Computers & Geosciences 152 (2021) 104762

Contents lists available at ScienceDirect

Computers and Geosciences

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

Deep generative models in inversion: The impact of the generator's nonlinearity and development of a new approach based on a variational autoencoder☆

Jorge Lopez-Alvis [a,b,*], Eric Laloy [c], Frédéric Nguyen [a], Thomas Hermans [b]

[a] Urban and Environmental Engineering, Applied Geophysics, University of Liège, Belgium
[b] Department of Geology, Ghent University, Belgium
[c] Engineered and Geosystems Analysis, Institute for Environment, Health and Safety, Belgian Nuclear Research Center, Belgium

ARTICLE INFO

Keywords:
Deep generative model
Geophysical inversion
Geological prior information
Variational autoencoder
Stochastic gradient descent
Deep learning

ABSTRACT

When solving inverse problems in geophysical imaging, deep generative models (DGMs) may be used to enforce the solution to display highly structured spatial patterns which are supported by independent information (e.g. the geological setting) of the subsurface. In such case, inversion may be formulated in a latent space where a low-dimensional parameterization of the patterns is defined and where Markov chain Monte Carlo or gradient-based methods may be applied. However, the generative mapping between the latent and the original (pixel) representations is usually highly nonlinear which may cause some difficulties for inversion, especially for gradient-based methods. In this contribution we review the conceptual framework of inversion with DGMs and propose that this nonlinearity is caused mainly by changes in topology and curvature induced by the generative function. As a result, we identify a conflict between two goals: the accuracy of the generated patterns and the feasibility of gradient-based inversion. In addition, we show how some of the training parameters of a variational autoencoder, which is a particular instance of a DGM, may be chosen so that a tradeoff between these two goals is achieved and acceptable inversion results are obtained with a stochastic gradient-descent scheme. A series of test cases using synthetic models with channel patterns of different complexity and cross-borehole traveltime tomographic data involving both a linear and a nonlinear forward operator show that the proposed method provides useful results and performs better compared to previous approaches using DGMs with gradient-based inversion.

1. Introduction

A common task in the geosciences is to solve an inverse problem in order to obtain a model (or image) from a set of measurements sensing a heterogeneous spatial domain. When characterizing subsurface environments, the corresponding inverse problem is usually ill-posed yielding non-unique and potentially unstable solutions. This is mainly because the measurements do not provide sufficiently independent information on the distribution of subsurface properties. In such cases it is possible to constrain the solution to allow only certain spatial patterns. In practice, such patterns may be supported by independent (prior) information of the sensed domain (e.g. knowledge of the geological setting) and used with the aim of appropriately reconstructing heterogeneity. Classical regularization may be used to impose the model to be smooth or of minimum magnitude (Tikhonov and Arsenin, 1977) but in many cases this does not yield satisfactory results in areas poorly constrained by the data (Hermans et al., 2012; Caterina et al., 2014). Recently, the use of deep generative models (DGMs) to constrain the solution space of inverse problems has been proposed so that resulting models have specific spatial patterns (Bora et al., 2017; Laloy et al., 2017; Hand and Voroninski, 2018; Seo et al., 2019). DGMs can deal with realistic (natural) patterns which are not captured by classical

☆ Jorge Lopez-Alvis conceived the main idea of the paper, developed and tested the proposed method (including its software implementation) and wrote a first draft of the manuscript. Eric Laloy conceived the main idea of the paper, performed some software tests, suggested improvements to the proposed method and edited the first draft. Frédéric Nguyen and Thomas Hermans are supervisors of the first author, conceived the main idea of the paper, suggested improvements to the proposed method, obtained funding for the research and edited the first draft.

* Corresponding author. Urban and Environmental Engineering, Applied Geophysics, University of Liège, Belgium.
E-mail address: jlopez@uliege.be (J. Lopez-Alvis).

https://doi.org/10.1016/j.cageo.2021.104762
Received 4 August 2020; Received in revised form 11 March 2021; Accepted 20 March 2021
Available online 1 April 2021

regularization or random processes defined by second-order statistics (Linde et al., 2015). In this way, inversion with DGMs provides an alternative to inversion with either multiple-point geostatistics (MPS) (Caers and Hoffman, 2006; González et al., 2008; Hansen et al., 2012; Linde et al., 2015; Rezaee and Marcotte, 2018) or example-based texture synthesis (ETS) (Zahner et al., 2016). While other methods exist that are also able to produce realistic models with inversion e.g. using plurigaussian fields (Armstrong et al., 2011; Liu and Oliver, 2005), they are usually not as flexible as DGMs, MPS or ETS in terms of the patterns they can generate.

All the previously mentioned methods generally rely on gridded representations for the models (i.e. by dividing the spatial domain in cells or pixels). They all require a large number of training examples of the desired patterns to work, which are usually provided as a large training image (or exemplar). However, the procedure for generating a model with MPS or ETS differs from that of DGMs. Both MPS and ETS build the models sequentially (i.e. pixel by pixel or patch by patch) either by directly sampling from the training image (Mariethoz et al., 2010) or by sampling from an empirical probability distribution that was previously obtained from the training image (Strebelle, 2002). In contrast, DGMs rely on a generative function and a low-dimensional reparameterization that follows a known probability distribution. The DGM is first trained with many examples of the desired patterns (e.g. many croppings of the training image) to obtain the generative function. A model is then generated by taking one sample from the low-dimensional probability distribution and passing it through the generative function. This low-dimensional reparameterization is often referred to as latent vector and the space where it is represented is called the latent space. Note that finding a low-dimensional representation is generally feasible for highly structured spatial patterns. The usual geometric argument for this statement is as follows: any gridded model may be represented as a vector in "pixel" space (a space where each pixel is one dimension) and when the models are restricted to those with certain spatial patterns, their vectors will take up only a subset of this pixel space. This subset usually defines a manifold of lower dimensionality than the pixel space (Fefferman et al., 2016) and the latent space is simply a low-dimensional space where such manifold is represented.

Most inversion methods require a perturbation step to search for models that fit the data but such a step is not straightforward to compute for highly structured patterns (Linde et al., 2015; Hansen et al., 2012). The latent space of DGMs provides a useful frame to compute a perturbation step (Laloy et al., 2017) or even a local gradient-descent direction (Laloy et al., 2019) which generally results in better exploration of the posterior distribution and/or faster convergence compared to inversion with MPS or ETS. So far, inversion with DGMs has been done successfully with regular MCMC sampling methods (Laloy et al., 2017, 2018). However, when applicable, gradient-based methods may be preferred given their lower computational demand. Gradient-based deterministic inversion with DGMs has been pursued with encouraging results (Richardson, 2018; Laloy et al., 2019), however, convergence to the true model was shown to be dependent on the initial model. In the framework of probabilistic inversion, MCMC methods that use the gradient to guide the sampling in the latent space have shown to be less prone to get trapped in local minima than gradient-based deterministic methods while they are also expected to reach convergence faster than regular MCMC (Mosser et al., 2018). A different inversion strategy that has also been applied successfully with DGMs and has a relatively low computational cost is the Ensemble Smoother (Canchumuni et al., 2019; Mo et al., 2020).

Recently, Laloy et al. (2019) studied the difficulties of performing gradient-based deterministic inversion with a specific DGM. They concluded that the nonlinearity of their generative function or "generator" (i.e. the mapping from the latent space to the pixel space) was high enough to hinder gradient-based optimization, causing the latter to often fail in finding the global minimum even when the objective function was known to be convex (in pixel space). In order to approximate manifolds of realistic patterns, most common DGMs involve (artificial) neural networks with several layers and nonlinear (activation) functions. For a specific subsurface pattern, the degree of nonlinearity of the generative function may be controlled mainly by its architecture and the way it is trained (Goodfellow et al., 2016). Regarding difference in training, two common types of DGMs can be distinguished: generative adversarial networks (GANs) (Goodfellow et al., 2014) and variational autoencoders (VAEs) (Kingma and Welling, 2014). VAEs training relies on a variational inference strategy where a DNN is used to approximate the required variational distribution. Such distribution is equivalent to a probabilistic encoder (see details in Section 2.3). GANs training is based on adversarial learning: the generator is trained together with a discriminator in such a way that the models generated by the former are aimed to fool the latter. In both cases training generally takes the form of optimizing a loss function, but in the case of GANs one has to alternate between optimizing the generator and the discriminator. GANs and VAEs require specification of a probability distribution in the latent space and an architecture for the discriminator or encoder (respectively) in addition to the one for their generators. They might also require other parameters to be specified such as the weights on the different terms of the loss function. Frequently, some of these choices use default values, but generally all of them may affect the degree of nonlinearity of the generator (Rolinek et al., 2019).

Given all the possible choices to train the generator it is interesting to investigate whether one can find those that allow both for a good reproduction of the patterns and a good performance of less computationally demanding gradient-based inversion. In this work, we review some of the difficulties of performing inversion with DGMs and show how to obtain a well-balanced tradeoff between accuracy in patterns and applicability of gradient-based methods. In particular, we propose to use the training choice of a VAE as DGM and to select some of its parameters in order to achieve good results with gradient-based inversion. Then, we compare this to the training choice of a GAN that has been tested with gradient-based inversion in prior studies (Laloy et al., 2019; Richardson, 2018). Furthermore, we show that since the resulting VAE inversion is only mildly nonlinear, modified stochastic gradient-descent (SGD) methods are generally sufficient to avoid getting trapped in local minima and provide a better alternative than regular gradient-based methods while also retaining a low computational cost.

The remainder of this paper is structured as follows. Section 2.1 explains DGMs and their conceptualization as approximating the real (pattern) manifold. In Section 2.2 the use of DGMs to represent prior information in inversion and the difficulties of performing gradient-based inversion are reviewed. Sections 2.3 and 2.4 show how to use a VAE and SGD to cope with some of the mentioned difficulties. Then, Section 3 shows some results of the proposed approach. Section 4 discusses the obtained results and points to some remaining challenges. Finally, Section 5 presents the conclusions of this work.

2. Methods

2.1. Deep generative models (DGM) to represent realistic patterns

The term "deep learning" generally refers to machine learning methods that involve several layers of multidimensional functions. This general "deep" setting has been shown to allow for complex mappings to be accurately approximated by building a succession of intermediate (simpler) representations or concepts (Goodfellow et al., 2016). Consider, for instance, deep neural networks (DNNs) which are mappings defined by a composition of a set of (multidimensional) functions φ_k as:

$$\mathbf{g}(\boldsymbol{\tau}) = (\varphi_L \circ \ldots \circ \varphi_2 \circ \varphi_1)(\boldsymbol{\tau}) \tag{1}$$

where $\boldsymbol{\tau}$ is a multidimensional (vector) input, $k = \{1, \ldots, L\}$ denotes the function (layer) index and composition follows the order from right to

left. Furthermore, each φ_k is defined as:

$$\varphi_k(\boldsymbol{\xi}) = \psi_k(\mathbf{M}_k\boldsymbol{\xi} + \mathbf{b}_k) \tag{2}$$

in which ψ_k is a (nonlinear) activation function, $\mathbf{M}_k$ is a matrix of weights, $\mathbf{b}_k$ is a vector of biases and $\boldsymbol{\xi}$ denotes the output of the previous function (layer) φ_{k-1} for $k > 1$ or the initial input $\boldsymbol{\tau}$ for $k = 1$. Then, training the DNN involves estimating the values for all the parameters $\boldsymbol{\theta} = \{\mathbf{M}_k, \mathbf{b}_k | 1 \leq k \leq L\}$ where each $\mathbf{M}_k$ or $\mathbf{b}_k$ may be of different dimensionality depending on the layer. In practice, the number of parameters $\boldsymbol{\theta}$ for such models may reach the order of 10^6, therefore training is achieved by relying on autodifferentiation (see e.g. Paszke et al., 2017) and fast optimization techniques based on SGD (see e.g. Kingma and Ba, 2017), both usually implemented for and run in highly parallel (GPU) computing architectures.

A deep generative model (DGM) is a particular application of such deep methods (Salakhutdinov, 2015). In a DGM a set of training examples $\mathbf{X} = \{\mathbf{x}^{(i)} | 1 \leq i \leq N\}$ and a simple low-dimensional probability distribution $p(\mathbf{z})$ are used to learn a model $\mathbf{g}(\mathbf{z})$ that is capable of generating new samples of $\mathbf{x}$ (which are consistent with the training set) by using as input samples from $p(\mathbf{z})$. This can be written as:

$$\mathbf{x} = \mathbf{g}(\mathbf{z}), \ \mathbf{z} \sim p(\mathbf{z}) \tag{3}$$

where $\mathbf{g}(\mathbf{z})$ is referred to as the "generator" and $\mathbf{z}$ denotes a vector of latent variables or "code". While the training (and generated) samples $\mathbf{x}$ are usually represented in a high-dimensional space $\mathbb{R}^D$, the probability distribution $p(\mathbf{z})$ is defined in a low-dimensional space $\mathbb{R}^d$. The space $\mathbb{R}^D$ is often referred to as "ambient space" while the space $\mathbb{R}^d$ is called the "latent space". Fig. 1 shows a schematic representation of the general setting of DGMs with inversion where (a) and (c) show an ambient space with $D = 3$ and a latent space with $d = 2$. A typical application of DGMs is the generation of images (see e.g. Kingma and Welling, 2014; Goodfellow et al., 2014) for which the ambient space is just the pixel space. Gridded representations of subsurface models may be seen as two- or three-dimensional images of the subsurface.

The underlying assumption in DGMs is that real-world data are generally structured in their high-dimensional ambient space $\mathbb{R}^D$ and therefore have an intrinsic lower dimensionality—such assumption is known in machine learning literature as the manifold hypothesis (Fefferman et al., 2016) because it states that high-dimensional data usually lie on (or lie close to) a lower-dimensional manifold $\mathcal{M} \subset \mathbb{R}^D$. For instance, when studying a subsurface region it is usually assumed that geological processes gave it certain degree of structure then, to allow for a flexible base on which to represent the distribution of the different subsurface materials, the region is usually divided in homogeneous pixels (or cells). Such gridded representation "lives" in the high-dimensional pixel space (the ambient space) but since it has some structure there should be a lower dimensional space (the latent space) where the same distribution of subsurface materials might be represented. Technically, while both the latent space $\mathbb{R}^d$ and the manifold $\mathcal{M}$ are usually low-dimensional, they may differ in dimensionality and/or the manifold may only occupy a certain portion of the latent space (e.g. the shaded region in Fig. 1c). Manifolds are geometrical objects that have a topology and a curvature. A topology is the structure of a geometrical object that is preserved under continuous deformations (e.g. stretching or bending). In other words, when a non-continuous operation such as gluing or tearing occurs the topology of the object changes. These changes may be described in terms of different topological properties such as compactness, connectedness and simple-connectedness. In this work, the concept of curvature is used to state that in general one starts with a "flat" domain in the latent space and then one has to curve it to fit the real manifold. In this way, the concept helps to understand where part of the nonlinearity of the generative function comes from. While formal definitions of curvature exist (e.g. Riemannian curvature as applied to smooth manifolds) they are not used in this work.

Considering the manifold assumption described above, a DGM may be regarded as a model to implicitly approximate the "real" manifold $\mathcal{M}$

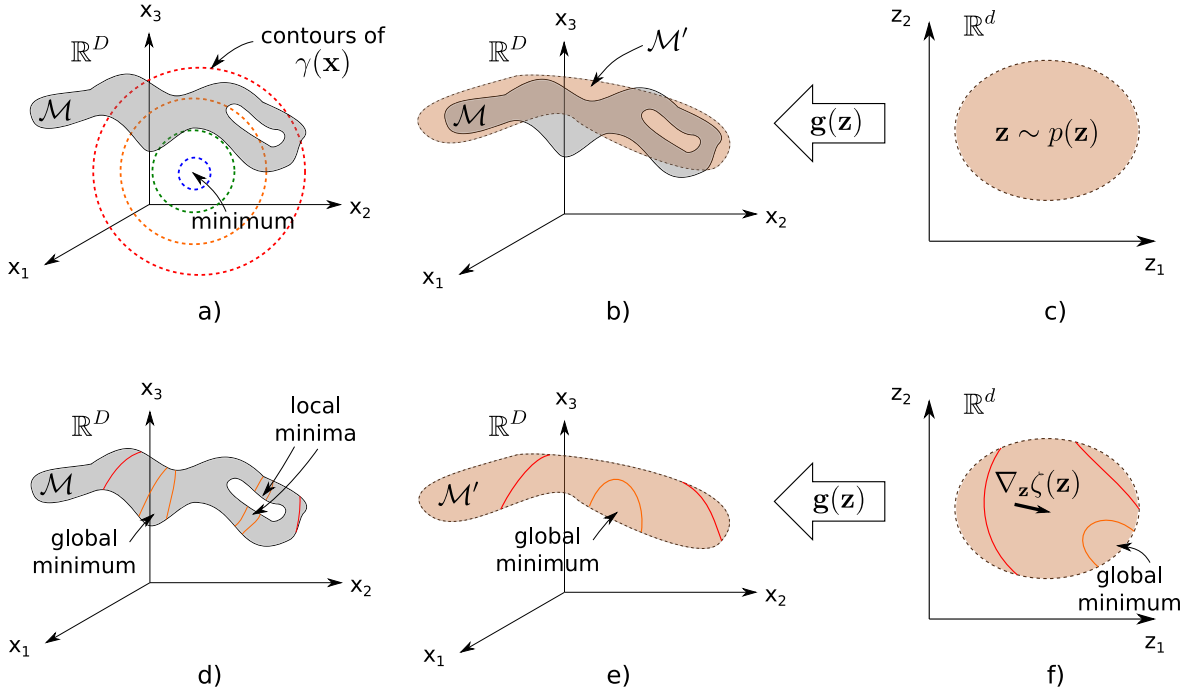

Fig. 1. Sketch of the different parts involved in DGMs with inversion: approximation of the real manifold (a–c) and the impact of the approximated manifold in inversion (d–f). (a) Real manifold $\mathcal{M}$ and inversion's misfit function $\gamma(\mathbf{x})$ in ambient space $\mathbb{R}^D$. (b) Approximate manifold $\mathcal{M}'$ overlaying the real manifold. (c) Region of latent space $\mathbb{R}^d$ where the approximate manifold is implicitly defined by the probability distribution $p(\mathbf{z})$. (d) Misfit function contours intersected by the real manifold. (e) Misfit function contours intersected by approximate manifold. (f) Misfit function contours back-mapped onto the latent space and the related gradient $\nabla_{\mathbf{z}}\zeta(\mathbf{z})$ computed at one iteration.

by generating samples that closely follow such manifold, i.e. that lie on an approximate manifold $\mathscr{M}'$ (Fig. 1b). Samples of this approximate manifold are generated by sampling first from a simple probability distribution $p(\mathbf{z})$ in latent space (e.g. a normal or uniform distribution) and then passing them through the generator $\mathbf{g}(\mathbf{z})$. Since the probability distribution $p(\mathbf{z})$ defines indirectly a region (or subset) in latent space that generally has a different curvature and topology than the real manifold, the generator $\mathbf{g}(\mathbf{z})$ must be able to approximate both curvature and topology when mapping the samples of $p(\mathbf{z})$ to ambient space. This generally requires the generator to be a highly nonlinear function. As an instance, consider the case of certain spatial patterns whose real manifold is a highly curved surface with "holes" in ambient space and the (input) region defined by a uniform $p(\mathbf{z})$ is a (flat) plane in a two-dimensional latent space. Regarding their topological properties, one technically says that this plane is simply connected while the real manifold is not (see e.g. Kim and Zhang, 2019). Then, the generative function has to deform this plane in such a way as to approximate (or cover) the real manifold as close as possible. An important property of DGMs is that since a probability distribution in latent space is used, the sample "density" of such plane (and its mapping) also plays an important role. For instance, the generative function may approximate the "holes" of the real manifold by creating regions of very low density of samples when mapping to ambient space (to picture this one can imagine locally stretching a flexible material without changing its curvature). The combined deformation needed to curve the plane and to "make" the holes causes the generative function to be highly nonlinear. Note that when considering a DGM that uses a DNN with rectified linear unit (ReLU) activation functions as generator $\mathbf{g}(\mathbf{z})$, it is also possible for $\mathbf{g}(\mathbf{z})$ to change topology of the input by "folding" transformations (Naitzat et al., 2020).

While one should always strive to accurately approximate the real manifold, since a finite set of training samples is used a tradeoff between accuracy and diversity in the generated samples may be a better objective. Indeed, the use of the prescribed probability distributions is done to continuously "fill" the space between the samples and therefore generate samples of a continuous manifold. Recent success—in terms of accuracy and diversity of generated samples—has been achieved with two DGMs that are based on deep neural networks (DNNs): generative adversarial networks (GANs) (Goodfellow et al., 2014) and variational autoencoders (VAEs) (Kingma and Welling, 2014). The generator $g(\mathbf{z})$ on both strategies is a mapping from low-dimensional input $\mathbf{z} \in \mathbb{R}^d$ to high-dimensional output $\mathbf{x} \in \mathbb{R}^D$. In contrast, the mappings corresponding to the discriminator and the encoder take high-dimensional inputs $\mathbf{x}$ and return low-dimensional outputs.

2.2. Gradient-based inversion with DGMs

Consider a survey or experiment for which a vector of noisy measurements $\mathbf{y} = (y_1, \ldots, y_Q)^T \in \mathbb{R}^Q$ of a physical process is available. A simplified description of the process may be expressed by a (mathematical) forward operator $\mathbf{f} : \mathbb{R}^D \rightarrow \mathbb{R}^Q$ that takes as input a subsurface model vector $\mathbf{x} = (x_1, \ldots, x_D)^T \in \mathbb{R}^D$ (obtained by discretizing the spatial distribution of physical properties) and outputs a simulated response $\mathbf{f}(\mathbf{x})$. Commonly, this operator is in the form of a (numerical) discretization of a set of partial differential equations describing the process under study and is an approximation of the real process. As a result of such approximation and the use of noisy data, an error term $\boldsymbol{\eta}$ is added to the simulation to represent total uncertainty. Then, the relation between the operator and the measurements may be written as (see e.g. Aster et al., 2013):

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\eta} \tag{4}$$

The corresponding inverse problem or inversion of Eq. (4), aims to obtain an estimation of the vector $\mathbf{x}$ from the (noisy) data $\mathbf{y}$. Deterministic inversion does so by optimizing a misfit function $\gamma(\mathbf{x})$ that is usually given in the form of a distance function between simulated response $\mathbf{f}(\mathbf{x})$ and data $\mathbf{y}$:

$$\gamma(\mathbf{x}) = ||\mathbf{f}(\mathbf{x}) - \mathbf{y}||^2 \tag{5}$$

where $\| \cdot \|$ denotes the l_2 norm. In this way, traditional gradient-based inversion requires the gradient $\nabla_{\mathbf{x}}\gamma(\mathbf{x})$ whose elements are:

$$[\nabla_{\mathbf{x}}\gamma(\mathbf{x})]_i = \frac{\partial \gamma(\mathbf{x})}{\partial x_i} \tag{6}$$

and are computed by considering Eq. (4) together with the chosen misfit.

DGMs may be used with inversion of subsurface data $\mathbf{y}$ to obtain geologically realistic spatial distributions of physical properties $\mathbf{x}$ (Laloy et al., 2017). While this is also possible with traditional deterministic inversion where a regularization term is added directly in Eq. (5) (i.e. in ambient space) to obtain models with the imposed structures that minimize the misfit (Lange et al., 2012; Caterina et al., 2014), DGMs are more flexible because they can simultaneously enforce different kind of patterns provided they are trained with samples of all such patterns (Bergmann et al., 2017). In the DGM setting, the low-dimensional samples $\mathbf{z}$ that input to the generator $\mathbf{g}(\mathbf{z})$ may be seen as defining a low-dimensional parameterization (or encoding) of realistic patterns $\mathbf{x}$ and therefore exploration of the set of feasible models may be done in the latent space $\mathbb{R}^d$, as long as the search is done within the region where the approximated manifold $\mathscr{M}'$ is defined (depicted by shading in Fig. 1c).

Since the misfit $\gamma(\mathbf{x})$ is typically defined in ambient space $\mathbb{R}^D$ (e.g. in Fig. 1a), gradient-based inversion with DGMs may be seen as optimizing the intersection of $\gamma(\mathbf{x})$ with the approximate manifold $\mathscr{M}'$ (Fig. 1e). Such intersected misfit is mapped into the latent space (Fig. 1f) and may be expressed as $\gamma(\mathbf{g}(\mathbf{z}))$. Also note that when probability distributions $p(\mathbf{z})$ with infinite support are used (e.g. a normal distribution), one can guide the search in the latent space by adding controlling (regularization) terms to the mapped misfit (see e.g. Bora et al., 2017) and the resulting objective function may be written as:

$$\begin{aligned}\zeta(\mathbf{z}) &= \gamma(\mathbf{g}(\mathbf{z})) + \lambda R(\mathbf{z}) \\ &= ||\mathbf{f}(\mathbf{g}(\mathbf{z})) - \mathbf{y}||^2 + \lambda R(\mathbf{z})\end{aligned} \tag{7}$$

where $R(z)$ is a regularization term defined in the latent space and λ is the corresponding regularization factor. The goal of the regularization term is to make the search consistent with the selected probability distribution, i.e. optimization stays preferentially within the high-density regions in the latent space.

In practice, no exhaustive mapping has to be done and the gradient $\nabla_{\mathbf{z}}\zeta(\mathbf{z})$ is only computed for the points in latent space where optimization lands in each iteration (in Fig. 1f the gradient is represented for one iteration). The gradient $\nabla_{\mathbf{z}}\zeta(\mathbf{z})$ is computed by adding a derivative layer corresponding to $\nabla_{\mathbf{x}}\gamma(\mathbf{x})$ to the autodifferentation that was set up for $\mathbf{g}(\mathbf{z})$ while training the DGM (see e.g. Laloy et al., 2019). Such autodifferentiation setup may be seen as implicitly obtaining the Jacobian $\mathbf{J}(\mathbf{z})$ of size $D \times d$ whose elements are:

$$[\mathbf{J}(\mathbf{z})]_{i,j} = \frac{\partial g_i(\mathbf{z})}{\partial z_j} \tag{8}$$

Then, the gradient $\nabla_{\mathbf{z}}\zeta(\mathbf{z})$ is obtained from Eq. (7) by using the chain rule given by the product of Eqs. (6) and (8):

$$\begin{aligned}\nabla_{\mathbf{z}}\zeta(\mathbf{z}) &= \nabla_{\mathbf{z}}\gamma(\mathbf{g}(\mathbf{z})) + \lambda\nabla_{\mathbf{z}}R(\mathbf{z}) \\ &= \mathbf{J}(\mathbf{z})^T\nabla_{\mathbf{x}}\gamma(\mathbf{x}) + \lambda\nabla_{\mathbf{z}}R(\mathbf{z})\end{aligned} \tag{9}$$

The latter may also be done implicitly by incorporating directly in the autodifferentiation framework, e.g. putting it on top of the so called computational graph (Richardson, 2018; Mosser et al., 2018).

Even when the considered misfit function $\gamma(\mathbf{x})$ is convex in ambient

space $\mathbb{R}^D$ (as depicted by concentric contours in Fig. 1a), difficulties to perform gradient-based deterministic inversion may arise due to the generator $\mathbf{g}(\mathbf{z})$ (Laloy et al., 2019). We propose that such difficulties arise because the generator (1) is highly nonlinear and (2) changes the topology of the input region defined by $p(\mathbf{z})$. Both of these properties often cause distances (between samples) in latent space to be significantly different than distances in ambient space. Consider again the example of a real manifold that is a highly curved surface with "holes" in it and a uniform distribution $p(\mathbf{z})$ is used as input to the generator, then the latter might be able to approximate both the curvature and the holes at the cost of increasing nonlinearity and/or changing topology. When considering this backwards—e.g. when mapping the misfit function $\gamma(\mathbf{x})$ in the latent space—the approximation of both high curvature and differences in topology often translate in discontinuities or high nonlinearities because a continuous mapping onto the uniform distribution is enforced. This results in high curvature being effectively "flattened" and holes effectively "glued", both of which cause distances to be highly distorted. In this work, we will call a generator "well-behaved" when it is only mildly nonlinear and preserves topology.

Both the generator's nonlinearity and its ability to change topology, may be controlled by two factors: (1) the generator architecture (type and size of each layer and total number of layers) and (2) the way it is trained (including training parameters). If the goal is to perform gradient-based inversion with DGMs, one should try to preserve convexity of $\gamma(\mathbf{x})$ as much as possible when mapping it to the latent space as $\gamma(\mathbf{g}(\mathbf{z}))$ while not degrading the generator's ability to reproduce the desired patterns. To aid in preserving such convexity, we propose to enforce the generator $\mathbf{g}(\mathbf{z})$ to be well-behaved. This means that the generator will approximate the real manifold $\mathscr{M}$ with a manifold $\mathscr{M}'$ with a moderate curvature and whose topology is the same as the region defined in latent space by $p(\mathbf{z})$. By enforcing a moderate curvature manifold, local oscillations that may give rise to local minima (as those shown in Fig. 1d) but only have minimum impact in pattern accuracy are avoided in the approximate manifold $\mathscr{M}'$ (the local minima are no longer present in Fig. 1e). In turn, when the generator is encouraged to preserve topology no more local minima should arise in $\mathbb{R}^d$ than the ones resulting from intersecting $\gamma(\mathbf{x})$ with the approximate manifold $\mathscr{M}'$ in $\mathbb{R}^D$ (note e.g. there is one local minima in both Fig. 1e and f). The latter is in line with the proposal of Falorsi et al. (2018), where they argue that for the purpose of representation learning (which basically means learning encodings that are useful for other tasks than just generative modeling) the mapping should preserve topology.

GANs often produce highly nonlinear generators that do not preserve topology, which may result in challenging inversion in the latent space. Laloy et al. (2018) provide an example of how architecture of a GAN is set to obtain a relatively well-behaved generator $\mathbf{g}(\mathbf{z})$. They propose to use a model called spatial generative adversarial network (SGAN) (Jetchev et al., 2017) that enforces different latent variables to affect different local regions in the ambient space. Their architecture results in a high compression (lower dimensionality of the latent space) and controls nonlinearity which allowed them to successfully perform MCMC-based inversion in the latent space. However, gradient-based deterministic inversion performed with the same DGM was shown to be highly dependent on the initial model (Laloy et al., 2019) pointing towards the existence of local minima. In addition, since training GANs is a rather complicated procedure where one has to find a balance between the performance of the generator and the discriminator, there is no straightforward way in which to modify such training to control nonlinearity. In this work we aim for robust gradient-based inversion in latent space by considering a VAE, the other predominant type of DGM, since its training may be tuned to produce a well-behaved generator.

2.3. VAE as DGM for inversion

A VAE is the model resulting from using a reparameterized gradient estimator for the evidence lower bound while applying (amortized) variational inference to an autoencoder, i.e. an architecture involving an encoder and a decoder which are both (possibly deep) neural networks (Kingma and Welling, 2014; Zhang et al., 2018). To train a VAE one uses a dataset $\mathbf{X} = \{\mathbf{x}^{(i)} | 1 \leq i \leq N\}$ where each $\mathbf{x}^{(i)}$ is a sample (e.g. an image) with the desired patterns and then maximizes the sum of the evidence (or marginal likelihood) lower bound of each individual sample. The evidence lower bound for each sample can be written as (Kingma and Welling, 2014)

$$\mathscr{L}\left(\theta, \vartheta; \mathbf{x}^{(i)}\right) = \mathscr{L}^x + \mathscr{L}^z \tag{10}$$

with

$$\mathscr{L}^x = \mathbb{E}_{q_\vartheta(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log\left(p_\theta\left(\mathbf{x}^{(i)}|\mathbf{z}\right)\right)\right] \tag{11}$$

and

$$\mathscr{L}^z = -D_{KL}\left(q_\vartheta\left(\mathbf{z}|\mathbf{x}^{(i)}\right)||p(\mathbf{z})\right) \tag{12}$$

where $\mathbf{z}$ refers to the codes or latent vectors, $p_\theta(\mathbf{x}|\mathbf{z})$ is the (probabilistic) decoder, $q_\vartheta(\mathbf{z}|\mathbf{x})$ is the (probabilistic) encoder, $\mathbb{E}$ denotes the expectation operator, D_{KL} denotes the Kullback-Leibler distance and, θ and ϑ are the parameters (weights and biases) of the DNNs for the decoder and encoder, respectively.

In order to maximize the evidence lower bound in Eq. (10), its gradient with respect to both θ and ϑ is required, however, this is generally intractable and therefore an estimator is used. This estimator is based on a so called reparameterization trick of the random variable $\widetilde{\mathbf{z}} \sim q_\vartheta(\mathbf{z}|\mathbf{x})$ which uses an auxiliary noise ε. In the case of a VAE, the encoder is defined as a multivariate Gaussian with diagonal covariance:

$$q_\vartheta(\mathbf{z}|\mathbf{x}) = \mathscr{N}(\mathbf{h}_\vartheta(\mathbf{x}), \mathbf{u}_\vartheta(\mathbf{x}) \cdot I_d) \tag{13}$$

where $\mathbf{h}_\vartheta(\mathbf{x})$ and $\log\mathbf{u}_\vartheta(\mathbf{x})$ are modeled with DNNs and I_d is a $d \times d$ identity matrix. Then, the encoder and the auxiliary noise ε are used in the following way during training (Kingma and Welling, 2014)

$$\widetilde{\mathbf{z}} = \mathbf{h}_\vartheta(\mathbf{x}) + \mathbf{u}_\vartheta(\mathbf{x}) \odot \varepsilon, \ \varepsilon \sim p(\varepsilon) \tag{14}$$

where $\odot$ denotes an element-wise product. Often Eq. (12) has an analytical solution, then only Eq. (11) is approximated with the estimator as (Kingma and Welling, 2014)

$$\mathscr{L}^x = \frac{1}{M}\sum_{j=1}^{M} \log\left(p_\theta\left(\mathbf{x}^{(i)}|\widetilde{\mathbf{z}}^{(i,j)}\right)\right) \tag{15}$$

where $\widetilde{\mathbf{z}}^{(i,j)} = \mathbf{h}_\vartheta\left(\mathbf{x}^{(i)}\right) + \mathbf{u}_\vartheta\left(\mathbf{x}^{(i)}\right) \odot \varepsilon^{(j)}$, $\varepsilon^{(j)} \sim p(\varepsilon)$ and M is the number of samples used for the estimator. Further, if we set the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ as a multivariate Gaussian with diagonal covariance structure, then

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathscr{N}(\mathbf{g}_\theta(\mathbf{z}), \mathbf{v}_\theta(\mathbf{z}) \cdot I_D) \tag{16}$$

where $\mathbf{g}_\theta(\mathbf{z})$ and $\log\mathbf{v}_\theta(\mathbf{z})$ are modeled with DNNs and I_D is a $D \times D$ identity matrix. In this work, we consider only the mean of the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ which is just the (deterministic) generator $\mathbf{g}_\theta(\mathbf{z})$. Then, the corresponding loss function may be written as

$$\mathscr{L}^x = \frac{1}{M}\sum_{j=1}^{M} \left\|\mathbf{g}_\theta(\widetilde{\mathbf{z}}^{(i,j)}) - \mathbf{x}^{(i)}\right\|^2 \tag{17}$$

The described setting allows for the gradient to be computed with respect to both θ and ϑ and then stochastic gradient descent is used to maximize the lower bound in Eq. (10). In the rest of this work, we drop the subindex θ in $\mathbf{g}(\mathbf{z})$ to simplify notation and also because once the DGM is trained, the parameters θ do not change, i.e. they are fixed for the subsequent inversion.

As previously mentioned, it is often possible to analytically integrate the Kullback-Leibler distance in Eq. (12). In this work, we consider that

$p(\mathbf{z})$ and $q_\vartheta(\mathbf{z}|\mathbf{x})$ are both Gaussian therefore Eq. (12) may be rewritten as (Kingma and Welling, 2014):

$$\mathscr{L}^z = \frac{1}{2}\sum_{i=1}^{d}\left(1+\log\left((u_i)^2\right)-(h_i)^2-(u_i)^2\right) \tag{18}$$

where the sum is done for the d output dimensions of the encoder.

Note that the term in Eqs. (11), (15) and (17) may be interpreted as a reconstruction term that causes the outputs of the encode-decode operation to look similar to the training samples, while the term in Eqs. (12) and (18) may be considered a regularization term that enforces the encoder $q_\vartheta(\mathbf{z}|\mathbf{x})$ to be close to a prescribed distribution $p(\mathbf{z})$. In practice, one may add a weight to the second term (Higgins et al., 2017) of the lower bound as:

$$\widetilde{\mathscr{L}}\left(\theta,\vartheta;\mathbf{x}^{(i)}\right) = \mathscr{L}^x + \beta\mathscr{L}^z \tag{19}$$

to prevent samples to be encoded far from each other in the latent space, which may cause overfitting of the reconstruction term and degrade the VAE's generative performance. The overall process of training and generation for a VAE is depicted in Fig. 2.

The effect of the regularization weight β is such that when increased the encoded training samples tend to lie closer to the prescribed probability distribution $p(\mathbf{z})$. Then, one may picture the transformation of the encoder as taking the low-dimensional approximate manifold in the ambient space and charting it (e.g. by bending, stretching and even folding) into the region defined by $p(\mathbf{z})$ in the latent space and the generator as the transformation undoing such charting. While the effect of β in a VAE is relatively easy to understand, the effect of the noise distribution $p(\varepsilon)$ is not so straightforward. First, note that the typical choice of a diagonal noise as $p(\varepsilon) = \mathcal{N}(0, \alpha \cdot I_d)$ where α denotes a constant variance (frequently set to $\alpha = 1.0$) is usually done for tractability or computational convenience (Kingma and Welling, 2014; Rolinek et al., 2019). However, it has been proposed recently that the choice of a diagonal noise has an impact on a property called disentanglement (Rolinek et al., 2019). Such disentanglement basically means that different latent directions control different independent characteristics of the training (or generated) samples. They explain that a diagonal $p(\varepsilon)$ might induce an encoding that preserves local orthogonality of the ambient space. In this work, we argue that the choice of a diagonal $p(\varepsilon)$ (which is usually done only for computational convenience) might be useful in producing a well-behaved generator.

In order to visualize the joint effect of α and β, Fig. 3 shows a synthetic example where samples in a two-dimensional ambient space lie close to a rotated "eight-shaped" manifold (Fig. 3a). In addition, to study the impact on inversion, a convex data misfit function $\gamma(\mathbf{x})$ in the same space (created synthetically with a negative isotropic Gaussian function) is shown in Fig. 3b. The latent space is also chosen two-dimensional for visualization purposes but recall that for a real case the dimensionality of the latent space is usually much lower than the one of the ambient space. Then, Fig. 4 considers nine different combinations for the values of α and β to show how the (nonlinear) generator $\mathbf{g}(\mathbf{z})$ maps a region of the latent space (denoted by the z-axes in the first three rows) into the ambient space (denoted by the x-axes in the last three rows) in order to approximate the manifold in Fig. 3a. To visualize the deformation caused by the generator, an orthogonal grid in the z-axes and its mapping into the x-axes (a deformed grid) are shown (both on the left of each inset). The corresponding encoded training samples are shown in red in the z-axes (left of each inset) and their reconstruction (resulting from the operation of encode-decode) is shown also in red in the x-axes (right of each inset), where also the original training samples are shown (in blue) to assess the accuracy of reconstruction. Samples obtained from a Gaussian distribution with a unitary diagonal covariance $p(\mathbf{z})$ are shown in the z-axes in orange (left of each inset), while their generator-mapped values are shown also in orange in the x-axes (right of each inset). Finally, the mapping of the data misfit function in Fig. 3b into the latent space is shown in the z-axes (right of each inset).

It is worth mentioning a few effects visible in the illustrative example of Figs. 3 and 4. First, note that increasing α seems to cause the grid to be more "rigid" locally (grid lines tend to intersect more at right angles) while going through the generator which may in turn help in preserving topology and controlling nonlinearity (e.g. compare the deformation of the grids for different values of α for $\beta = 0.01$), and more importantly, in preserving the convexity of the data misfit function in the latent space (the mapped misfit function using $\alpha = 0.1$ and $\beta = 0.01$ has a single global minimum, while the misfit function for $\alpha = 0.01$ and $\beta = 0.01$ has two minima in latent space). Also note that both α and β should be set in order to not cause a significant degradation in: (1) the reconstruction of the patterns, e.g. the cases of $\alpha = 1.0$ with both $\beta = 0.1$ and $\beta = 0.01$ show that the "eight-shape" is not completely reconstructed (seen in red samples not fully overlaying the blue samples in x-axes), or (2) the similarity of the encoded samples to the prescribed distribution $p(\mathbf{z})$, e.g. the case of $\alpha = 0.01$ and $\beta = 0.1$ shows that encoded samples (red dots in z-axes) are too concentrated (lower variance) and therefore far from the prescribed normal distribution with unit variance (orange dots in z-axes). In this case, the intermediate values ($\alpha = 0.1$ and $\beta = 0.01$) seem to provide the best choice in terms of reconstruction of the patterns, generative accuracy and convexity of the misfit function in latent space. Cases with ($\alpha = 1.0, \beta = 0.001$) and ($\alpha = 0.1, \beta = 0.001$) also have good performance but show two minor defects: (1) a bit higher number of generated samples over the "holes" (orange dots in x-axes) which would translate into higher number of inaccurate patterns, and (2) a higher number of encoded samples (red dots in z-axes) in low-density regions which means the misplaced training patterns will be harder to generate.

In summary, a generator $\mathbf{g}(\mathbf{z})$ that preserves topology and contains nonlinearity is the best choice for gradient-based inversion in the latent space because it preserves convexity of the objective function. Note, however, that if the topology of the probability distribution $p(\mathbf{z})$ is different to the one of the real manifold $\mathscr{M}$, this strategy may result in approximate manifolds $\mathscr{M}'$ that do not account for all topological differences—e.g. that partially cover holes of the real one (see e.g.

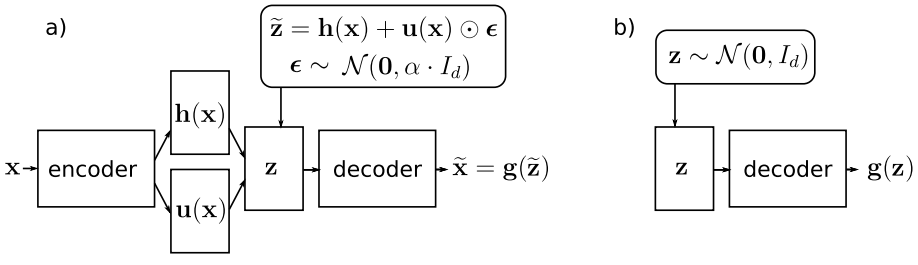

Fig. 2. A diagram for a VAE: (a) steps needed for training and (b) steps needed for generation.
Note that in setting up the VAE one has to choose: (1) the architectures of the encoder and decoder, (2) the probability distribution $p(\mathbf{z})$, (3) the noise distribution $p(\varepsilon)$ and (4) the regularization weight β. As mentioned in Section 2.2, these choices may impact the nonlinearity of the generator and its ability to preserve topology, which in turn affect the mapping of the data misfit function $\gamma(\mathbf{x})$ in latent space and possibly diminish the performance of inversion methods. While different choices in the architecture and probability distribution $p(\mathbf{z})$ may aid in obtaining a well-behaved generator, they are generally not straightforward and highly problem dependent. Therefore in this work we focus on the other two possible controls, the distribution $p(\varepsilon)$ and the regularization weight β, since they provide the simplest means of improving nonlinearity issues.

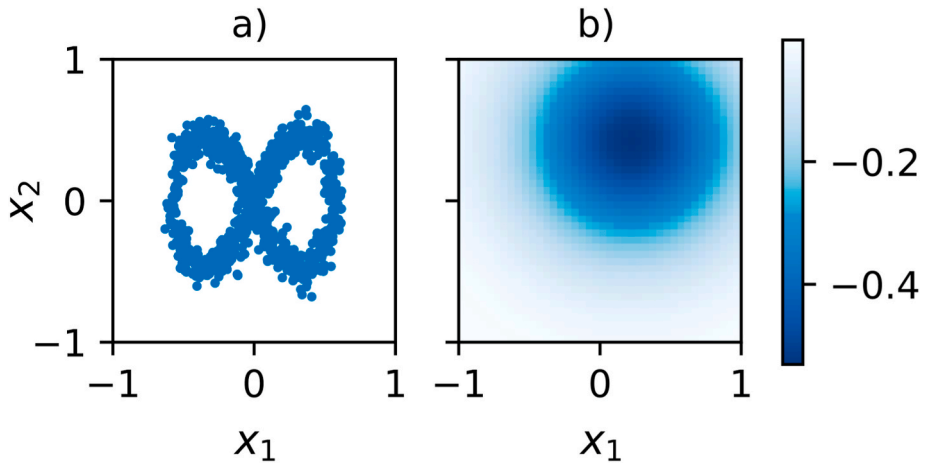

Fig. 3. Synthetic example of two-dimensional "eight-shaped" manifold: (a) training samples lying close to the manifold, and (b) synthetic misfit function $\gamma(\mathbf{x})$.

Fig. 1b)—and therefore might produce models that have non-accurate patterns when sampling from $p(\mathbf{z})$. We argue that the two training parameters α and β of a VAE may be chosen in order for the latter issue to not be severe, i.e. the generated patterns do not deviate too much from the training patterns, while still approximately preserving convexity of the objective function in the latent space.

To test our proposed method we implement a VAE in PyTorch (Paszke et al., 2017) and use training samples cropped from a "training image" which is large enough to have many repetitions of the patterns at the cropping size—a requirement similar in MPS. For our synthetic case, we use the training image of 2500 × 2500 pixels from Laloy et al. (2018) and the cropping size is chosen to fit the setting of our synthetic experiment (explained in detail in Sec. 2.5). Fig. 5a shows a patch of the training image and the position of the three (cropped) training samples shown Fig. 5b. Three generated samples from our proposed VAE trained with such croppings are shown Fig. 5c. Notice that the output of the generator is continuous (to allow for computation of gradients for training and inversion) with values between 0 and 1, and is later transformed to velocity values by a linear relation. For comparison, Fig. 5d shows three samples generated with the SGAN proposed by Laloy et al. (2019). Patterns of generated samples in Fig. 5c are not completely accurate comparing to those of the training image or the SGAN—they might display e.g. some breaking channels and smoothed edges (notice their output is also continuous but looks almost categorical). As mentioned above, this is expected for our proposed VAE because the approximate manifold fills some holes of the real manifold and may have less curvature. Also, the average proportion of channels from models generated from the VAE is a bit higher (0.36) than that of the training image (0.27). However, we argue that such inaccuracies may not cause significant error while performing inversion in practice because an informative dataset will generally make the inversion land in appropriate models (given the prescribed patterns were selected correctly). More importantly, in contrast to the SGAN, a modified gradient-based inversion (such as that presented in Sec. 2.4) will generally find a consistent minimum when applied with our proposed VAE regardless of the initial model.

2.4. Stochastic gradient descent with decreasing step size

Note that even when topology is preserved and nonlinearity is contained, the data misfit function in the latent space might still present some local minima. Using our proposed VAE approach in the synthetic case study, the resulting misfit function seems to have the shape of a global basin of attraction with some local minima of less amplitude. To deal with such remaining local minima we propose to use a SGD method instead of regular gradient-based optimization.

SGD methods are commonly used in training machine learning models to cope with large datasets (e.g. Kingma and Ba, 2017) and it has also been shown they are able to find minima that are useful in terms of generalization (Smith and Le, 2018). They essentially use an estimator for the gradient of the objective function computed only with a batch of the data. Such estimator is used in each gradient descent iteration and may be written for the case of inversion in the latent space as:

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \ell \cdot \nabla_{\mathbf{z}} \zeta(\mathbf{z})_k \tag{20}$$

where k denotes the iteration index, ℓ is the step size (or learning rate) and the gradient estimator $\nabla_{\mathbf{z}} \zeta(\mathbf{z})_k$ is computed by using Eq. (9) for a data batch (i.e. a subset of $\mathbf{y}$) which is different for each k-th iteration but of constant size b. Relying on such estimator makes SGD methods less likely to get trapped in local minima when the objective function has the shape of a global basin of attraction mentioned above (Kleinberg et al., 2018).

Recently, it has been proposed that using SGD may be seen as optimizing a smoothed version of the objective function obtained by convolving it with the gradient "noise" resulting from batching (Kleinberg et al., 2018). The degree of noise (and therefore the degree of smoothness) is controlled by the ratio of the learning rate to the batch size ℓ/b (Chaudhari and Soatto, 2018; Smith and Le, 2018). Therefore if we choose to decrease the value of ℓ (while keeping b constant) as the optimization progresses we might be able to achieve lower misfit values i.e. get sufficiently close to the global minimum. This may be implemented by using:

$$\ell_{k+1} = c_\ell \cdot \ell_k \tag{21}$$

where a constant value of $c_\ell < 1.0$ and a starting value ℓ_0 must be chosen. In practice, the method may be further improved by also decreasing the controlling (regularization) term in Eqs. (7) and (9) in order to prevent that large initial steps diverge from the region of the latent space where the manifold is defined (Bora et al., 2017; Luo et al., 2015). Then, similarly to ℓ this may be done as:

$$\lambda_{k+1} = c_\lambda \cdot \lambda_k \tag{22}$$

again a constant $c_\lambda < 1.0$ and a starting value λ_0 must be selected.

The combined effect of simultaneously decreasing ℓ and λ is illustrated in Fig. (6) for a simple synthetic problem in a two-dimensional ($d = 2$) latent space $\mathscr{R}^d$. The misfit term (i.e. first term of Eq. (7)) of the synthetic problem is shown in Fig. 6a. Assuming that $p(\mathbf{z})$ is a normal distribution $\mathscr{N}(0, I_d)$, we propose a specific regularization term $R(\mathbf{z})$ that will preferentially stay in the regions of higher mass (where most samples are located). This is done by radially constraining the search space by means of a χ-distribution, i.e. the regularization term is written as:

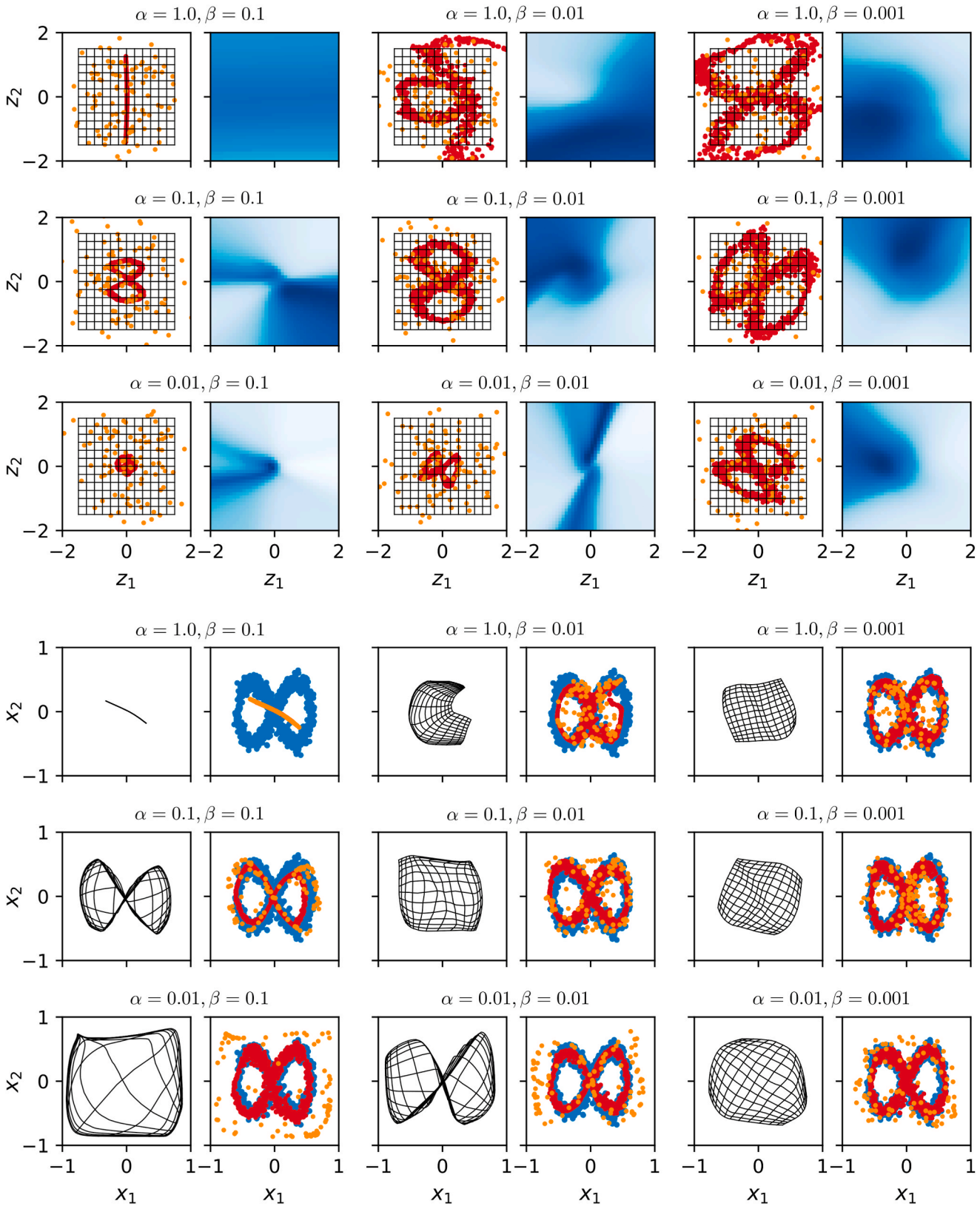

Fig. 4. Mapping a region of the latent space by the generator $\mathbf{g}(\mathbf{z})$ and mapping of the misfit function $\gamma(\mathbf{x})$ to the latent space with different values for α and β. The first three rows (z-axes) depict the latent space where each case shows: (left frame) orthogonal grid (black), encoded training samples (red) and generated samples (orange); (right frame) misfit function mapped in latent space (blue). The last three rows (x-axes) depict the ambient space where each case shows: (left frame) the same grid but mapped by the generator; (right frame) training (blue), reconstructed (red) and generated samples (orange). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

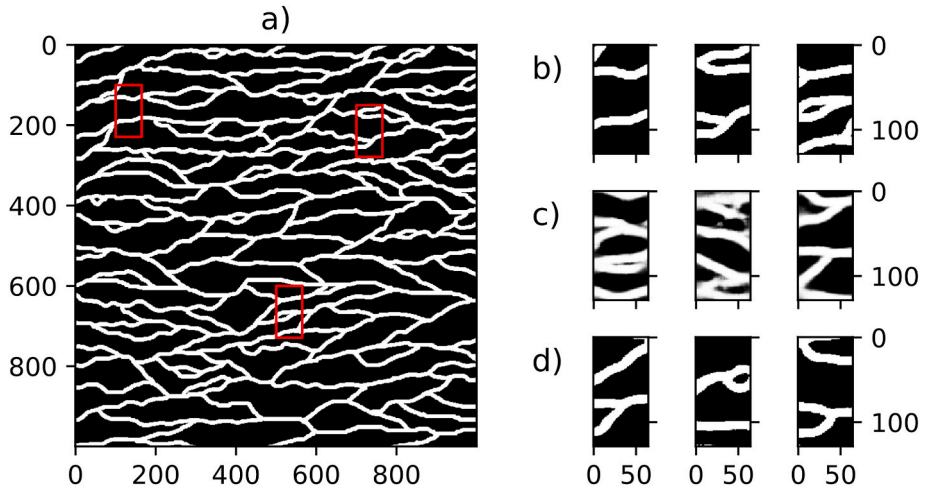

Fig. 5. (a) A 1000 × 1000 patch of the training image of Laloy et al. (2018), (b) cropped training samples whose location in (a) is shown red, (c) generated samples from our proposed VAE, and (d) generated samples from the SGAN proposed by Laloy et al. (2018). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

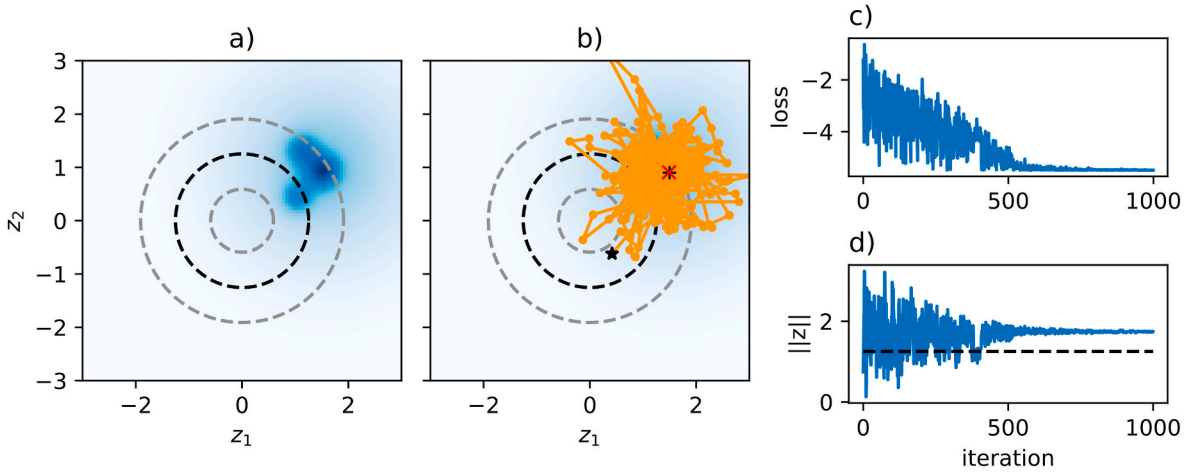

Fig. 6. Regularized gradient-based inversion in a synthetic two-dimensional latent space: (a) misfit (blue) and mean of χ-distribution (black dashed) together with 16- and 84-th percentiles (gray dashed), (b) the same setting of (a) with an overlay of an instance of optimization (trajectory in orange) for a random initial model (black '$\star$'), showing also final model (red ' × ') and true model (black ' + '), (c) misfit vs. iteration number, and (d) norm of $\mathbf{z}$ vs. iteration number. Dashed line in (d) corresponds to the norm of the radius defined by the mean of the χ-distribution. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

$$R(\mathbf{z}) = (||\mathbf{z}|| - \mu_\chi)^2 \tag{23}$$

where μ_χ is the mean for a χ-distribution with d degrees of freedom. We refer to this strategy as "ring" regularization since for a two-dimensional latent space it enforces inversion to preferentially stay within a region with the shape of a ring. Dashed lines in Fig. 6a denote this mean together with the 16- and 84-th percentiles. In general, this is especially useful for higher dimensionalities where most of the mass of a normal distribution is far from its center (Domingos, 2012). Then, Eq. (7) may be rewritten as:

$$\zeta(\mathbf{z}) = ||\mathbf{f}(\mathbf{g}(\mathbf{z})) - \mathbf{y}||^2 + \lambda(||\mathbf{z}|| - \mu_\chi)^2 \tag{24}$$

and correspondingly Eq. (9) may be expressed as:

$$\nabla_\mathbf{z}\zeta(\mathbf{z}) = \mathbf{J}(\mathbf{z})^T \nabla_\mathbf{x}\gamma(\mathbf{x}) + 2\lambda\mathbf{z}\left(1 - \frac{\mu_\chi}{||\mathbf{z}||}\right) \tag{25}$$

As mentioned above, this gradient is often computed simply by adding a layer to the autodifferentiation of the generator. One optimization instance for a random initial model is shown in Fig. 6b, while the behavior of the misfit and $\|\mathbf{z}\|$ is shown in Fig. 6c and d. Notice the rather "noisy" inversion trajectory, but also its ability to escape local minima. The effect of decreasing ℓ is seen in Fig. 6c by the decreasing of the oscillations amplitude as the optimization progresses, while the effect of decreasing λ is noticeable in Fig. 6d by the progressive shifting of $\|\mathbf{z}\|$ away from μ_χ.

The strategy described above and stated by Eq. (24) is generally applicable to DGMs that use an independent normal distribution as its probability distribution $p(\mathbf{z})$ and whose generator is well-behaved. In this work, we consider a VAE whose training parameters β and $p(\varepsilon)$ are chosen so that it results in a mildly nonlinear inversion for which such SGD strategy is generally useful.

2.5. Inverse problem: traveltime tomography

To test our proposed method and compare it with a previous instance of inversion with a DGM, we consider an identical setting to that used in Laloy et al. (2019). Such setting considers a dataset of crosshole ground

penetrating radar (GPR) traveltime tomography. To obtain a subsurface model $\mathbf{x} \in \mathbb{R}^D$ this method relies in contrasts of electromagnetic wave velocity which is related to moisture content and therefore to porosity for saturated media. The tomographic array considers a transmitter antenna in one borehole and a receiver antenna in the other, each of which is moved to different positions and a vector of measurements $\mathbf{y} \in \mathbb{R}^Q$ is obtained by taking the traveltime of the wave's first arrival for each transmitter-receiver combination. We assume that the sensed physical domain is a 6.5×12.9 m plane (i.e. the two-dimensional region between the boreholes) and is discretized in 0.1×0.1 m cells of constant velocity to represent spatial heterogeneity (i.e. a representation of $65 \times 129 = 8385$ cells is obtained). We consider a binary subsurface (e.g. composed of two materials with different porosity) with respective wave velocities of 0.06 and 0.08 m ns^{-1}. Measurements are taken every 0.5 m in depth (the first being at 0.5 m and the last at 12.5 m) resulting in a dataset of $Q = 625$ traveltimes. Note that though this model provides a good learning tool and a rather challenging test case, it is unrealistic, e.g. subsurface environments usually contain many more materials and further variability within each them. For one instance of our synthetic case, we add normal independent noise $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 \cdot I_Q)$ where σ^2 is the noise variance and I_Q is a 625×625 identity matrix. In the case a different noise distribution is used, one needs to add a weight matrix to the misfit term in Eq. (7) so that inversion takes such distribution into account, e.g. when different data points have different magnitudes for the noise, inversion should put more weight on those that are less affected by noise.

Similarly to Laloy et al. (2019), we first consider a fully linear forward operator $\mathbf{f}$ for which raypaths are always straight, i.e. independent of the velocity spatial distribution. For this case Eq. (4) may be rewritten as:

$$\mathbf{y} = \mathbf{F}\mathbf{x} + \boldsymbol{\eta} \tag{26}$$

where $\mathbf{F}$ is a matrix of dimension $Q \times D$ in which a certain row contains the length of the raypath in each cell of the model for a certain transmitter-receiver combination. The corresponding gradient of the misfit $\nabla_{\mathbf{x}}\gamma(\mathbf{x})$ to be used in Eq. (25) for the solution of the inversion is:

$$\nabla_{\mathbf{x}}\gamma(\mathbf{x}) = -2\mathbf{F}^T(\mathbf{y} - \mathbf{F}\mathbf{x}) \tag{27}$$

We also consider the case of a more physically realistic nonlinear forward operator $\mathbf{f}$ (see Eq. (4)) for which raypaths are not straight. In particular, we consider a shortest path (graph) method which uses secondary nodes to improve the accuracy of the simulated traveltimes as proposed by Giroux and Larouche (2013) and implemented in PyGIMLi (Rücker et al., 2017). For this case, when inversion with Eq. (25) is pursued, we linearize the forward operator $\mathbf{f}$ in order to compute the gradient:

$$\nabla_{\mathbf{x}}\gamma(\mathbf{x}) = -\mathbf{S}(\mathbf{x})^T(\mathbf{y} - \mathbf{f}(\mathbf{x})) \tag{28}$$

where $\mathbf{S}(\mathbf{x})$ is the $Q \times D$ Jacobian matrix of the forward operator whose elements are:

$$[\mathbf{S}(\mathbf{x})]_{i,j} = \frac{\partial f_i(\mathbf{x})}{\partial x_j} \tag{29}$$

The elements of the Jacobian $\mathbf{S}(\mathbf{x})$ are computed by the shortest path method and also represent lengths of raypaths. In contrast to the linear case, these have to be recomputed in every iteration. Both the nonlinear forward operator and the need for recomputing the Jacobian result in higher computational cost compared to the linear operator.

The method proposed in Sec. 2.4 to perform gradient-based inversion with a VAE should work for the linear forward operator because the nonlinearity in the inverse problem arises only due to the nonlinearity of the generator $\mathbf{g}(\mathbf{z})$ which is moderate when the latter is well-behaved. However, since the considered nonlinear forward operator in Eq. (28) is only mildly nonlinear (when contrast in velocities is not extreme), the same method may also provide good inversion results for this operator.

3. Results

3.1. Training of VAE

As previously mentioned, our proposed method relies on a VAE whose training parameters are selected in order to improve gradient-based inversion. The training samples are the croppings detailed in Sec. 2.3 whose dimensionality is $D = 8325$ and we consider a latent code of dimensionality $d = 20$. Different values for d were tested and $d = 20$ was chosen because higher values did not significantly improve the reconstruction of the training samples but did have a negative impact on the accuracy of the generated patterns (for this, generated patterns were assessed visually from a set of generated models such as those shown in Fig. 5c). Moreover, since the value of d also impacts the diversity of the generated patterns (i.e. how much they depart from training patterns), $d = 20$ provided a trade-off where patterns display sufficient diversity but still resemble those of the training image. The probability distribution $p(\mathbf{z})$ is an independent multinormal distribution $\mathcal{N}(0, I_d)$ with I_d an identity matrix of size 20×20. The architecture of the encoder and the decoder includes 4 convolutional layers, 2 fully-connected layers and instance normalization is used between each layer. The VAE has around 4.5 million parameters in total (weights and biases), which is a typical number for convolutional neural networks (further details may be consulted in the associated code). In order to show their impact on our proposed method, α and β are set to span three orders of magnitude. Table 1 shows the values of α and β that were used and their impact in the data RMSE of the linear case explained below. This means that nine different VAEs are trained for this test. Each VAE is trained by maximizing the lower bound in Eq. (19) using 10^5 iterations and batches of 100 random croppings in each iteration (a GeForce RTX 2060 GPU was used in which training took ~ 2 h). In the following, we first test the impact of α and β on inversion with a linear forward model. Then, we select the VAE with the best training parameters to study the impact of the different factors added in our approach (such as regularization and data batching) and make a comparison with methods from previous studies. Finally, we present some results of our approach using a mildly nonlinear forward operator.

3.2. Case with a linear forward model

In this section, we consider the linear operator in Eq. (26) and assess the performance of our proposed DGM inversion approach: using VAEs trained as detailed above and SGD with both decreasing step size and regularization to optimize Eq. (24). We aim to show that, when appropriate values of α and β are chosen, this approach is robust regarding its convergence to the global minimum and therefore assess its performance by using 100 different initial models. To test this, we considered three different true models (with different degrees of complexity) that were cropped from the training image and not considered during the VAE's training (models mc1, mc_2 and mc_3 in first row of Fig. 7). Table 1 shows the inversion data RMSE obtained for all combinations of α and β that were tested for this linear forward operator (the average data RMSE values are simply summed for the three true models). These results are

Table 1
Sum of average data RMSE for the cases mc_1, mc_2 and mc_3. The average is computed for 100 initial models for each case.

	$\beta = 10^4$	$\beta = 10^3$	$\beta = 10^2$
$\alpha = 1.0$	2.551	1.765	2.940
$\alpha = 0.1$	3.116	1.763	3.756
$\alpha = 0.01$	2.747	1.937	2.701

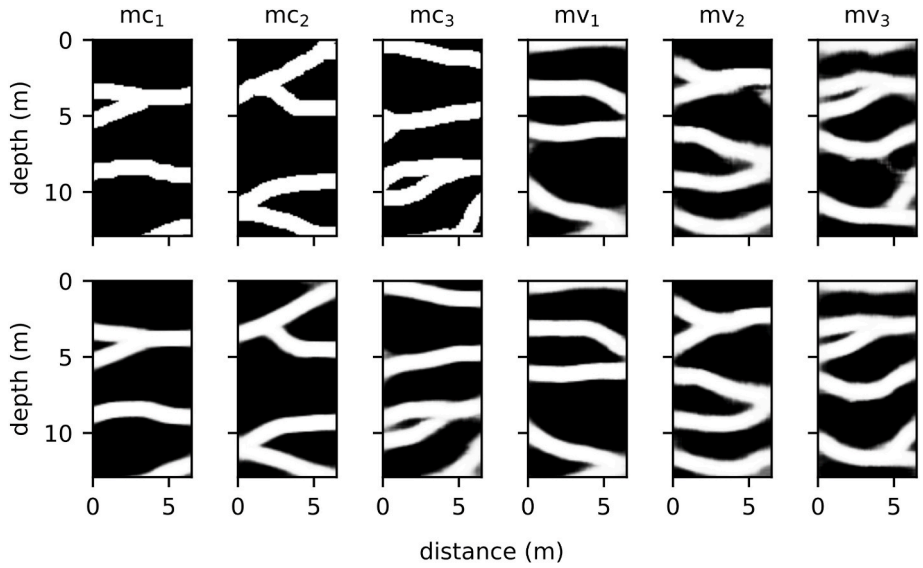

Fig. 7. Truth models (first row): cropped from training image (denoted by "mc") and generated from trained VAE (denoted by "mv"). Corresponding models resulting from encode-decode of truth models (second row). Subindex indicates level of complexity, with "1" being the least complex.

consistent with our explanation in Fig. 4, since both α and β have a noticeable impact on inversion performance. It is interesting to note that the values yielding the lowest data RMSE ($\alpha = 0.1$ and $\beta = 1000$) are not those typically used in previous studies ($\alpha = 1.0$ and $\beta < 100$). Also, the impact of α seems to be lower compared to β.

To further assess our approach and to compare with previous studies, only the VAE with the values yielding the lowest data RMSE is considered in the remainder of this section. The main differences of our proposed approach with the method of Laloy et al. (2019) are in the type of DGM and the optimization strategy. We make a comparison with their method and also to other base cases listed in Table 2 to show the impact of each factor involved in our approach. As denoted by the columns of this table, the different cases consider: (1) VAE and SGAN as DGMs, (2) SGD and Adam (Kingma and Ba, 2017) as stochastic optimizers, (3) data batching for computing the gradient $\nabla_{\mathbf{z}}\zeta(\mathbf{z})$, which basically means using SGD when batching and using (regular) gradient-descent when not batching, (4) regularization in the latent space, with "origin" being the one proposed in Bora et al. (2017) and "ring" the one proposed herein, and (5) decreasing of the step size (or learning rate). Our proposed approach is then labeled as "VSbrd". We also show the chosen values for the step size ℓ and its decreasing factor c_ℓ when applicable—for these cases the values of $\lambda = 10.0$ and $c_\lambda = 0.999$ are used. When data batching is used, the batch size b is 25 (of a total of 625) and is sampled with no replacement, then the whole dataset is used every 25 iterations (i.e. with 120 epochs, the total is $120 \times 25 = 3000$ iterations). The number of iterations for the cases with no data batching is also set to 3000. For our synthetic cases, once the DGMs are trained, there is no need for GPU acceleration to perform inversion, so all inversions were done in CPU. Compared to MCMC methods used in previous studies where the number of forward model evaluations was between 96,000 and 200,000 (Laloy et al., 2017, 2018) the computational cost is herein significantly reduced. Note that we also compare against the approach in Laloy et al. (2019), where SGAN is used as DGM and Adam (gradient-descent with adaptive moments) are used to optimize the resulting objective function—this case is labeled "SAnnn" in Table 2. The difference in computational time for this case (17.3 s) and our proposed method (7.3 s) was minor. We also consider the case where we apply our proposed SGD to the same SGAN (labeled as "SSbnd"). For both of these cases instead of regularization we use stochastic clipping in the latent space (Laloy et al. 2018, 2019) because a uniform $p(\mathbf{z})$ with finite support is used.

We consider 6 different true subsurface models to assess our method and compare with the base cases: (1) the set of three models cropped directly from the training image described above and (2) a set of three models obtained by generating from the trained VAE. Both sets include models with three different degrees of complexity. These truth models are shown in the first row of Fig. 7 where "mc" refers to the first set, "mv" refers to the second set and the degree of complexity is denoted by a subscript, where "1" denotes least complex and "3" most complex. The second set (mv) is similar to the one used by Laloy et al. (2019) to test the performance of their setup, only in their case the models were generated from a SGAN instead of a VAE. For each one of these truths, we generate synthetic data applying the forward operator $\mathbf{F}$ and use these data to perform gradient-based inversion for each case in Table 2.

We first consider no added noise to the synthetic dataset, hence after inversion the data misfit should be close to zero for inverted models that are sufficiently close to the global minimum. To define a threshold for this data misfit beyond which inverted models are "accepted", we use the RMSE between these synthetic data and data obtained by applying the forward operator on models resulting from passing the truth models through a VAE's encoding-decoding (these models are shown in the second row of Fig. 7 and the corresponding values for the threshold are

Table 2
Configuration of our proposed approach (VSbrd) and the base cases for comparison. The case marked with * corresponds to the one considered by (Laloy et al., 2019).

Case	DGM	GD	Data batching	Regularization	Decreasing	ℓ	c_ℓ
VSnnn	VAE	SGD	no	none	no	1e-4	–
VSbnn	VAE	SGD	yes	none	no	1e-4	–
VSbod	VAE	SGD	yes	origin	yes	1e-2	0.95
VSbrd	VAE	SGD	yes	ring	yes	1e-2	0.95
SAnnn*	SGAN	Adam	no	none	no	1e-2	–
SSbnd	SGAN	SGD	yes	none	yes	1e-3	0.95

shown in Table 3). This is done because we found the encode-decode reconstructed models to be visually very similar to the truth models (compare first and second rows of Fig. 7) and also show a low model RMSE when compared to them. The model RMSE is computed as the difference of pixel values (previous to transforming to velocity values, so they have values between 0 and 1) between truth model and the encode-decode model and shown Table 3. Once such a threshold is defined for each truth model, gradient-based inversion is run for the same 100 initial models for all cases in Table 2. Note that no convergence criteria were set in order to compare to all base cases (some cases such as "SAnnn" do not allow for easily defining such criteria) but in practice it is possible to set them for our proposed approach (VSbrd) in terms of a minimal change in either step size and/or data misfit. This also means that for some cases (including our proposed VSbrd) the 3000 iterations may not be necessary for all truths and all initial models. Results for the number of accepted inverted models are shown in Table 4 while the corresponding mean of the misfit (expressed as RMSE) for the 100 inversions is shown in Table 5.

As seen in Table 4, given our defined threshold: (1) the cases where VAE and SGD with decreasing step were used (VSbod and VSbrd) resulted in all inverted models being accepted, (2) the cases where SGAN was used (SAnnn and SSbnd) resulted in all models being rejected, and (3) the cases where VAE and non-decreasing step size SGD was used (VSnnn and VSbnn) resulted in some inverted models being accepted. Note also that using SGD (data batching) without a decreasing step size (VSbnn) results in less accepted models compared to GD (VSnnn), highlighting the importance of our proposed decreasing step size and regularization. As shown in Table 5 a higher mean RMSE is related to a lower number of accepted models. Furthermore, Table 5 shows that there is a general improvement caused by our proposed regularization compared to the one from Bora et al. (2017).

Examples of inverted models obtained for the different cases in Table 2 using the cropped truth with moderate complexity (mc_2) are shown in Fig. 8. Here, truth models are shown in Fig. 8a while Fig. 8b shows one example of an accepted model for cases that have at least one (VSnnn, VSbnn, VSbod and VSbrd). Similarly, Fig. 8c shows one example of a rejected model for applicable cases (VSnnn, VSbnn, SAnnn and SSbnd). Finally, the corresponding data RMSE vs. iteration number plots are shown in Fig. 8d (in blue for accepted models and red for rejected ones) and corresponding model RMSE plots are shown in Fig. 8e. Note both the higher similarity with the truth model (i.e. note the low model RMSE and compare models in Fig. 8b and c with those in Fig. 8a) and the lower RMSE for accepted models. Also, examples of inverted models for our proposed approach (VSbrd) using all the truths are shown in Fig. 9b, together with plots of RMSE vs. iteration number (Fig. 9d) and norm of $\mathbf{z}$ vs. iteration number (Fig. 9e). For cropped truths (mc) it seems that visual similarity decreases and final data RMSE of inverted models increases as complexity increases, whereas for generated truths they seem independent of complexity. Notice the overshoot in $||\mathbf{z}||$ in the initial iterations and its eventual convergence close to μ_χ as defined in Eq. (23).

To study the effect of noise for our proposed approach (VSbrd), we added noise with a standard deviation $\sigma = 0.25$ ns to the synthetic traveltime data. Corresponding results are shown in the rightmost column of Tables 4 and 5 and in Fig. 9c (with corresponding data RMSE and $\mathbf{z}$ norm plots in Fig. 9d and e). The threshold in this case is set equal to the one for the noise-free case plus σ and when using it all inverted models with our proposed approach are accepted. It is also worth noticing the relative robustness of the method to noise, as shown by the corresponding mean misfit values in Table 5 that indicate no significant overfitting, i.e. the mean misfit values are close to the noise-free threshold plus σ even if no traditional regularization was used. The latter means that optimizing in the latent space of the DGM is effectively constraining the inverted models to display the prescribed patterns. A higher value of $\sigma = 1.0$ ns was also tested which produced similar results (not shown).

Table 3
Data RMSE (ns) of encode-decode operation used to define thresholds (for the linear forward operator) and corresponding model RMSE.

	data RMSE (ns)	model RMSE (−)
mc_1	0.724	0.112
mc_2	0.854	0.133
mc_3	1.395	0.176
mv_1	0.749	0.097
mv_2	1.380	0.146
mv_3	1.436	0.145

Table 4
Number of accepted inversions (using 100 different initial models) according to the defined threshold.

	VSnnn	VSbnn	VSbod	VSbrd	SAnnn	SSbnd	VSbrd (noise)
mc_1	91	33	100	100	0	0	100
mc_2	86	59	100	100	0	0	100
mc_3	91	35	100	100	0	0	100
mv_1	91	30	100	100	0	0	100
mv_2	95	71	100	100	0	0	100
mv_3	98	77	100	100	0	0	100

3.3. Case with a nonlinear forward model

After showing that our proposed method works with the linear forward operator for the synthetic case considered, we now test its performance with a nonlinear forward operator. For inversion, the general form of Eq. (4) is used and the gradient in the latent space given in Eq. (25) is computed using Eq. (28). As mentioned in Sec. 2.5, we consider a shortest path method to solve for the traveltime for which we use 3 secondary nodes added to the edges of the velocity grid. Note that the Jacobian $\mathbf{S}(\mathbf{x})$ in Eq. (28) has to be recomputed at every iteration. Given the higher computational demand for inversion with the nonlinear forward operator and since it was already shown to be the best performing approach for the linear forward operator, we only test our proposed approach VSbrd with all the truths and for a single initial model (Fig. 10). This was done both without noise and with noise added using the same standard deviation $\sigma = 0.25$ as in the linear operator scenario. We select the following values for the required inversion parameters: $\ell = 0.1$, $c_\ell = 0.8$, $\lambda = 1.0$ and $c_\lambda = 0.99$. The total number of iterations is 750 with data batching of size 25 similar to the linear case. Note that to further reduce the number of iterations required for inversion we use a lower c_ℓ compared to the linear case, but the decreasing in Eq. (21) is only done every 5 iterations. This may cause the method to converge to the global minimum with lower probability, however it seems to still be high enough since all of the inversions with no added noise are very similar to the truth models. Also, using the threshold obtained by encoding-decoding the truth models (now computed with the nonlinear forward operator) all inverted models are accepted (these models are shown in Fig. 10b). When considering added noise, results are similar but inversion seems to converge to the global minimum with slightly lower probability (6 out of 8 inversions are accepted) and accepted models are shown in in Fig. 10c. The behavior of the misfit during optimization (Fig. 10d) is similar to the linear case, although oscillations of a slightly higher amplitude are still visible in the last iterations (mainly due to the lower number of iterations). To partially solve the latter issue, we take as inverted model the model with lowest misfit and not the one for the final iteration (these are the models shown in Fig. 10b and c). The plot of the norm of $\mathbf{z}$ vs. iterations in Fig. 10e shows a similar behavior to the linear case, although there seems to be more oscillations in $||\mathbf{z}||$ during initial iterations.

Table 5
Mean RMSE (ns) of inversions using 100 different initial models and defined threshold for accepting models.

VSnnn	VSbnn	VSbod	VSbrd	SAnnn	SSbnd	threshold	VSbrd	(noise)
mc_1	0.536	1.169	0.551	0.434	4.538	3.988	0.724	0.501
mc_2	0.832	1.518	0.626	0.541	5.266	4.495	0.854	0.583
mc_3	0.908	1.543	0.853	0.788	3.298	3.775	1.395	0.827
mv_1	0.296	1.418	0.353	0.055	3.952	4.226	0.749	0.259
mv_2	0.568	1.286	0.618	0.078	4.161	5.251	1.380	0.268
mv_3	0.557	0.854	0.232	0.036	4.591	5.537	1.436	0.256

4. Discussion

For both our toy example in Fig. 4 and our synthetic case for the linear forward operator (Table 1), results show that α and β have an impact on inversion. While in both cases the value yielding the lowest data RMSE for α is 0.1, β spans a larger range of values. This occurs mainly because α is coupled to the imposed unit variance of $p(\mathbf{z})$, since larger values of α tend to place samples further apart in the latent space and therefore make it inconsistent with $p(\mathbf{z})$. In contrast, due to the nature of the VAE training loss function in Eq. (19), β depends on the dimensionality of both the training samples (D) and the latent vectors (d). In order to have more comparable values between studies, Higgins et al. (2017) proposed normalizing β as $\beta' = \beta \times d/D$. In our synthetic case, the value of $\beta = 1000$ yields $\beta' = 2.4$ which is still high compared to what Higgins et al. (2017) found for an optimal disentangling (for $d = 20$) or to what Laloy et al. (2017) used in their study (both around $\beta' = 0.1$). This may be related to the fact that these studies focused either on disentangling or on generative accuracy for selecting β, instead of inversion performance as done in here. Note that normalized values of β are the most appropriate to provide guidelines for future studies. Our results suggest that setting $\beta' > 1.0$ may be useful for inversion, but further testing with different kinds of patterns is still required to support this.

In order to select SGD parameters in our proposed approach, we suggest looking jointly at the behavior of the misfit and norm of $\mathbf{z}$. For instance, if a certain number of iterations is desired for computational reasons, we suggest choosing first ℓ' and $c_{\ell'}$ that produce a behavior of the misfit similar to that in Fig. 9d, i.e. oscillations of high amplitude at the beginning and then progressive attenuation of the oscillations in such a way that at the end they are negligible. Note, however, that inversion may have to be run a few times because divergence may occur during initial iterations (this is easily seen in the value of $||\mathbf{z}||$ taking values far from μ_χ). Once ℓ' and $c_{\ell'}$ are chosen, the selection of λ and c_λ is done only to prevent divergence, this may be achieved by looking for a behavior similar to that in Fig. 9e. An initial overshoot in $||\mathbf{z}||$ is normal (and even necessary) since the method is exploring more rapidly the latent space, however, it should eventually converge to a value close to μ_χ.

The results for gradient-based inversion using our proposed approach point to a (possible) conflict between the accuracy of the reproduced patterns and the feasibility of gradient-based inversion with DGMs. As mentioned above, this is due to a non-convex objective function in latent space resulting from the generator's nonlinearity and its induced changes in topology. In this work, we argue that nonlinearity and changes in topology might be safely controlled by selecting certain values of α and β while training a VAE in order to improve performance of gradient-based inversion. We empirically show the validity of this statement by considering different values of α and β for our linear case study. In general (for inversion with DGMs), this implies that a tradeoff between generative accuracy and a well-behaved generator may be found. The latter statement also supports our assumption regarding the "holes" of the real manifold for the case of channel patterns (as mentioned in Section 2.3): when approximating the real manifold using a VAE with a well-behaved generator, the approximate manifold will tend to fill the holes and therefore produce breaking channels. While the generator's nonlinearity was already identified by Laloy et al. (2019) as a potential factor for hindering gradient-based inversion, its causes (curvature and topology of the real manifold) and the possible induced changes in topology have not been previously explained as factors in degrading the performance of gradient-based inversion in the latent space (to the authors' knowledge).

In general, good performance of DNNs for some tasks is usually associated with their ability to change topology (Naitzat et al., 2020). However, when one wants to use the latent variables or codes of DGMs for further tasks and not just for generation, these changes in topology might become an issue. For instance, we interpret the misfit "jumps" seen in gradient-based inversion with SGAN (as seen in Fig. 8c for case SAnnn) as resulting from the "gluing" or "collapsing" in latent space of holes in the real manifold—either caused by an induced change in topology or a high nonlinearity in the SGAN generator. Some studies have even suggested that if one wants to obtain useful geometric interpretations in the latent space (e.g. to perform interpolation), the activation functions should be restricted to ones that are smooth (Shao et al., 2017; Arvanitidis et al., 2018), that means e.g. not using the ReLU activation function that is generally recognized to result in faster learning. In contrast, in this work we do consider ReLU activation functions but control the changes in topology by means of a combination of α and β, whether this might nullify the advantages of ReLU is still an open question. Note however that, in general, control of induced changes in topology and high nonlinearities (as in our proposed approach) might be useful for any inversion method that relies in the concept of a neighborhood (e.g. MCMC and ensemble smoothers).

Besides its good performance for gradient-based inversion, a further advantage of our approach when compared to the previous approaches is that when the data used for inversion is not sufficiently informative, regularization in the latent space might be used to constrain to the most common patterns with our regularization term in Eq. (23). This statement provides an interesting paradigm where regularization in latent space might be seen as a flexible way to incorporate complex regularization. In contrast, a disadvantage of our proposed approach is that GANs in general result in higher generative accuracy (all generated patterns look more similar to those in the training image). However, as previously mentioned this may negatively affect inversion performance, at least for gradient-based methods. Also, as may be noticed in the relation between the data misfit and the degree of complexity for cropped truths, a limiting factor in using our VAE is its inability to produce new highly complex patterns. Nevertheless, this lack of innovation (or sample diversity) is generally present in other methods and may be even more severe for regular GANs, where the phenomenon is known as mode collapse. Recently, different ways to control such mode collapse in VAEs and GANs have been proposed (Metz et al., 2017; Salimans et al., 2016).

Regarding the SGD optimization method proposed, we must note that similar results might be obtained with a MCMC method where information about the gradient is taken into account. For example, Mosser et al. (2018) use a Metropolis-adjusted Langevin method which basically follows a gradient-descent and adds some noise to the step. However, the noise added to the gradient step in our approach is different—SGD noise has been shown to be approximately constant but anisotropic (Chaudhari and Soatto, 2018). Another possible alternative to our method is to

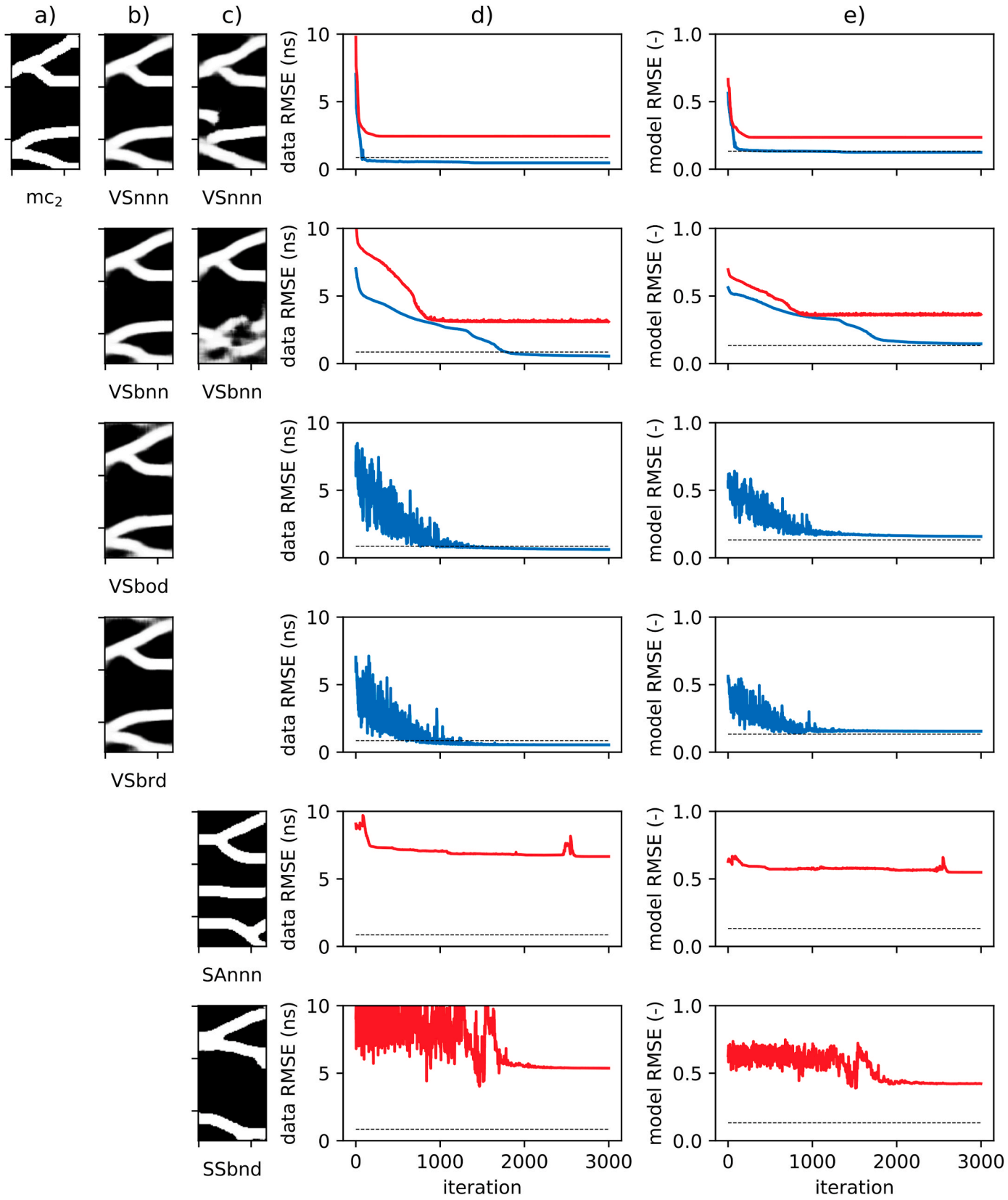

Fig. 8. Examples of inverted models for mc_2 truth for all cases in Table 2: (a) truth model, (b) accepted models according to defined threshold, (c) rejected models, (d) data RMSE vs. iterations plots (blue for accepted models and red for rejected models and dashed line indicates defined threshold) and (e) model RMSE vs. iterations plots (dashed line indicates model RMSE for encode-decode operation). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

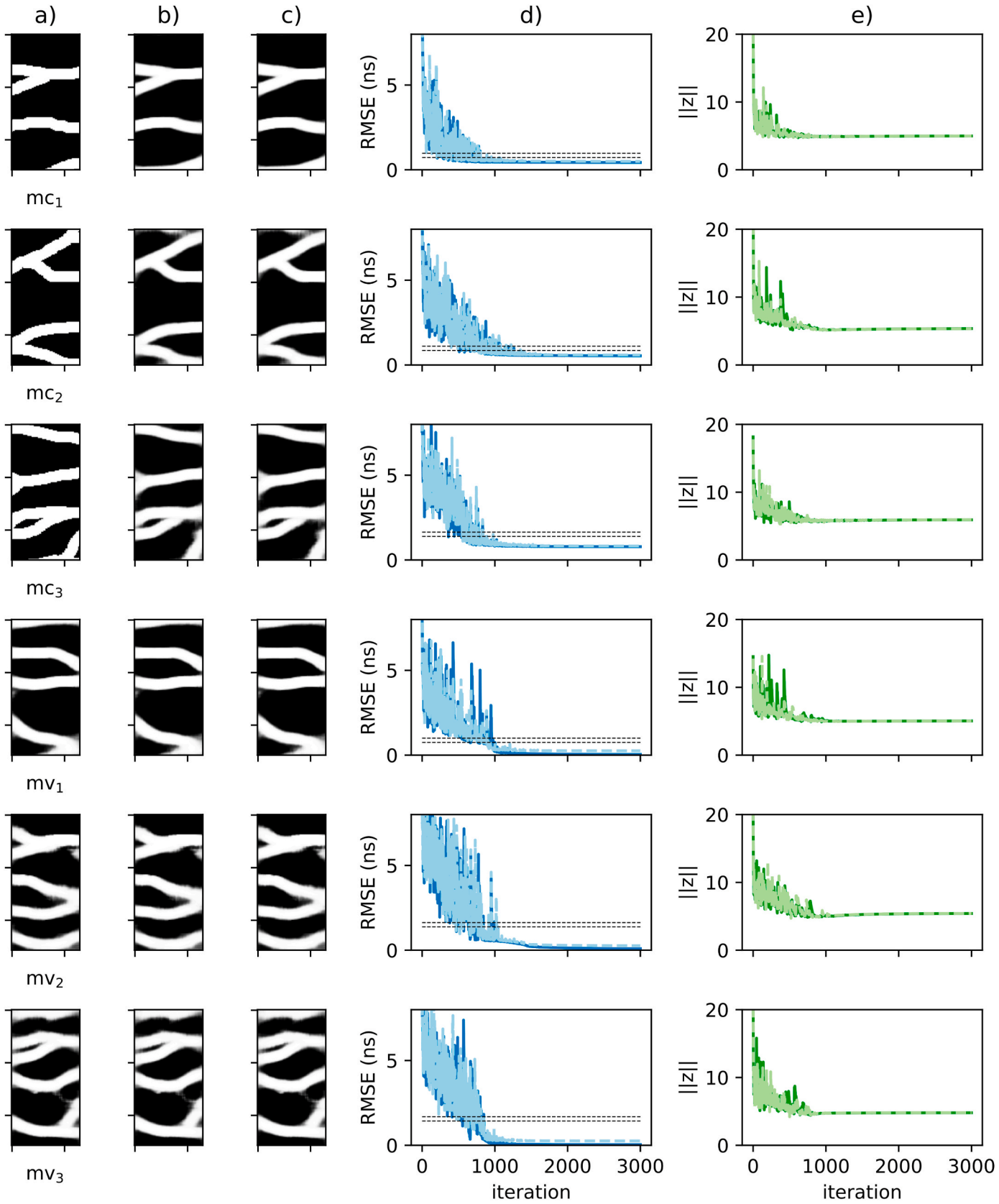

Fig. 9. Examples of gradient-based inversion using our proposed approach (VSbrd) for all truth models and the linear forward operator: (a) truth models, (b) inverted models with no added noise, (c) inverted models with added noise, (d) RMSE vs. iterations plots (no noise case in dark blue and noise case in light blue; lower dashed line indicates the defined threshold while upper dashed line is threshold plus $\sigma = 0.25$), and (e) norm of $\mathbf{z}$ vs. iterations plots (no noise case in dark green and noise case in light green). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

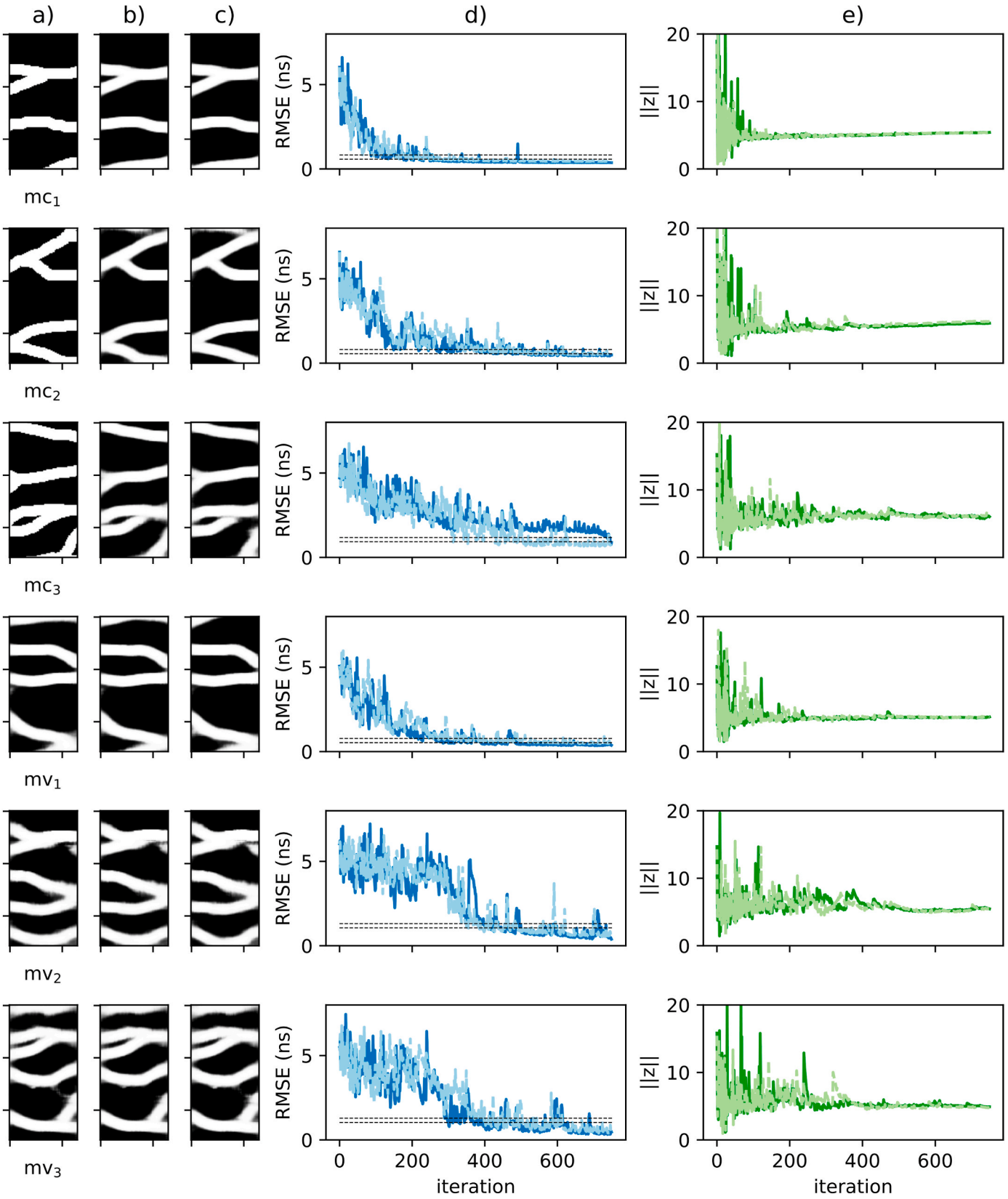

Fig. 10. Examples of gradient-based inversion using our proposed approach (VSbrd) for all truth models and the nonlinear forward operator: (a) truth models, (b) inverted models with no added noise, (c) inverted models with added noise, (d) RMSE vs. iterations plots (no noise case in dark blue and noise case in light blue; lower dashed line indicates the defined threshold while upper dashed line is threshold plus $\sigma = 0.25$), and (c) norm of $\mathbf{z}$ vs. iterations plots (no noise case in dark green and noise case in light green). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

use Riemannian optimization, which is possible when the DGM approximate manifold is smooth. Although it is possible to compute the direction of the gradient by using the pullback Riemannian metric, which may be obtained as suggested by e.g. Shao et al. (2017); Chen et al. (2018); Arvanitidis et al. (2018), it is not straightforward to compute the step because it would have to be along a geodesic curve instead of a straight path and such geodesics are computationally demanding to obtain.

Finally, we acknowledge that in order to be applied for a variety of field conditions, our proposed method needs to be extended to: (1) handle multiple materials (i.e. not a binary subsurface), (2) consider further variability inside each material, (3) estimate the velocity values directly (i.e. not assume they are known as was done above), (4) consider larger domains, and (5) condition to observed values of materials (e.g. in wells). To address the first two points and since DGMs are not restricted to categorical outputs, the VAE could simply be trained using samples with continuous outputs, however, the accuracy of the patterns may not be as good as in the binary case for a training image of the same size (Laloy et al., 2018). If one chooses to approximate the subsurface with a multi-categorical output, a different and more consistent loss function for the training such as the cross-entropy loss may give better results. Regarding the estimation of velocity values, a simple way to achieve this for binary models would be to include two extra parameters in inversion by assuming a linear relationship to shift and scale the output of the VAE. A similar approach may be used for multi-categorical or continuous outputs, although its usefulness may be more limited since the scaling and shifting operations do not significantly change the contrasts between the materials from those of the DGM outputs. When the spatial domain being studied is large or, more specifically, when it has many repetitions of the patterns, our method would require a very large training image which is generally difficult to obtain. A possible solution for this is to use an architecture that is more efficient for repetitive patterns. For instance, one may propose a spatial VAE (similar to the spatial GAN) which relies on 2D or 3D tensors instead of vectors as latent variables. Of course one then would need to test that efficient inversion (e.g. gradient-based) is still possible with such an architecture. Finally, conditioning to direct material observations may be achieved by adding a term to the inversion objective function in Eq. (7), although it has been shown that this does not produce perfect fitting to such observations (Laloy et al., 2017, 2018) so further study in this topic is required.

5. Conclusions

In this work both the impact and the causes of nonlinearity on inversion with DGMs are studied and a conflict between generated pattern accuracy and feasibility of gradient-based inversion is identified. Also, an approach based on a VAE as DGM and a modified stochastic gradient descent method for optimization is proposed to address such conflict. We show that two training parameters of the VAE (the weight factor β and the variance α of the encoder's noise distribution $p(\varepsilon)$) may be chosen in order to obtain a well-behaved generator $\mathbf{g}(\mathbf{z})$, i.e. one that is mildly nonlinear and approximately preserves topology when mapping from latent space to ambient space. This helps in maintaining the convexity of the misfit function in the latent space and therefore improves the behavior of gradient-based inversion. We highlight changes in topology which have not been previously identified as impacting the convexity of the inversion objective function. In contrast to prior studies where gradient-based inversion was used, our approach converges to the neighborhood of the global minimum with very high probability for both a linear forward operator and a mildly nonlinear forward operator with and without noise. We argue that when using DGMs in inversion, a tradeoff may be found where inverted models are close enough to the prescribed patterns while low cost gradient-based inversion is still applicable. Indeed, our proposed approach finds such tradeoff and produces inverted models with significant similarity to the training patterns and a sufficiently low data misfit.

Computer code availability

All code necessary to reproduce the test case is available at: https://github.com/jlalvis/VAE_SGD.

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.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement number 722028 (ENIGMA ITN). We thank the anonymous reviewers and the editor for their valuable comments that greatly improved the manuscript.

References

Armstrong, M., Galli, A., Beucher, H., Loc'h, G., Renard, D., Doligez, B., Eschard, R., Geffroy, F., 2011. Plurigaussian Simulations in Geosciences. Springer Berlin Heidelberg, Berlin, Heidelberg. URL. http://link.springer.com/10.1007/978-3-642-19607-2.

Arvanitidis, G., Hansen, L.K., Hauberg, S., Jan. 2018. Latent Space Oddity: on the Curvature of Deep Generative Models arXiv:1710.11379 [stat]ArXiv: 1710.11379. URL. http://arxiv.org/abs/1710.11379.

Aster, R., Borchers, B., Thurber, C., 2013. Parameters Estimation and Inverse Problems, second ed. Academic press.

Bergmann, U., Jetchev, N., Vollgraf, R., Sep. 2017. Learning Texture Manifolds with the Periodic Spatial GAN arXiv:1705.06566 [cs, stat]ArXiv: 1705.06566. URL. http://arxiv.org/abs/1705.06566.

Bora, A., Jalal, A., Price, E., Dimakis, A.G., Mar. 2017. Compressed Sensing Using Generative Models arXiv:1703.03208 [cs, math, stat]ArXiv: 1703.03208. URL. http://arxiv.org/abs/1703.03208.

Caers, J., Hoffman, T., Jan. 2006. The probability perturbation method: a new look at bayesian inverse modeling. Math. Geol. 38 (1), 81–100. URL. http://link.springer.com/10.1007/s11004-005-9005-9.

Canchumuni, S.W., Emerick, A.A., Pacheco, M.A.C., Jul. 2019. Towards a robust parameterization for conditioning facies models using deep variational autoencoders and ensemble smoother. Comput. Geosci. 128, 87–102. URL. https://linkinghub.elsevier.com/retrieve/pii/S0098300419300378.

Caterina, D., Hermans, T., Nguyen, F., Aug. 2014. Case studies of incorporation of prior information in electrical resistivity tomography: comparison of different approaches. Near Surf. Geophys. 12, 451–465. URL. http://nsg.eage.org/publication/publicationdetails/?publication=76904.

Chaudhari, P., Soatto, S., Jan. 2018. Stochastic Gradient Descent Performs Variational Inference, Converges to Limit Cycles for Deep Networks arXiv:1710.11029 [cond-mat, stat]ArXiv: 1710.11029. URL. http://arxiv.org/abs/1710.11029.

Chen, N., Klushyn, A., Kurle, R., Jiang, X., Bayer, J., van der Smagt, P., Feb. 2018. Metrics for Deep Generative Models arXiv:1711.01204 [cs, stat]ArXiv: 1711.01204. URL. http://arxiv.org/abs/1711.01204.

Domingos, P., Oct. 2012. A few useful things to know about machine learning. Commun. ACM 55 (10), 78. URL. http://dl.acm.org/citation.cfm?doid=2347736.2347755.

Falorsi, L., de Haan, P., Davidson, T.R., De Cao, N., Weiler, M., Forré, P., Cohen, T.S., Jul. 2018. Explorations in Homeomorphic Variational Auto-Encoding arXiv:1807.04689 [cs, stat]ArXiv: 1807.04689. URL. http://arxiv.org/abs/1807.04689.

Fefferman, C., Mitter, S., Narayanan, H., Feb. 2016. Testing the manifold hypothesis. J. Am. Math. Soc. 29 (4), 983–1049. URL. http://www.ams.org/jams/2016-29-04/S0894-0347-2016-00852-4/.

Giroux, B., Larouche, B., Apr. 2013. Task-parallel implementation of 3D shortest path raytracing for geophysical applications. Comput. Geosci. 54, 130–141. URL. https://linkinghub.elsevier.com/retrieve/pii/S0098300412004128.

González, E.F., Mukerji, T., Mavko, G., Jan. 2008. Seismic inversion combining rock physics and multiple-point geostatistics. Geophysics 73 (1), R11–R21. URL. http://library.seg.org/doi/10.1190/1.2803748.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., Jun. 2014. Generative Adversarial Networks arXiv:1406.2661 [cs, stat]ArXiv: 1406.2661. URL. http://arxiv.org/abs/1406.2661.

Hand, P., Voroninski, V., Dec. 2018. Global Guarantees for Enforcing Deep Generative Priors by Empirical Risk arXiv:1705.07576 [cs, math]ArXiv: 1705.07576. URL. http://arxiv.org/abs/1705.07576.

Hansen, T.M., Cordua, K.S., Mosegaard, K., Jun. 2012. Inverse problems with non-trivial priors: efficient solution through sequential Gibbs sampling. Comput. Geosci. 16 (3), 593–611. URL. http://link.springer.com/10.1007/s10596-011-9271-1.

Hermans, T., Vandenbohede, A., Lebbe, L., Martin, R., Kemna, A., Beaujean, J., Nguyen, F., 2012. Imaging artificial salt water infiltration using electrical resistivity tomography constrained by geostatistical data. J. Hydrol. 438–439, 168–180.
Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A., 2017. Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, vol. 13.
Jetchev, N., Bergmann, U., Vollgraf, R., Sep. 2017. Texture Synthesis with Spatial Generative Adversarial Networks arXiv:1611.08207 [cs, stat]ArXiv: 1611.08207. URL. http://arxiv.org/abs/1611.08207.
Kim, J., Zhang, B.-T., Jan. 2019. Data Interpolations in Deep Generative Models under Non-simply-connected Manifold Topology arXiv:1901.08553 [cs, stat]ArXiv: 1901.08553. URL. http://arxiv.org/abs/1901.08553.
Kingma, D.P., Ba, J., Jan. 2017. Adam: A Method for Stochastic Optimization arXiv: 1412.6980 [cs]ArXiv: 1412.6980. URL. http://arxiv.org/abs/1412.6980.
Kingma, D.P., Welling, M., May 2014. Auto-Encoding Variational Bayes arXiv:1312.6114 [cs, stat]ArXiv: 1312.6114. URL. http://arxiv.org/abs/1312.6114.
Kleinberg, R., Li, Y., Yuan, Y., Aug. 2018. An Alternative View: when Does SGD Escape Local Minima? arXiv:1802.06175 [cs]ArXiv: 1802.06175. URL. http://arxiv.org/abs/1802.06175.
Laloy, E., Hérault, R., Jacques, D., Linde, N., Jan. 2018. Training-image based geostatistical inversion using a spatial generative adversarial neural network. Water Resour. Res. 54 (1), 381–406. https://doi.org/10.1002/2017WR022148. URL.
Laloy, E., Hérault, R., Lee, J., Jacques, D., Linde, N., Dec. 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. URL. https://linkinghub.elsevier.com/retrieve/pii/S0309170817306243.
Laloy, E., Linde, N., Ruffino, C., Hérault, R., Gasso, G., Jacques, D., Dec. 2019. Gradient-based deterministic inversion of geophysical data with generative adversarial networks: is it feasible? Comput. Geosci. 133, 104333. URL. https://linkinghub.elsevier.com/retrieve/pii/S009830041831207X.
Lange, K., Frydendall, J., Cordua, K.S., Hansen, T.M., Melnikova, Y., Mosegaard, K., Oct. 2012. A frequency matching method: solving inverse problems by use of geologically realistic prior information. Math. Geosci. 44 (7), 783–803. URL. http://link.springer.com/10.1007/s11004-012-9417-2.
Linde, N., Renard, P., Mukerji, T., Caers, J., Dec. 2015. Geological realism in hydrogeological and geophysical inverse modeling: a review. Adv. Water Resour. 86, 86–101.
Liu, N., Oliver, D.S., Jun. 2005. Ensemble Kalman filter for automatic history matching of geologic facies. J. Petrol. Sci. Eng. 47 (3), 147–161. URL. https://www.sciencedirect.com/science/article/pii/S0920410505000550.
Luo, X., Stordal, A.S., Lorentzen, R.J., Nævdal, G., Oct. 2015. Iterative ensemble smoother as an approximate solution to a regularized minimum-average-cost problem: theory and applications, 05 SPE J. 20, 962–982. https://doi.org/10.2118/176023-PA. URL.
Mariethoz, G., Renard, P., Straubhaar, J., Nov. 2010. The Direct Sampling method to perform multiple-point geostatistical simulations: performing multiple-points simulations. Water Resour. Res. 46 (11) https://doi.org/10.1029/2008WR007621. URL.
Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J., May 2017. Unrolled Generative Adversarial Networks arXiv:1611.02163 [cs, stat]ArXiv: 1611.02163. URL. http://arxiv.org/abs/1611.02163.
Mo, S., Zabaras, N., Shi, X., Wu, J., Feb, 2020. Integration of adversarial autoencoders with residual dense convolutional networks for estimation of non-Gaussian hydraulic conductivities. Water Resour. Res. 56 (2). URL. https://onlinelibrary.wiley.com/doi/abs/10.1029/2019WR026082.
Mosser, L., Dubrule, O., Blunt, M.J., Jun. 2018. Stochastic Seismic Waveform Inversion Using Generative Adversarial Networks as a Geological Prior arXiv:1806.03720 [physics, stat]ArXiv: 1806.03720. URL. http://arxiv.org/abs/1806.03720.
Naitzat, G., Zhitnikov, A., Lim, L.-H., Apr. 2020. Topology of Deep Neural Networks arXiv:2004.06093 [cs, math, stat]ArXiv: 2004.06093. URL. http://arxiv.org/abs/2004.06093.
Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A., 2017. Automatic Differentiation in PyTorch, vol. 4.
Rezaee, H., Marcotte, D., Apr. 2018. Calibration of categorical simulations by evolutionary gradual deformation method. Comput. Geosci. 22 (2), 587–605. http://link.springer.com/10.1007/s10596-017-9711-7.
Richardson, A., Jun. 2018. Generative Adversarial Networks for Model Order Reduction in Seismic Full-Waveform Inversion arXiv:1806.00828 [physics]ArXiv: 1806.00828. URL. http://arxiv.org/abs/1806.00828.
Rolinek, M., Zietlow, D., Martius, G., Apr. 2019. Variational Autoencoders Pursue PCA Directions (By Accident) arXiv:1812.06775 [cs, stat]ArXiv: 1812.06775. URL. http://arxiv.org/abs/1812.06775.
Rücker, C., Günther, T., Wagner, F.M., Dec. 2017. pyGIMLi: an open-source library for modelling and inversion in geophysics. Comput. Geosci. 109, 106–123.
Salakhutdinov, R., Apr. 2015. Learning deep generative models. Annual Review of Statistics and Its Application 2 (1), 361–385. URL. http://www.annualreviews.org/doi/10.1146/annurev-statistics-010814-020120.
Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., Jun. 2016. Improved Techniques for Training GANs arXiv:1606.03498 [cs]ArXiv: 1606.03498. URL. http://arxiv.org/abs/1606.03498.
Seo, J.K., Kim, K.C., Jargal, A., Lee, K., Harrach, B., Jan. 2019. A learning-based method for solving ill-posed nonlinear inverse problems: a simulation study of lung EIT. SIAM J. Imag. Sci. 12 (3), 1275–1295. URL. https://epubs.siam.org/doi/10.1137/18M1222600.
Shao, H., Kumar, A., Fletcher, P.T., Nov. 2017. The Riemannian Geometry of Deep Generative Models arXiv:1711.08014 [cs, stat]ArXiv: 1711.08014. URL. http://arxiv.org/abs/1711.08014.
Smith, S.L., Le, Q.V., Feb. 2018. A Bayesian Perspective on Generalization and Stochastic Gradient Descent arXiv:1710.06451 [cs, stat]ArXiv: 1710.06451. URL. http://arxiv.org/abs/1710.06451.
Strebelle, S., 2002. Conditional simulation of complex geological structures using multiple-point statistics. Math. Geol. 34 (1), 1–21. URL. http://www.springerlink.com/index/8G2MEAGU5K0U07PK.pdf.
Tikhonov, A.N., Arsenin, V.I.A., 1977. Solutions of Ill-Posed Problems. Winston. URL. https://books.google.be/books?id=ECrvAAAAMAAJ.
Zahner, T., Lochbühler, T., Mariethoz, G., Linde, N., Feb. 2016. Image synthesis with graph cuts: a fast model proposal mechanism in probabilistic inversion. Geophys. J. Int. 204 (2), 1179–1190. URL. https://academic.oup.com/gji/article-lookup/doi/10.1093/gji/ggv517.
Zhang, C., Butepage, J., Kjellstrom, H., Mandt, S., Oct. 2018. Advances in Variational Inference arXiv:1711.05597 [cs, stat]ArXiv: 1711.05597. URL. http://arxiv.org/abs/1711.05597.