR, pp. 881–889.

Good, I.J., 1952. Rational decisions. J. R. Stat. Soc. Ser. B Stat. Methodol. 14 (1), 107–114.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press, http://www.deeplearningbook.org.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. Adv. Neural Inf. Process. Syst. 27.

Hansen, T.M., Cordua, K.S., Mosegaard, K., 2012. Inverse problems with non-trivial priors: efficient solution through sequential Gibbs sampling. Comput. Geosci. 16 (3), 593–611.

Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.

Hoffman, M.D., Blei, D.M., Wang, C., Paisley, J., 2013. Stochastic variational inference. J. Mach. Learn. Res..

Hoffman, M., Sountsov, P., Dillon, J.V., Langmore, I., Tran, D., Vasudevan, S., 2019. Neutra-lizing bad geometry in Hamiltonian Monte Carlo using neural transport. arXiv preprint arXiv:1903.03704.

Hou, X., Shen, L., Sun, K., Qiu, G., 2017. Deep feature consistent variational autoencoder. In: 2017 IEEE Winter Conference on Applications of Computer Vision. WACV, pp. 1133–1141.

Jetchev, N., Bergmann, U., Vollgraf, R., 2016. Texture synthesis with spatial generative adversarial networks. arXiv preprint arXiv:1611.08207.

Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K., 1999. An introduction to variational methods for graphical models. Mach. Learn. 37 (2), 183–233.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kingma, D.P., Dhariwal, P., 2018. Glow: Generative flow with invertible 1x1 convolutions. Adv. Neural Inf. Process. Syst. 31.

Kingma, D.P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., Welling, M., 2016. Improved variational inference with inverse autoregressive flow. Adv. Neural Inf. Process. Syst. 29, 4743–4751.

Kingma, D.P., Welling, M., 2014. Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114.

Kobyzev, I., Prince, S.J., Brubaker, M.A., 2021. Normalizing flows: An introduction and review of current methods. IEEE Trans. Pattern Anal. Mach. Intell. 43 (11), 3964–3979.

Konaté, A.A., Pan, H., Ma, H., Cao, X., Yevenyo Ziggah, Y., Oloo, M., Khan, N., 2015. Application of dimensionality reduction technique to improve geophysical log data classification performance in crystalline rocks. J. Pet. Sci. Eng. 133, 633–645.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., Blei, D.M., 2017. Automatic differentiation variational inference. J. Mach. Learn. Res. 18 (1), 430–474.

Kullback, S., Leibler, R.A., 1951. On information and sufficiency. Ann. Math. Stat. 22 (1), 79–86.
Laloy, E., Hérault, R., Jacques, D., Linde, N., 2018. Training-image based geostatistical inversion using a spatial generative adversarial neural network. Water Resour. Res. 54 (1), 381–406.
Laloy, E., Hérault, R., Lee, J., Jacques, D., Linde, N., 2017. Inversion using a new low-dimensional representation of complex binary geological media based on a deep neural network. Adv. Water Resour. 110, 387–405.
Laloy, E., Linde, N., Jacques, D., 2021. Approaching geoscientific inverse problems with vector-to-image domain transfer networks. Adv. Water Resour. 152, 103917.
Laloy, E., Linde, N., Ruffino, C., Hérault, R., Gasso, G., Jacques, D., 2019. Gradient-based deterministic inversion of geophysical data with generative adversarial networks: Is it feasible? Comput. Geosci. 133, 104333.
Laloy, E., Vrugt, J.A., 2012. High-dimensional posterior exploration of hydrologic models using multiple-try DREAM$_{(ZS)}$ and high-performance computing. Water Resour. Res. 48 (1).
Levy, S., Hunziker, J., Laloy, E., Irving, J., Linde, N., 2022. Using deep generative neural networks to account for model errors in Markov chain Monte Carlo inversion. Geophys. J. Int. 228 (2), 1098–1118.
Liu, Q., Wang, D., 2016. Stein variational gradient descent: a general purpose Bayesian inference algorithm. Adv. Neural Inf. Process. Syst. 29, 2378–2386.
Lopez-Alvis, J., Laloy, E., Nguyen, F., Hermans, T., 2021. Deep generative models in inversion: The impact of the generator's nonlinearity and development of a new approach based on a variational autoencoder. Comput. Geosci. 152, 104762.
Mariethoz, G., Renard, P., Caers, J., 2010. Bayesian inverse problem and optimization with iterative spatial resampling. Water Resour. Res. 46 (11).
Mosser, L., Dubrule, O., Blunt, M.J., 2018. Conditioning of generative adversarial networks for pore and reservoir scale models. In: 80th EAGE Conference and Exhibition 2018, Vol. 2018. European Association of Geoscientists & Engineers, pp. 1–5.
Mosser, L., Dubrule, O., Blunt, M.J., 2020. Stochastic seismic waveform inversion using generative adversarial networks as a geological prior. Math. Geosci. 52 (1), 53–79.
Neal, R.M., 2011. MCMC using Hamiltonian dynamics. Handb. Markov Chain Monte Carlo 2 (11), 113–162.
Nijkamp, E., Gao, R., Sountsov, P., Vasudevan, S., Pang, B., Zhu, S.-C., Wu, Y.N., 2020. Learning energy-based model with flow-based backbone by neural transport MCMC. arXiv preprint arXiv:2006.06897.
Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B., 2021. Normalizing flows for probabilistic modeling and inference. J. Mach. Learn. Res. 22 (57), 1–64.
Papamakarios, G., Pavlakou, T., Murray, I., 2017. Masked autoregressive flow for density estimation. Adv. Neural Inf. Process. Syst. 30.
Ramgraber, M., Weatherl, R., Blumensaat, F., Schirmer, M., 2021. Non-Gaussian parameter inference for hydrogeological models using stein variational gradient descent. Water Resour. Res. 57 (4), e2020WR029339.
Rezende, D., Mohamed, S., 2015. Variational inference with normalizing flows. In: International Conference on Machine Learning. PMLR, pp. 1530–1538.
Richardson, A., 2018. Generative adversarial networks for model order reduction in seismic full-waveform inversion. arXiv preprint arXiv:1806.00828.
Robert, C.P., Casella, G., Casella, G., 1999. Monte Carlo Statistical Methods, Vol. 2. Springer.
Rücker, C., Günther, T., Wagner, F.M., 2017. pyGIMLi: An open-source library for modelling and inversion in geophysics. Comput. Geosci. 109, 106–123.
Song, S., Mukerji, T., Hou, J., 2021a. Bridging the gap between geophysics and geology with generative adversarial networks. IEEE Trans. Geosci. Remote Sens. 60, 1–11.
Song, S., Mukerji, T., Hou, J., 2021b. GANSim: Conditional facies simulation using an improved progressive growing of generative adversarial networks (GANs). Math. Geosci. 53 (7), 1413–1444.
Tahmasebi, P., 2018. Multiple point statistics: a review. In: Handbook of Mathematical Geosciences. Springer, Cham, pp. 613–643.
ter Braak, C.J., Vrugt, J.A., 2008. Differential evolution Markov Chain with snooker updater and fewer chains. Stat. Comput. 18 (4), 435–446.
Urozayev, D., Ait-El-Fquih, B., Hoteit, I., Peter, D., 2021. A reduced-order variational Bayesian approach for efficient subsurface imaging. Geophys. J. Int..
Vrugt, J.A., Ter Braak, C.J., Gupta, H.V., Robinson, B.A., 2009. Equifinality of formal (DREAM) and informal (GLUE) Bayesian approaches in hydrologic modeling? Stoch. Environ. Res. Risk Assess. 23 (7), 1011–1026.
Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P., 2004. Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. 13 (4), 600–612.
Wold, S., Esbensen, K., Geladi, P., 1987. Principal component analysis. Chemometr. Intell. Lab. Syst. 2 (1–3), 37–52.
Zahner, T., Lochbühler, T., Mariethoz, G., Linde, N., 2016. Image synthesis with graph cuts: a fast model proposal mechanism in probabilistic inversion. Geophys. J. Int. 204 (2), 1179–1190.
Zhang, X., Curtis, A., 2020a. Seismic tomography using variational inference methods. J. Geophys. Res.: Solid Earth 125 (4), e2019JB018589.
Zhang, X., Curtis, A., 2020b. Variational full-waveform inversion. Geophys. J. Int. 222 (1), 406–411.
Zhang, X., Curtis, A., 2021. Bayesian full-waveform inversion with realistic priors. Geophysics 86 (5), A45–A49.
Zhao, X., Curtis, A., Zhang, X., 2021. Bayesian seismic tomography using normalizing flows. Geophys. J. Int. 228 (1), 213–239.