Geophysical Journal International

Geophys. J. Int. (2022) **230,** 916–931
Advance Access publication 2022 March 10
GJI General Geophysical Methods

https://doi.org/10.1093/gji/ggac100

Upscaling and downscaling Monte Carlo ensembles with generative models

Matthias Scheiter [1], Andrew Valentine [1,2] and Malcolm Sambridge [1]

[1] *Research School of Earth Sciences, Australian National University, Canberra, Acton ACT 0200, Australia. E-mail: matthias.scheiter@anu.edu.au*
[2] *Department of Earth Sciences, Durham University, Durham DH1 3LE, United Kingdom*

Accepted 2022 0in original form 2021 November 25

SUMMARY

Monte Carlo methods are widespread in geophysics and have proved to be powerful in non-linear inverse problems. However, they are associated with significant practical challenges, including long calculation times, large output ensembles of Earth models, and difficulties in the appraisal of the results. This paper addresses some of these challenges using generative models, a family of tools that have recently attracted much attention in the machine learning literature. Generative models can, in principle, learn a probability distribution from a set of given samples and also provide a means for rapid generation of new samples which follow that approximated distribution. These two features make them well suited for application to the outputs of Monte Carlo algorithms. In particular, training a generative model on the posterior distribution of a Bayesian inference problem provides two main possibilities. First, the number of parameters in the generative model is much smaller than the number of values stored in the ensemble, leading to large compression rates. Secondly, once trained, the generative model can be used to draw any number of samples, thereby eliminating the dependence on an often large and unwieldy ensemble. These advantages pave new pathways for the use of Monte Carlo ensembles, including improved storage and communication of the results, enhanced calculation of numerical integrals, and the potential for convergence assessment of the Monte Carlo procedure. Here, these concepts are initially demonstrated using a simple synthetic example that scales into higher dimensions. They are then applied to a large ensemble of shear wave velocity models of the core–mantle boundary, recently produced in a Monte Carlo study. These examples demonstrate the effectiveness of using generative models to approximate posterior ensembles, and indicate directions to address various challenges in Monte Carlo inversion.

Key words: Inverse theory; Neural networks, fuzzy logic; Probability distributions; Statistical methods; Machine learning.

1 INTRODUCTION

In geophysics, we are primarily interested in the structure and processes of Earth's interior, from shallow ore deposits in the crust to processes taking place in the mantle and the structure of Earth's core. However, almost all observations are made with instruments at Earth's surface, or satellites above it. As a consequence, available constraints on a particular study object are of indirect nature, leading to challenges when building models of Earth's interior from sparse and incomplete surface observations. For a particular set of observations, often a large number of possible Earth models are equally well suited. Also, model parameters are often correlated so that care has to be taken when exploring the parameter space, for example to obtain statistically independent samples in a probabilistic setting.

Due to these common characteristics of non-uniqueness and trade-offs it is often not meaningful to construct a single solution to an inverse problem, but a wide range of solutions should be taken into account. One route to this comes from ensemble inversion, where a large set of samples is built which together characterize the uncertainty and non-uniqueness of the model properties. A popular class of methods are Monte Carlo algorithms, and the resulting ensemble is usually interpreted in terms of a probability density function (for an overview see, e.g. Sambridge & Mosegaard 2002). Recent applications of Monte Carlo inverse methods in geophysics are diverse and include, among many others, the estimation of Curie depth from different types of observations (Mather & Fullea 2019), inversion for fault slip parameters (Hallo & Gallovič 2020), a framework to invert for mantle conditions from mafic rock samples (Oliveira *et al.* 2021), and the creation of a high-resolution

shear-velocity map of the core–mantle boundary (Mousavi *et al.* 2021).

After constructing a solution to the inverse problem, a final ensemble may contain of the order of 10^6 to 10^7 individual models, and this may be only a subset of the total number of models tested. The computational cost associated with each model may be high, so that it is not uncommon for computations to take several months (e.g. Mousavi *et al.* 2021), making Monte Carlo methods infamous for the high computational effort they require. The large number of models in an ensemble, combined with a high-dimensional parameter space (model dimensions of order 10^2 to 10^3 are common) leads to a huge volume of digital outputs that must be stored, interrogated and manipulated. Despite this high volume, there are often still applications where an even larger ensemble would be desirable. The employment of supercomputers can help to mitigate these challenges, but the necessary amount of samples often scales exponentially with model dimension so that new statistical approaches seem to offer more sustainable solutions.

Another common issue involves the interpretation of the solution. Given the challenges of accessing and working with a large ensemble, it can be difficult to fully appreciate the information it contains. Furthermore, publishing the complete solution, or sharing it with collaborators, may be infeasible. Many studies rely on a detailed analysis, for example exploiting information contained in covariance properties (Burdick & Lekić 2017; Rudolph *et al.* 2020) or the analysis of multimodal structures in the model parameters (Olugboji *et al.* 2017; Burdick & Lekić 2017). However, in many other studies only mean and variance of a model ensemble are reported, ignoring potentially important details of the higher-order information contained in the ensemble. Later studies are then unable to fully exploit or contest earlier results, and there is potential for misleading results.

In this study, we devise new pathways which can help overcome some of the challenges inherent to Monte Carlo ensembles. We leverage recent progress in the field of machine learning, and use a class of models known as generative models (e.g. Kingma & Welling 2014; Goodfellow *et al.* 2014; Sohl-Dickstein *et al.* 2015; Rezende & Mohamed 2016). These are able to capture the intrinsic distribution of a large collection of samples, and are therefore well-suited for the use in combination with outputs from Monte Carlo inversion. We focus on the situation where a model ensemble has already been obtained, and outline different ways in which generative models may be applied to this ensemble in order to improve its handling, interpretability and usage in subsequent studies.

The methodology presented can be seen as an 'add-on' rather than replacement of existing sampling strategies, and is compatible with any Monte Carlo algorithm. It enables the information contained within a model ensemble to be represented in an efficient form. Rather than storing the individual realizations that comprise the ensemble, a generative model aims to capture its underlying statistical properties and enable the generation of new sets of samples that are indistinguishable from the original. Storage requirements for this are a fraction of those for the full ensemble, allowing easy sharing with collaborators and the community. Subsequent studies can then exploit the full information content of the ensemble, rather than relying on low-order approximations, and the ability to generate new samples can help to minimize sample size errors in evaluation of integrals such as mean, marginals, and covariances. In addition, we propose ways to analyse and interpret the results from Monte Carlo inversion in a more meaningful manner, better-exploiting the vast amount of information they contain.

We begin the paper in Section 2 with a motivating example, a brief introduction to generative models, and an overview on previous applications in geophysics. Then, our main concepts are illustrated with a synthetic example in Section 3, including a demonstration of enhanced calculation of numerical integrals, which are common in Monte Carlo inversion. In Section 4, we apply a generative model to a real geophysical ensemble from a previous study, showing the applicability to large ensembles. In Section 5, we discuss the main contributions of our approach and outline potential future directions. We conclude the study by summarizing our main findings in Section 6.

2 METHODOLOGY

2.1 A motivating example

Consider the following situation: we have obtained samples from a certain distribution, but for some reason, the number of samples we could collect was limited. This is a common problem, especially when the generation of each sample is expensive, for example rock samples from a remote study area or outputs from a complex computer code. This leads to a sparsely sampled distribution, which may then create challenges for further analysis. An example is shown in Fig. 1(a), where 30 samples have been obtained that are distributed according to the displayed histogram.

It is common in such a situation to calculate mean, mode or median along with the variance for an estimate of location and

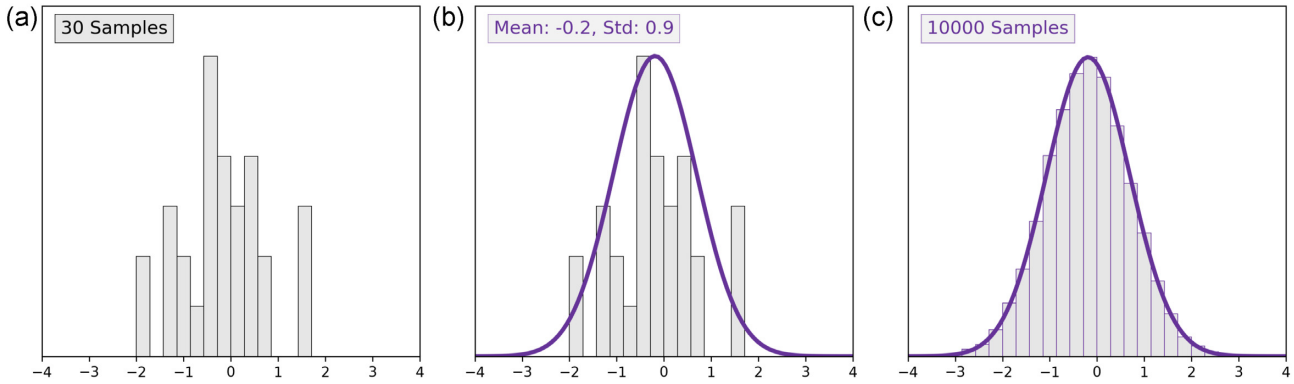

Figure 1. A simplified Gaussian example showing the capabilities of generative models to compress and enhance ensembles. (a) A set of 30 observed samples; (b) Gaussian approximation to observed samples; (c) 10 000 samples drawn from the Gaussian distribution. Two values (mean and standard deviation) are sufficient to store the original information and then generate any amount of samples with the same approximate characteristics.

spread of the samples. Hence, we can reduce the information in our 30 samples down to only two numbers, which from a compression perspective reduces the amount of data by 93 per cent. If we wished, we could go a step further and approximate the data set with a Gaussian distribution, which is shown in Fig. 1(b) together with its mean and standard deviation.

This Gaussian approximation can not only be used to describe the data, but also defines a probability distribution from which samples can be drawn. It can therefore be seen as a simple generative model. In Fig. 1(c), 10 000 samples were drawn, yielding a smooth histogram that approximates the statistical properties of the original ensemble. The ability to generate such samples may be advantageous: for example, subsequent modelling or analysis may depend on a high sample density for stability.

This simple example demonstrates some of the most important points that we aim to convey in this study: it is possible to find an appropriately chosen parametric distribution that captures the statistical properties of interest of a given ensemble of observations. This can then be used to generate more samples at low cost. In the present example, the data structure is very simple, so that a Gaussian approximation is sufficient. When the data are distributed in a more complicated manner, we can make use of more sophisticated generative models to achieve the same goal. This is the idea that the remainder of this study explores.

2.2 Generative models

A generative model is a tool that can generate random samples, used as a representation of some system or process. As we have seen, simple data sets may be adequately represented as a Gaussian distribution to capture the statistical properties of interest; more complex situations require a more sophisticated approach. Just as Gaussian-distributed random numbers can be generated by applying the Box–Muller transform (Box & Muller 1958) to uniform random samples, we can represent arbitrary distributions as transformations of simple ones. In particular, this suggests approaches that borrow from the heritage of machine learning and neural networks, leading to many varieties of generative models such as variational autoencoders (Kingma & Welling 2014), generative adversarial networks (GANs; Goodfellow *et al.* 2014), diffusion models (Sohl-Dickstein *et al.* 2015) and flow-based models (Rezende & Mohamed 2016).

GANs are one of the most popular types of generative models, and they consist of two neural networks, a 'generator' and a 'discriminator'. Each of these networks has its own task: the generator aims to produce samples ('fake data') that look as similar as possible to a given set of samples from the target distribution ('real data'); the discriminator's goal is to distinguish real from fake data. The optimization ('training') of the internal network parameters is done in a competitive manner where each network seeks to outperform the other. If training is successful, the output from the generator is indistinguishable from the training data and samples from the generator can be used as if they came from the target distribution.

In this study, we adopt GANs as our representative of generative models and use different variants of them in the examples below. However, nothing in our approach is specific to GANs, and other classes of generative model could also be used. A more comprehensive introduction to GANs is given in Appendix A, and the exact settings of all used GANs are summarized and discussed in Appendix B.

2.2.1 Previous applications in geophysics

In geophysics, generative models have previously been used in a variety of settings. Among the first is the study by Li *et al.* (2018) who used the discriminator of a GAN for earthquake detection in an earthquake early warning context. Florez *et al.* (2020) trained a Wasserstein GAN on a large data set of accelerograms, allowing for generation of synthetic accelerograms in an engineering application. Other applications include the generation of seismic signals from volcanic eruptions (Grijalva *et al.* 2021), seismic data interpolation (Oliveira *et al.* 2018), seismic data reconstruction (Siahkoohi *et al.* 2018) and first arrival picking (Zhang & Sheng 2020). Henriques *et al.* (2021) used variational autoencoders and normalizing flows to augment data in an application related to the imaging of salt bodies.

A number of previous studies have also used generative models in a geophysical inversion context. Araya-Polo *et al.* (2019) used GANs to augment existing seismic data, which was then used to drive neural network-based tomography. Another example is the application of a cycle-GAN to map between geological structure and seismic wavefield, enabling fast forward modelling and fast inversion (Mosser *et al.* 2018). Several studies have attempted to perform an inversion in the low-dimensional latent space of a generative model rather than in the more complex physical model space, both in deterministic, gradient-based inversion (Laloy *et al.* 2019; Lopez-Alvis *et al.* 2021) and in probabilistic, sampling-based settings (Laloy *et al.* 2018; Mosser *et al.* 2020). Zhang & Curtis (2021) included a latent space into invertible neural networks which after training allows probabilistic sampling from the posterior ensemble, which can effectively be seen as a generative model. In active seismics, normalizing flows have been used for imaging purposes (Siahkoohi & Herrmann 2021). Moment tensor inversion has been performed with an ensemble of Bayesian neural networks (Steinberg *et al.* 2021). From a viewpoint of variational inference, Zhao *et al.* (2022) used normalizing flows in an attempt to replace McMC algorithms by more efficient methods.

All these previous studies applied generative models to data analysis or directly in inversion applications. To our knowledge, this study is the first that focuses on an application of generative models to an existing ensemble of Earth models, for the purpose of making its handling less cumbersome and analysis more practical.

3 SYNTHETIC EXAMPLE

3.1 A non-Gaussian example

In this section, we extend the motivating example from Section 2.1 to a non-Gaussian case. We adopt an example used by Käufl *et al.* (2016), where the governing probability density function of an ensemble has all samples at the same distance $d = 0.7$ from the origin, with added Gaussian noise $\sigma = 0.1$. This leads to an ensemble forming a ring around the origin in two dimensions and a (hyper-)sphere in three (or more) dimensions. We start with the 2-D case in which the ensemble is distributed in the shape of a circle, and an McMC-generated ensemble of 100 000 samples is shown in the top right panel in Fig. 2, together with its marginals in each coordinate.

Due to its multimodality and the coupling between dimensions, this distribution can not be approximated by a (multivariate) normal distribution in Cartesian coordinates, but is well suited to use with a generative model such as a GAN. With sufficient samples, training a GAN on this ensemble gives a good representation of the circular distribution, including its marginals (see bottom right panel of

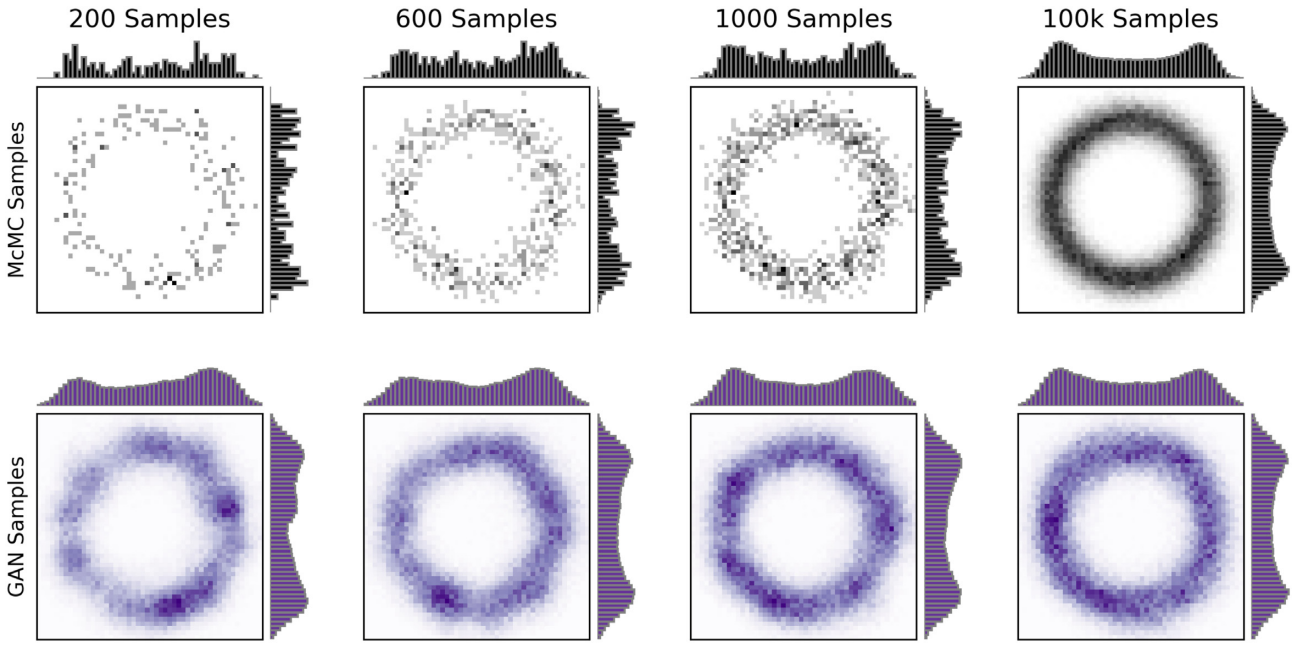

Figure 2. A non-Gaussian example. Ensembles of different sizes (black) are used as training data for GANs which aim to reproduce them (purple). GANs are able to reproduce arbitrary ensembles and to approximate the governing probability distribution from a finite number of samples.

Fig. 2). As the full ensemble has 100 000 2-D points, 200 000 numbers need to be stored for the full ensemble. After training the GAN, the same information can be stored in 10 252 trainable parameters of the generator, leading to a compression rate of 95 per cent (refer to Appendix B1 for details on the network architecture). While the ring might be expressed by a smaller set of parameters in a suitable basis (e.g. a Gaussian in radius), we note that GANs do not assume any particular parametric form for the distribution and are, in principle, agnostic to the underlying structure of the data they are trained on. They are therefore more generally applicable and the example presented here might easily be exchanged by one without a simple human interpretation.

Similar to the first example from Section 2.1, we can also use the GAN to enhance an undersampled distribution. The first three panels of Fig. 2 show reconstruction based on a small fraction of the same ensemble. Large gaps are seen in the joint distribution and the marginals are represented unsatisfactorily. Again, we can train a GAN to generate new samples for each of these cases, leading to the results shown in the first three panels of the second row of Fig. 2. In all cases, the new samples are consistent with the larger ensemble with some artefacts appearing for the smallest training set. In this experiment, we added uncorrelated Gaussian noise with a standard deviation of 0.1 to the training samples. While this leads to some smoothing of the obtained circle, it is an effective and common means to augment the training data and decrease the potential for overfitting (e.g. Bishop 2006).

3.2 GAN-enhanced numerical integration

In many applications, samples are collected to be used for further analysis. This often involves mapping the samples into a different domain (e.g. by conducting numerical or physical testing on each), and the accuracy or interpretability of results may depend on the density of the original set of samples. For example, an ensemble might be sufficient to represent the overall structure of a function, but might need to be combined with another function of higher complexity. A common example of this class of problem is the task of numerical integration. To illustrate the problem, we investigate numerical integrals of the general form

$$I = \int p(x) f(x) \mathrm{d}x, \tag{1}$$

where $f(x)$ is a function that should be integrated, weighted by a probability distribution $p(x)$. It is worth recalling that the calculation of all properties of interest for Monte Carlo ensembles can be cast in this form, for example mean, marginals and covariances (e.g. Sambridge & Mosegaard 2002). We assume that the distribution $p(x)$ is known only in the form of an ensemble of samples. It is therefore necessary to approximate integrals as in eq. (1) with Monte Carlo integration. Typically, this involves a sum

$$I \approx \frac{1}{N} \sum_{x_i \in \mathcal{X}} f(x_i), \tag{2}$$

where $\mathcal{X}$ is a set of N random samples that are distributed according to $p(x)$. The accuracy of this approximation improves as N increases, and is governed by the complexity of both p and f. One can therefore readily envisage situations where the structure of p can be captured using a modest number of samples, which proves insufficient for accurate evaluation of the integral.

In this case, we propose that a generative model can be constructed to mimic the distribution $p(x)$. This can then be used to generate the ensemble $\mathcal{X}$, with as many samples as are required to adequately capture the features of $f(x)$. In cases where sampling from p is expensive or otherwise difficult, this may be a highly effective approach.

In the following example, the probability distribution from the previous section acts as $p(x)$, and the function to be integrated is defined as

$$f(\varphi) = \sin(\omega\varphi), \qquad \omega = 0.5, 1.5, 2.5, \ldots, \tag{3}$$

where $\varphi = \arctan(\frac{x}{y})$ is the azimuth of a sample on the circle and ω is a frequency of the sine curve. The upper three panels in Fig. 3 show

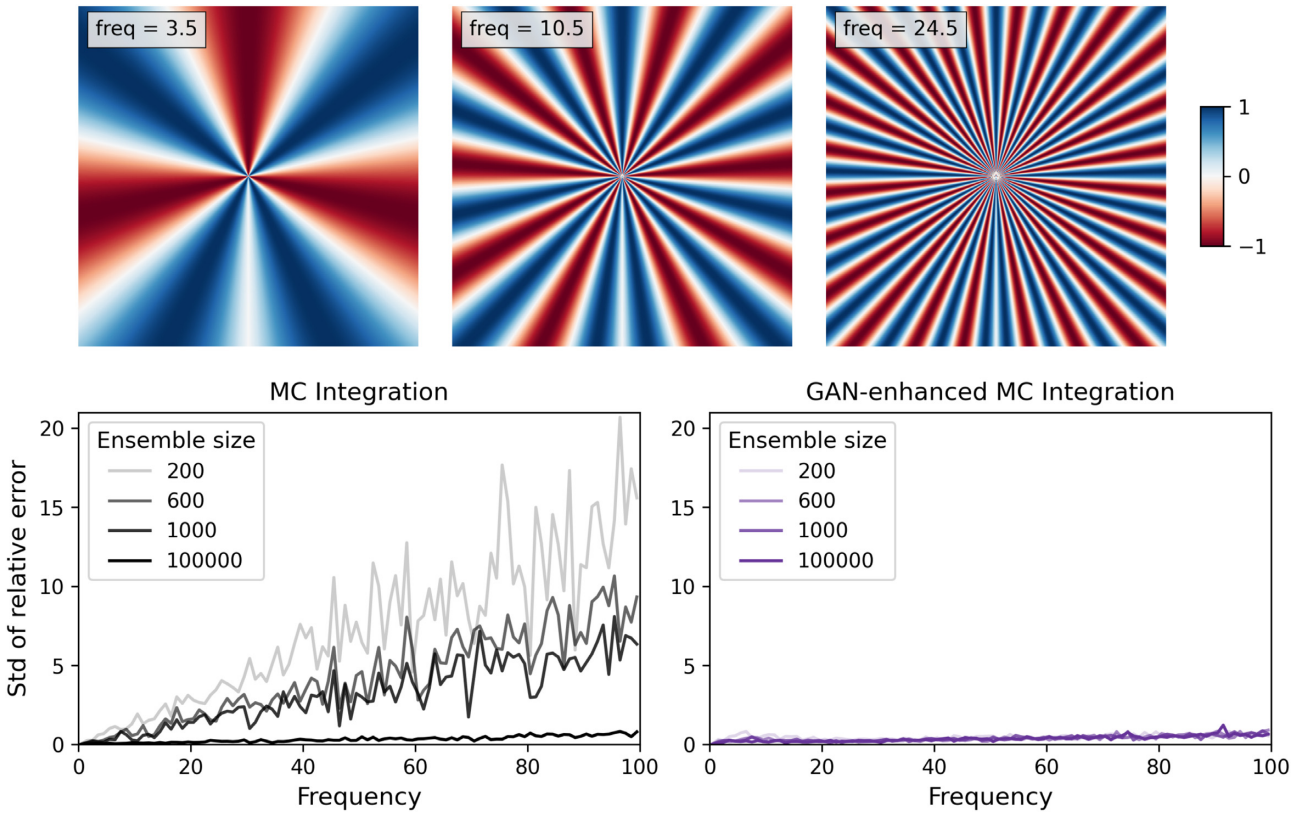

Figure 3. Example for improved solution of numerical integration with generative models. Top row: three examples with different frequencies of the function that is integrated. Bottom left-hand panel: relative error of the integral for the ensembles from Fig. 2. Bottom right-hand panel: relative error when training a GAN on each ensemble and then drawing a large amount of samples.

three examples of $f(x)$ with different angular frequencies of 3.5, 10.5 and 24.5. The complexity of the function increases for higher frequencies, where it becomes increasingly important to perform the integration with a larger ensemble.

The lower left-hand panel in Fig. 3 shows the results when performing the integration with the original Monte Carlo ensembles of 200, 600, 1000 and 100 000 samples that were investigated in the previous section. The integral of $p(x)f(\varphi)$ is then computed for each frequency between 0.5 and 99.5, and the relative error with respect to the analytical solution

$$I_\omega = \frac{2}{\omega} \tag{4}$$

is calculated. This is repeated for 10 different independent chains, and the standard deviation of the relative error across the chains is calculated for each frequency.

The results show that for the limited ensembles (200, 600, 1000 samples) the error increases rapidly with the complexity of the function (i.e. the sine frequency). A similar tendency can be observed for a large, densely sampled ensemble of 100 000 samples, but the overall accuracy is much better.

Next, a GAN was trained on each representation of the circular distribution, leading to 40 GANs in total (four ensemble sizes with 10 chains each). From each of these GANs, 100 000 samples were drawn and used to compute the value of the integrals across frequencies. As the lower right-hand panel in Fig. 3 shows, the accuracy of all these experiments is comparable to the large original ensemble. This means that even the ensemble with only 200 samples can give excellent results in numerical integration for the highest frequency case by training a GAN on it. In this case, 100 000 samples were drawn from the GANs, and the cost of GAN samples are much lower than obtaining them from Monte Carlo sampling of the distribution $p(x)$ or $p(x)f(x)$.

3.3 The curse of dimensionality

Extending the example problem to higher dimensions, we obtain hyperspheres, which are shown in the first row of Fig. 4 for 2, 3, 4, 7 and 10 dimensions, projected into the x_1–x_2 plane. While in two dimensions this gives the ring from the previous example, for higher dimensions the samples are increasingly focused in the centre. This effect is a consequence of the well-known curse of dimensionality (e.g. Curtis & Lomax 2001).

The challenges that come with the curse of dimensionality are clearly visible in the second row of Fig. 4, which shows only those samples that are close ($d < 0.15$) to the x_1–x_2 plane. The number of points contained in this plane rapidly decreases in higher dimensions. If we wish to densely sample this plane of a high-dimensional hypersphere, it quickly becomes infeasible, and practitioners are often limited to sparse ensembles.

Applying the upscaling approach, we train a GAN on each of the ensembles (for details see Appendix B1). From this, GAN samples can be drawn at limited cost, improving the ability to obtain a densely sampled ensemble, as shown in the third row of Fig. 4. In the 10-D case, 2 billion samples had to be drawn to obtain 300 000 samples close to the plane. This took around 46 min once the network was trained, a dramatic reduction compared to the 120 hr that would be required to obtain the same amount of samples from McMC sampling (all calculations were performed on a 3.7 GHz Intel Core

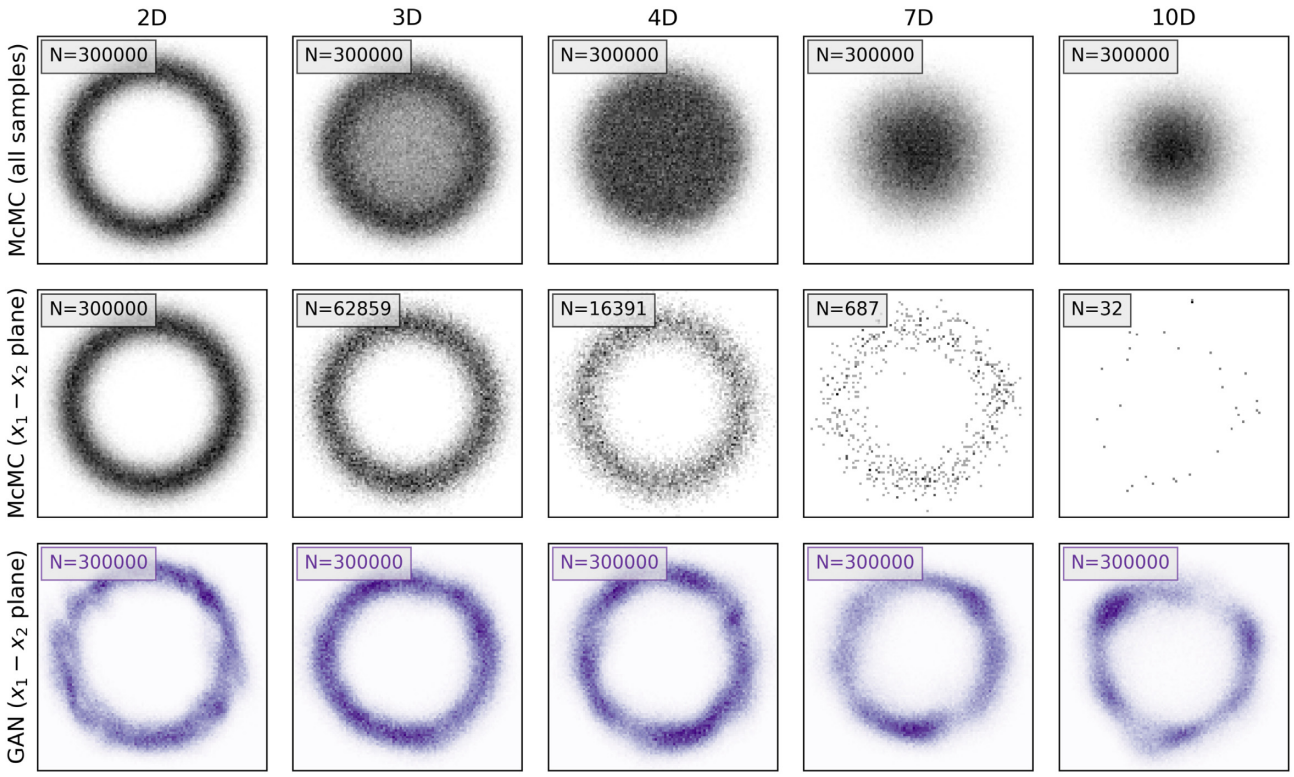

Figure 4. First row: samples on hyperspheres of different dimensionalities. Second row: only those samples that are close ($d < 0.15$) to the x_1–x_2 plane. Third row: GAN samples in the x_1–x_2 plane. A GAN can significantly increase the sample density based on a sparse ensemble.

i5 processor with 16 GB 2667 MHz DDR4 RAM). Therefore, our approach offers a strategy to deal with the curse of dimensionality and enhance an ensemble if the available samples represent the structure of the underlying distribution well, and if the GAN is able to represent that structure in terms of the statistical properties of interest.

3.4 Discussion

These experiments on a synthetic example illustrate all the major concepts we aim to convey in this study, and it is worth summarizing them before moving on to a real geophysical example. The tests in Section 3.1 have demonstrated that it is possible to use generative models to (i) compress a large and complex ensemble into a neural network of much smaller size; and (ii) to enhance a small ensemble to a larger size in cases where the available samples contain all significant information on the ensemble structure. While (i) can be useful in storing and communicating the results without depending on the original large and static ensemble, (ii) leads to the possibility of obtaining larger ensembles without additional sampling, at low cost once the network is trained.

This larger ensemble can then be used for subsequent calculations, a common example of which is the calculation of numerical integrals which may depend strongly on the number of available samples. This example has been demonstrated successfully in Section 3.2, where training a generative model on an available sparse collection of samples improved the quality of the resulting integrals significantly. Another related aspect that might be useful to practitioners is shown in Section 3.3, where it was demonstrated that some challenges associated with the curse of dimensionality can be overcome by enhancing the ensemble size of an original, sparse ensemble.

These concepts are further examined and discussed in the following sections, where a real ensemble from a geophysical inversion study is learned with a generative model. This example comes with further challenges: it has a much higher dimensionality, the distribution is more irregular, the digital volume of the ensemble is larger, and consequently training the generative model becomes more challenging and cost-intensive. Nevertheless, the results show that all the concepts introduced in the synthetic example can also be applied to this larger example, qualitatively demonstrating its applicability to real geophysical inversion studies.

4 GEOPHYSICAL EXAMPLE

4.1 A shear velocity model from the core–mantle boundary

To demonstrate these concepts on a high-dimensional geophysical problem, we apply the same methodology to an ensemble of a 2-D shear wave velocities at the core–mantle boundary, generated by a trans-dimensional Bayesian sampling algorithm (Mousavi *et al.* 2021). The full ensemble consists of 54 million models across 45 chains, each of which is comprised of on average 600 spherical Voronoi cells. The Voronoi cells each have three unknowns, latitude and longitude of its defining nucleus together with the shear wave velocity, leading to about 97 billion parameter values in the full ensemble (for details, see Mousavi *et al.* 2021). This huge digital volume occupies approximately 2 TB of disk space. In practice, this volume is reduced through burn-in and thinning, leaving 180 000 models, with a total of 325 million parameter values in the final

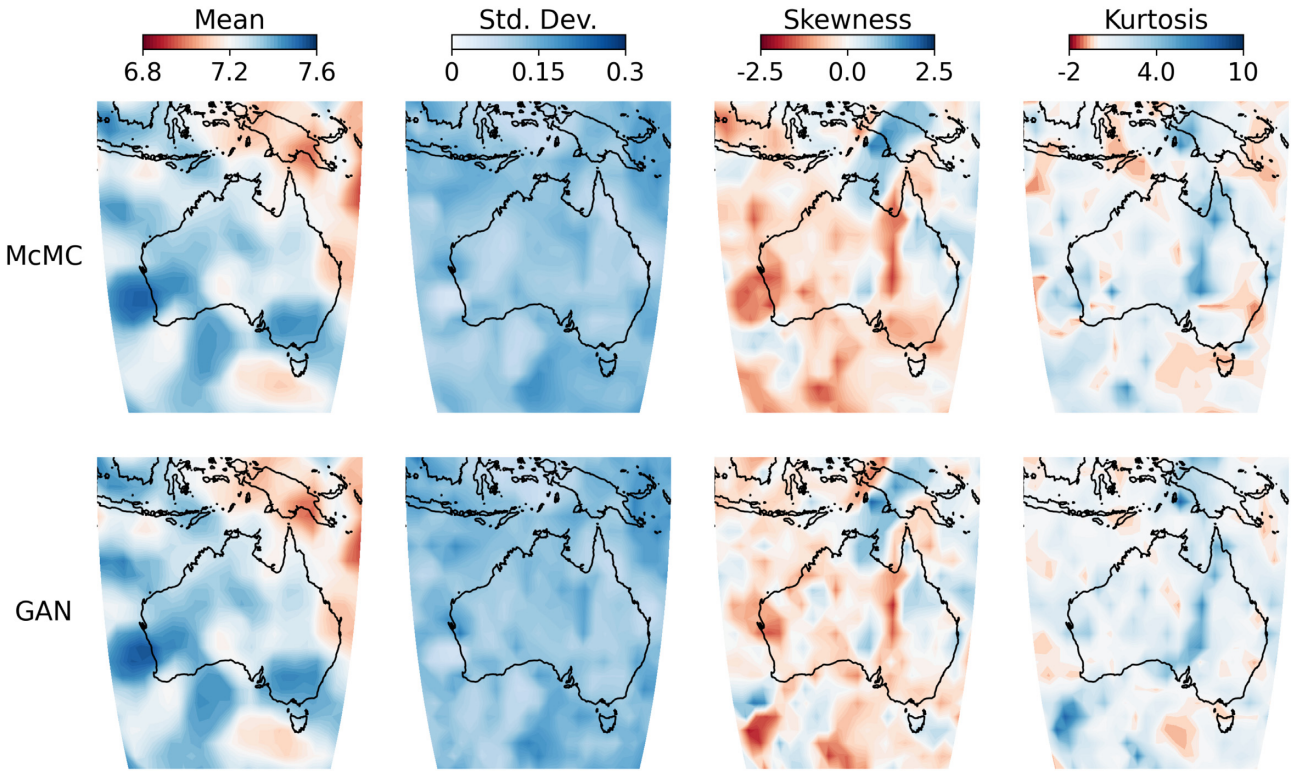

Figure 5. Comparison between the original McMC samples and an equal number sampled from the GAN. Displayed are the first four moments of the distribution. Overall, the GAN learns and then reproduces the original ensemble in great detail.

ensemble for analysis. Maps of mean and standard deviation of this ensemble can be seen in Fig. 9. In the study of Mousavi *et al.* (2021), the calculation of each chain took 28 000 CPU hours, amounting to a total of 1.3 million CPU hours for all 45 chains. This is equivalent to 150 yr computation on a single CPU, although parallelization reduces the computational cost to 110 d. This situation is typical, and illustrates the considerable computational effort required to create, store and analyse the outputs of McMC algorithms.

4.2 Reconstruction of a regional subset

In a first experiment, we select a subset of the global model ensemble covering the area around Australia. This consists of 23 × 23 points on regular grid of 50° × 50° (latitude × longitude), leading to a 529-dimensional ensemble with 180 000 models from the thinned chains. We train a deep convolutional GAN (DCGAN; Radford *et al.* 2015) on this ensemble. A discussion regarding the choice of a DCGAN can be found in Appendix B, and details of architecture and training are summarized in Appendix B1.

After training, we sample 180 000 models from the generator and compare it to the original ensemble of the same size. Fig. 5 compares the McMC (i.e. original) and GAN-generated ensembles, showing the first four moments: mean, standard deviation, skewness, and kurtosis. The mean maps correspond to the average shear wave velocity structure at the core–mantle boundary beneath Australia, and the McMC and GAN versions are very close to each other. The same holds for the standard deviation which is traditionally associated with the uncertainty of the velocity.

Often, studies indicate these first two moments as their main results in terms of velocity value and its uncertainty, implicitly making the assumption that the models are distributed in a Gaussian manner. We show that the GAN is able to incorporate information on the model ensemble beyond this Gaussian approximation, as can be seen in the maps of skewness and kurtosis, i.e. the third and fourth moment of the distribution, which are accurately reproduced by the GAN. The skewness indicates the shift towards the left or the right of a distribution with respect to a normal distribution, and the kurtosis is a measure of the 'bulkiness' or 'pointiness' of a distribution.

The trends seen in the plots for skewness and kurtosis can be further examined when looking at the marginals directly. Fig. 6(c) compares the marginals for McMC and GAN at 20 locations as indicated in the map in Fig. 6(a). It becomes clear from the intricate shapes of the marginals that the GAN is able to incorporate the most important features, more than would be possible with a Gaussian approximation. In terms of multimodalities, our results suggest that the GAN might in some cases not be able to capture them sufficiently. However, it is by no means clear that all multimodalities in the McMC ensemble are significant and physically meaningful. The GAN does not incorporate very small-scale (and likely noisy) variations in the histograms, which can be seen as an indication that it has been trained robustly against overfitting. Hence, there is a trade-off between over- and underfitting which needs to be carefully addressed when training the generative model.

Fig. 6(a) shows the earth mover distance (e.g. Monge 1781; Rubner *et al.* 2000) between the two distributions as a function of spatial position. This is a qualitative measure of misfit between the marginals of McMC and GAN, and shows an overall uniformly small difference confirming a reasonably accurate representation. Only at the margins does the quality drop, which could be expected as those locations are more weakly constrained in the DCGAN than points in the centre of the domain. Note that here we are comparing

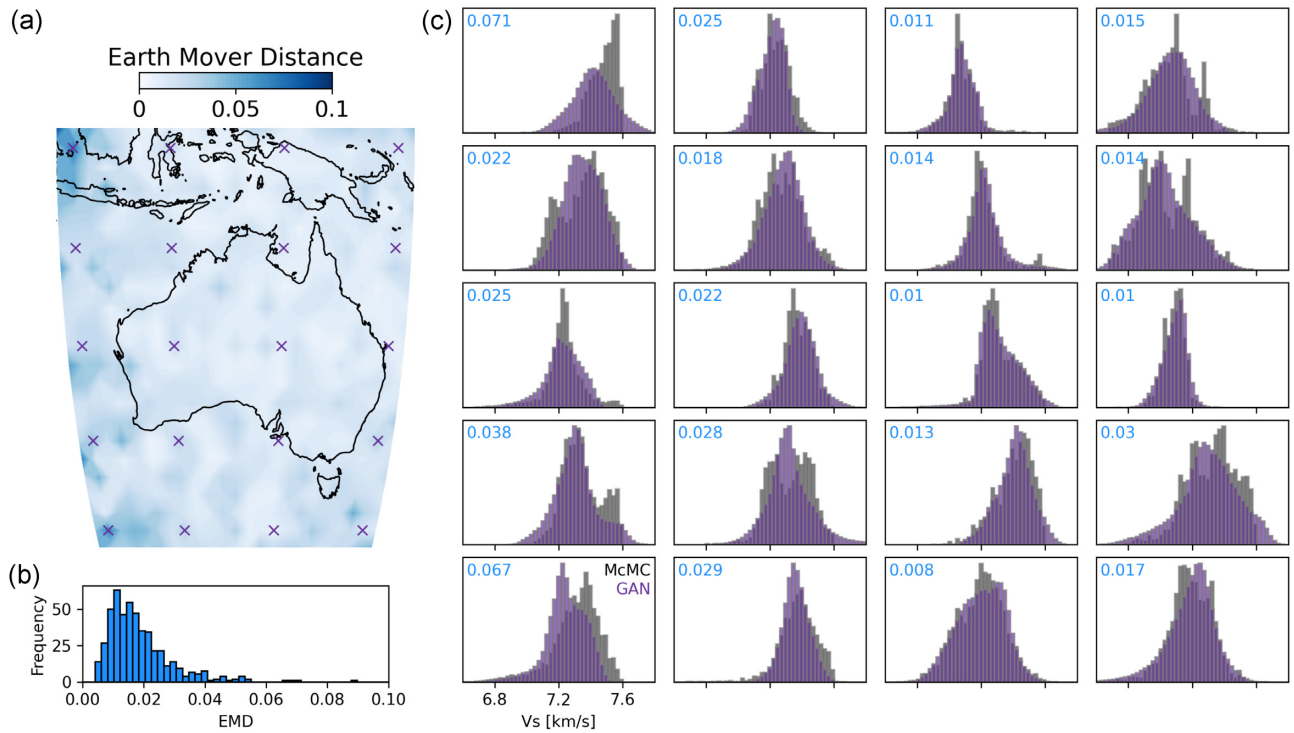

Figure 6. (a) Earth mover distance for individual points on the Australian subset, comparing the deviation of GAN marginals from the original McMC marginals. (b) Histogram of all earth mover distances (EMD). (c) Selected marginals for comparison, locations indicated by purple crosses in panel a; black: McMC marginals; purple: GAN marginals; blue: earth mover distance at locations. The great majority of marginals are well recovered by the GAN.

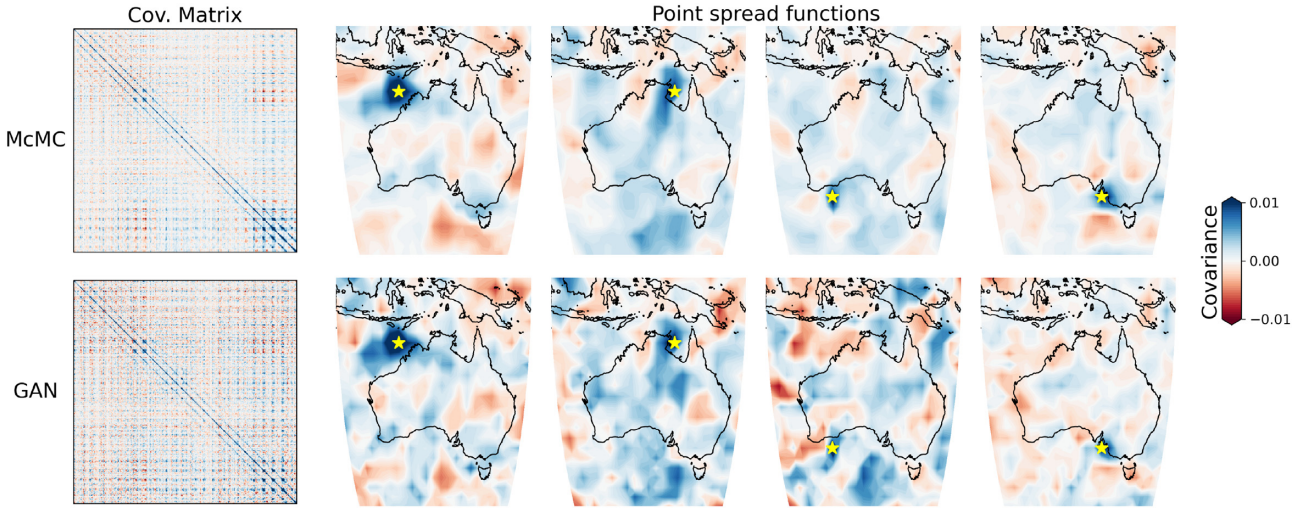

Figure 7. Covariance matrices for McMC and GAN ensembles together with point spread functions of four selected locations (shown as yellow markers).

relative values of the earth mover distances contained in our results, with values below 0.01 indicating a close-to-perfect recovery without overfitting (see marginals in Fig. 6).

Additionally, the earth mover distances of all locations are shown in the histogram in Fig. 6(b), with most values being at the smaller end of the range and only few locations having higher values. The fact that the median is low and the histogram decays in a convex manner towards higher values is an indication of good convergence of the GAN. It can be expected that in initial training stages the histogram has a more concave shape, but as GAN training proceeds it becomes increasingly difficult to further minimize the bulk of earth mover distances.

The covariance matrix represents the statistical correlations between different locations in the model ensemble, and we compare the McMC and GAN versions in Fig. 7. The GAN covariance matrix is an excellent reproduction of that produced by McMC, leading to the ability to sample models that not only follow the correct pattern across the ensemble, but also include spatial relationships within one particular model. This is also the case for local covariance structures as can be appreciated in the point-spread functions in Fig. 7 which show selected individual rows of the covariance matrix as a map. The good reproduction of these maps by the GAN suggests that models drawn from it have the ability to recover correlation structure over the model domain with respect to one specific

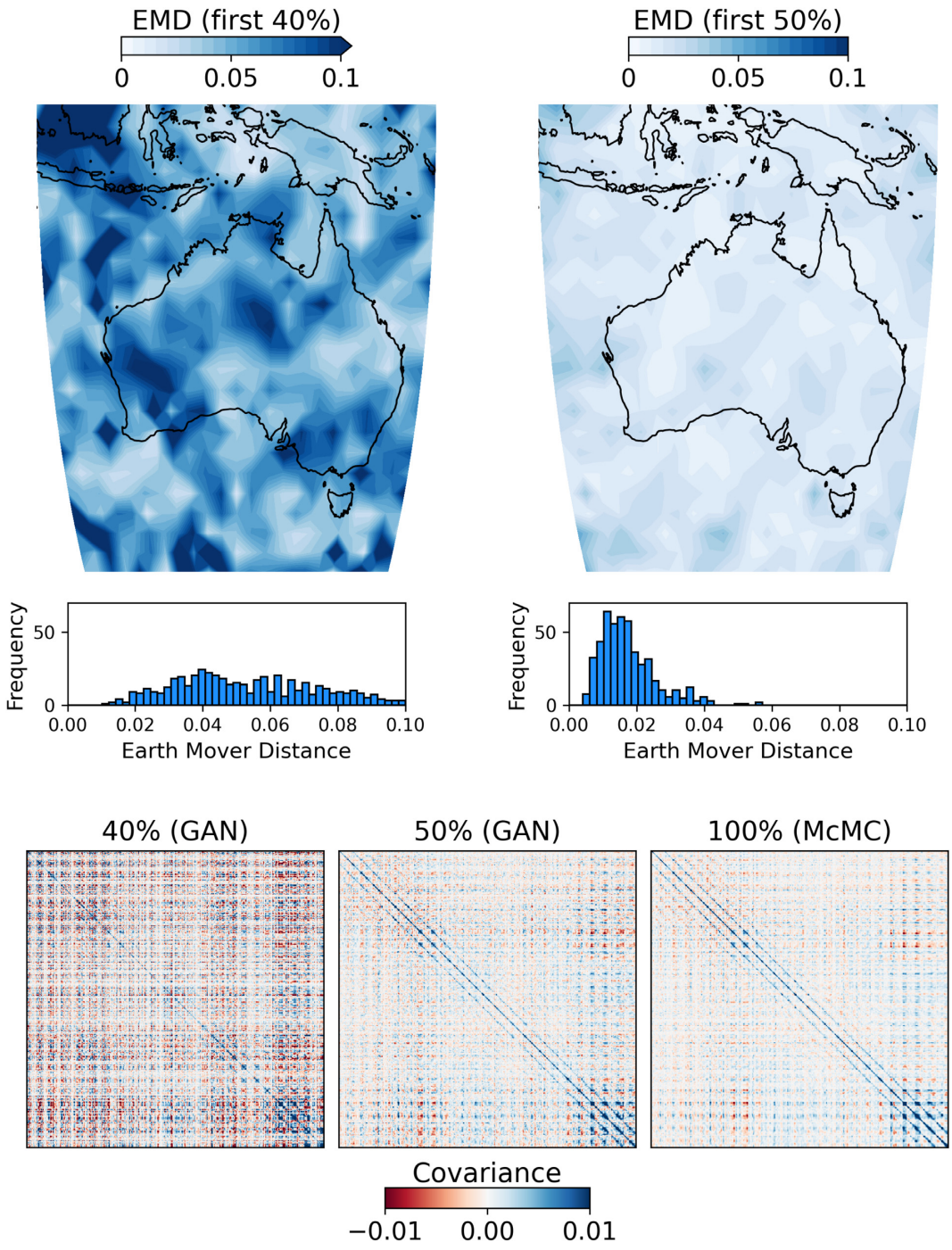

Figure 8. Training the GAN on limited amounts of samples (40 and 50 per cent) from the McMC ensemble. The generated marginals are compared to those from the full ensemble. With only 50 per cent of the samples, the GAN learns to reproduce the ensemble in the same way as with all samples. EMD, earth mover distance.

point (shown as a yellow stars in the maps). Maps such as these have previously been used to interpret ensemble-based inversion results (e.g. Burdick & Lekić 2017), and our results suggest that such analyses could also be performed with an approximation from a generative model.

These comparisons indicate that it is not sufficient to simply store the mean and standard deviation of an McMC ensemble, as important structures both within and across models may get lost. A generative model such as a GAN is able to approximate a distribution with many of its intricacies and details. In this case, a generator defined by around 3.6 million parameters is able to capture an ensemble that otherwise would be described by 95 million values, which from a compression perspective is a reduction of 96 per cent.

4.3 Guiding the convergence assessment of McMC

The results in the previous section have shown that a GAN is able to learn the structure of a high-dimensional model ensemble in some detail. In this context, a relevant question is what happens if we train

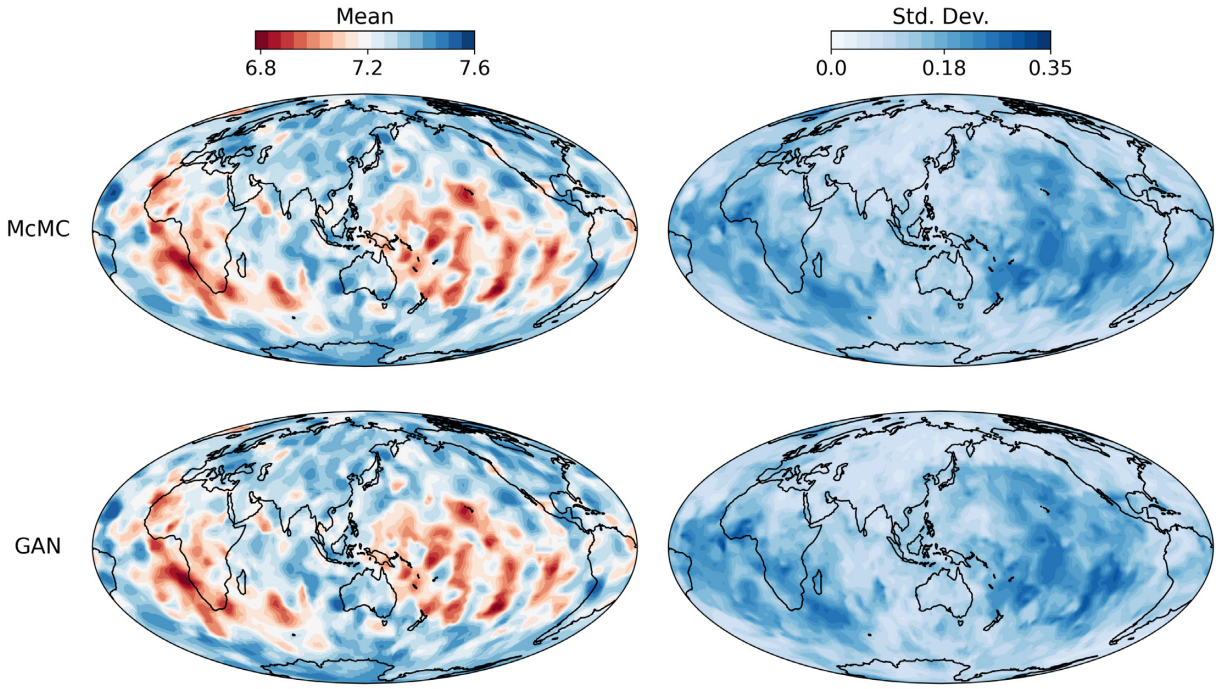

Figure 9. Maps of mean and standard deviation for the full model ensemble; comparison between McMC samples and GAN samples. The GANs can recreate the full velocity model ensemble from Mousavi *et al.* (2021) in high resolution.

the GAN on a reduced-sized ensemble, i.e. is it possible to obtain similar results from a fraction of the McMC ensemble?

To investigate this, we trained a GAN on the first 40 per cent and then the first 50 per cent of each of the 45 chains and compared the outputs to the full McMC ensemble. The earth mover distance between marginals is shown in the upper part of Fig. 8, comparable to Fig. 6(a). The left-hand panel is for the 40 per cent case and shows significant differences, indicating that the GAN trained on this number of samples may not contain sufficient information on the full ensemble to train a GAN. However, by taking another 10 per cent of the samples into account (right-hand panel), the situation changes significantly and is comparable to the one obtained when training the GAN with the full ensemble (see Fig. 6). The lower row of Fig. 8 shows the covariance matrices computed for the GAN ensembles, comparing them to that of the full McMC ensemble. Here, we see similar trends as when comparing the marginals: with the first 40 per cent of samples the covariance structures can not be fully recovered, whereas the version for 50 per cent of samples mimics well the covariance matrix of the full McMC ensemble.

This seems to suggest that there is a certain point where the McMC ensemble contains enough information so that more sampling does not change the quality of the GAN training. Another interpretation is that in the McMC procedure of Mousavi *et al.* (2021) the chains had reached equilibrium after around 50 per cent of the samples and additional sampling did not yield much significant information. This value of 50 per cent is highly likely to be problem-dependent only and may differ in other settings.

A practical outcome of these findings is the possibility that such an analysis might give 'on-the-fly' indications on when to stop the McMC sampling. By training a GAN, say, continuously alongside a McMC algorithm, it may be possible to decide precisely when to halt the Markov chain, which would be a significant improvement to existing practices of deciding on a fixed number of samples in advance. Such a judgement may be possible based on several indicators such as the marginals or the covariance properties of an ensemble.

4.4 Reconstruction of the global model ensemble

The results shown in the previous sections were only made for a subset of the global model ensemble. Here, we scale the approach to the global core–mantle boundary shear velocity model ensemble of Mousavi *et al.* (2021). First, we extract 8000 points that are approximately uniformly distributed on a sphere according to a deterministic spiral scheme (Koay 2011). The average distance between each pair of adjacent points is roughly 250 km, which is sufficient to capture structures up to a spherical harmonic degree of around 80 according to the Jeans relation (e.g. Wieczorek & Simons 2005).

To avoid having to handle 8000-dimensional vectors in the GAN training set, we divide the globe into 16 patches of approximately 500 points each, with areas similar to the region shown in Fig. 5 ($50^\circ \times 50^\circ$). On each of these patches, a Wasserstein GAN (WGAN; Arjovsky *et al.* 2017) is trained to reproduce the velocity models (see Appendix B1 for details). WGANs have better convergence characteristics than standard GANs, which becomes especially important in higher-dimensional settings.

From each WGAN, 180 000 samples are drawn for each patch and combined to obtain a global map shown in Fig. 9. The structure of both mean and standard deviation is well represented. The only drawback are inconsistencies along the boundaries of the patches, meaning that velocity correlations in these areas are not recovered. However, we are able to downscale the whole ensemble, described by 1.5 billion values, into 16 WGAN generators which together have 3.2 million parameters. This is equivalent to a compression rate of over 99 per cent, enabling straightforward sharing of the results and rapid sampling for subsequent applications for global models of this type.

5 DISCUSSION

5.1 Upscaling and downscaling model ensembles

The main contributions of our study fall in two categories of dealing with geophysical model ensembles: upscaling and downscaling. We have demonstrated approaches which address several well-known practical problems related to Monte Carlo inversion and in particular have shown how generative models can be useful. In this study, GANs have been used as our example of a generative model, but we note that all the discussed concepts hold for any type of generative model (see Appendix B for a technical discussion).

In a downscaling context, we have shown that it is possible to learn ensembles with a generative model which can then reproduce an ensemble in great detail. This has several advantages. First, it enables the user to compress a large model ensemble and store it in significantly less storage space. In our examples, the compression rates are around 95–99 per cent, suggesting that storage space can be reduced by at least two orders of magnitude with these techniques. While these compression rates are similar to those of typical video compression algorithms (e.g. Westwater & Furht 1997), our approach should not simply be regarded from a compression perspective, but in combination with its other benefits. If the goal is solely to compress a data set, established methods may be more suitable. In this context, it is worth mentioning that the emerging field of 'neural data compression' is strongly driven by generative models, providing potential alternatives to standard compression algorithms (e.g. Yang *et al.* 2022).

Secondly, the trained generative model can then be used as a generator of new samples that approximately follow the underlying distribution. This has particular advantages compared to distributing the original fixed ensemble. Our method facilitates the exchange and regeneration of a complete model ensemble at limited cost. In inversion studies with strong and significant multimodal distributions, for example near the boundaries of structures (e.g. Olugboji *et al.* 2017; Burdick & Lekić 2017), other types of generative models such as normalizing flows (e.g. Rezende & Mohamed 2016; Dinh *et al.* 2017) might be better able to reproduce these than GANs. This is because their objective function derived from variational inference is more directly targeted at the concrete shape of the distribution than the adversarial training of GANs which primarily aims at an equilibrium between the two networks.

The trained generative model has the capability to efficiently simulate new ensembles of any size. Of course, the samples drawn from the generative model will not be exactly the same ones as in the original ensemble, but, in the presented case, their statistical properties closely follow the original statistics (see Figs 5, 6, 7 and 9). This enables the use of our approach for studies that are interested in analysing the model ensemble in some detail (e.g. Olugboji *et al.* 2017; Burdick & Lekić 2017; Rudolph *et al.* 2020). The creation of these approximate ensembles takes in the order of hours to train the GANs (see Tables A1–A4) and in the order of minutes to draw samples. These times are insignificant compared to the build up times of Monte Carlo ensembles which typically are in the order of months (e.g. Mousavi *et al.* 2021). The concepts presented in this study are therefore available at negligible additional computational cost in the context of any large-scale Monte Carlo procedure.

Thirdly, the compressed version of the ensemble can be transmitted to collaborators in a straightforward way, as it is significantly smaller than the original ensemble. The recipient of the model ensemble can then explore and analyse it, and use it for their own subsequent studies. This will improve the ability for a clean processing pipeline, as it challenges the wide-spread habits of only sharing low-order versions of a model ensemble. As shown in Figs 5–7, there can be much more detail in the model ensemble which may get lost without such a generator. In this way one can better capture and exploit the huge effort that was necessary to build the model ensemble in the first place through Monte Carlo sampling.

The other main aspect is upscaling, which takes advantage of the fact that an effectively unlimited amount of samples can be drawn from a trained generative model. It is therefore possible to go beyond the ensemble size from the original Monte Carlo ensemble, which has significant implications. One of them is shown in Fig. 4, an example for cases where it might be computationally infeasible to draw sufficient samples from a high-dimensional distribution to obtain a smooth and dense ensemble. This direct consequence of the curse of dimensionality can be overcome by training a generative model, leading to a satisfactory sample density at a fraction of the computational time that would otherwise be required. In terms of error rate quantification between the original ensemble and the one drawn from the generative model, Figs 6 and 8 suggest that a measure such as the earth mover distance of marginals can give an indication of how well the original ensemble is approximated.

Another application of the upscaling aspect is when secondary analyses need to be performed based on an ensemble representing a probability distribution, for example moment calculation. Even if the probability distribution is well approximated by a given set of samples, the size of the ensemble might be insufficient if a moment-defining function has higher complexity and hence requires more samples. Generative models can help to enhance an ensemble in cases where the existing samples already cover the most significant aspects of the probability distribution. As shown with a generative model trained on only few samples, it is then possible in some cases to draw an effectively unlimited amount of samples, improving the accuracy of integrals estimated with limited computational expense.

The ability to draw a large amount of samples can also have beneficial consequences during the process of Monte Carlo sampling itself. The results presented in Fig. 8 suggest that in this specific case the first 50 per cent of the samples of each chain contain sufficient information to train a generative model whose outputs are very similar to the one trained on the full ensemble, whereas for 40 per cent of the samples the results are insufficient. This feature could be exploited to give an indication of a point in the sampling process where the Markov chain sampling should be stopped, which might significantly reduce computational costs. These costs could then be invested, for example in running more chains, which would improve the overall exploration of the model domain.

The exact point of convergence will depend on the specific ensemble and the 50 per cent obtained in this example should not be generalized. In the presented case, our results indicate that such a convergence criterion may be found based on different indicators from the ensembles such as information contained in the marginals or the covariance matrix. A convergence assessment based on the eigenvalues of the covariance matrix has previously been tested by Rudolph *et al.* (2020), which might also be possible with a GAN-based version of the ensemble. It is important to note that our proposed method is speculative in its current form, and further studies will need to address several questions which are beyond the scope of this paper. If further developed, the method seems to have some potential to become a viable alternative to one of the many existing diagnostics for McMC convergence (e.g. Gelman & Rubin 1992; Cowles & Carlin 1996; Roy 2020), with significant benefits from the nature of generative models as discussed in this study.

All concepts discussed here are not limited to geophysics, but have potential applications in other fields of the geosciences and beyond. Essentially, any situation where data are collected and interpreted in terms of a probability distribution can benefit from our methodology. It is a standard approach to make a Gaussian approximation (see Fig. 1), but the sampling aspect is not often considered. It might be helpful in order to provide a clean histogram or in cases where sampling for further applications is needed. When the structure in the data becomes non-Gaussian and higher-dimensional, a generative model may be useful. Potential applications include situations where sampling is expensive, carried out in remote places or time-consuming.

5.2 A way forward

In this study, we have focused on using generative models following application of Monte Carlo sampling. Therefore, our procedure does not include interactions between generative models and the McMC sampling process. We envision that such interactions can be possible in a multitude of ways, with both generative models and sampling process informing each other. The approach presented here might be viewed as an end-member case where the generative model is purely informed by McMC and does not feed back to the sampling process. Another end-member are studies such as that of Zhao *et al.* (2022) who replaced McMC entirely by normalizing flows. In between, there might be many possibilities to use generative models to aid the sampling procedure.

One scenario where a generative model can be trained to inform a Monte Carlo algorithm was briefly tested in Section 4.3. An extension to this approach might be to train a generative model after every certain amount of steps, and determine through some convergence criterion whether significant information has been added or if the new samples are redundant. Once the chains are deemed to have converged, the Monte Carlo sampling could be stopped and the generative model directly be used to provide the desired amount of samples. This scenario would therefore enable McMC early-stopping whilst making use of all concepts discussed in this study.

6 CONCLUSIONS

In this study, we used generative models to help overcome some of the main drawbacks of Monte Carlo methods in geophysical inversion. We showed that generative models can be used to approximately learn a density function consistent with the ensemble produced by a Monte Carlo algorithm (or more generally, any sampling approach). We suggest that this offers myriad practical benefits, allowing better use to be made of Monte Carlo results. Specific examples discussed include easier sharing and dissemination of results, and enhanced accuracy in calculations such as numerical integrals. Of course, the generative model is inherently an approximation to the target distribution, and in some circumstances this may be seen as a drawback. However, we suggest that even when analysis is ultimately to be performed using the original ensemble, it may be convenient to have access to a lightweight, portable representation that can be used during exploratory and development work. In addition, generative models show potential for assessing the convergence of McMC, and we expect these and other machine learning algorithms to have a significant impact on the way that probabilistic Monte Carlo inversion in geophysics is performed in the future. The methods proposed in this paper suggest some possibilities for how generative models and McMC can interact, and we hope that this work can stimulate further research in the area.

ACKNOWLEDGMENTS

We wish to thank Sima Mousavi for providing her model ensemble of the core–mantle boundary ahead of publication, and are grateful to Buse Turunçtur for many helpful discussions during the work on this paper. We acknowledge financial support from the Australian National University, the CSIRO Deep Earth Imaging Future Science Platform, and the Australian Research Council (ARC) via DP200100053 and DE180100040. This research was supported by an Australian Government Research Training Program (RTP) Scholarship. We thank the editor Frederik Simons for handling the manuscript, and Vedran Lekic and an anonymous reviewer for providing helpful suggestions that improved the quality of this paper.

DATA AVAILABILITY

Code and data associated with this study are available at: https://github.com/MScheiter/gan_mcmc.

REFERENCES

Araya-Polo, M., Farris, S. & Florez, M., 2019. Deep learning-driven velocity model building workflow, *Leading Edge,* **38**(11), 872a1–872a9.

Arjovsky, M., Chintala, S. & Bottou, L., 2017. Wasserstein generative adversarial networks, in *Proceedings of the 34th International Conference on Machine Learning,* Vol. **70,** pp. 214–223, PMLR.

Bishop, C.M., 2006. *Pattern Recognition and Machine Learning,* Vol. **4,** Springer.

Box, G.E. & Muller, M.E., 1958. A note on the generation of random normal deviates, *Ann. Math. Statist.,* **29,** 610–611.

Burdick, S. & Lekić, V., 2017. Velocity variations and uncertainty from transdimensional P-wave tomography of North America, *Geophys. J. Int.,* **209**(2), 1337–1351.

Cowles, M.K. & Carlin, B.P., 1996. Markov Chain Monte Carlo convergence diagnostics: a comparative review, *J. Am. Stat. Assoc.,* **91**(434), 883–904.

Curtis, A. & Lomax, A., 2001. Prior information, sampling distributions, and the curse of dimensionality, *Geophysics,* **66**(2), 372–378.

Dinh, L., Sohl-Dickstein, J. & Bengio, S., 2017. *Density estimation using Real NVP,* preprint (arXiv:1605.08803).

Florez, M.A., Caporale, M., Buabthong, P., Ross, Z.E., Asimaki, D. & Meier, M.-A., 2020. *Data-Driven Accelerogram Synthesis using Deep Generative Models,* preprint (arXiv:2011.09038).

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y., 2014. Generative adversarial nets, in *Advances in Neural Information Processing Systems,* Vol. **27,** Curran Associates, Inc.

Grijalva, F., Ramos, W., Pérez, N., Benítez, D., Lara-Cueva, R. & Ruiz, M., 2021. ESeismic-GAN: a generative model for seismic events from Cotopaxi Volcano, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.,* **14,** 7111–7120.

Hallo, M. & Gallovič, F., 2020. Bayesian self-adapting fault slip inversion with Green's functions uncertainty and application on the 2016 Mw7.1 Kumamoto earthquake, *J. geophys. Res.,* **125**(3), e2019JB018703.

Henriques, L.F., Colcher, S., Milidiú, R.L., Bulcão, A. & Barros, P., 2021. *Generating Data Augmentation samples for Semantic Segmentation of Salt Bodies in a Synthetic Seismic Image Dataset,* preprint (arXiv:2106.08269).

Käufl, P., Valentine, A.P., de Wit, R.W. & Trampert, J., 2016. Solving probabilistic inverse problems rapidly with prior samples, *Geophys. J. Int.,* **205**(3), 1710–1728.

Kingma, D.P. & Ba, J., 2017. *Adam: A Method for Stochastic Optimization,* preprint (arXiv:1412.6980).

Kingma, D.P. & Welling, M., 2014. *Auto-Encoding Variational Bayes,* preprint (arXiv:1312.6114).

Koay, C.G., 2011. A simple scheme for generating nearly uniform distribution of antipodally symmetric points on the unit sphere, *J. Comput. Sci.,* **2**(4), 377–381.

Laloy, E., Hérault, R., Jacques, D. & Linde, N., 2018. Training-image based geostatistical inversion using a spatial generative adversarial neural network, *Water Resour. Res.,* **54**(1), 381–406.

Laloy, E., Linde, N., Ruffino, C., Hérault, R., Gasso, G. & Jacques, D., 2019. Gradient-based deterministic inversion of geophysical data with generative adversarial networks: is it feasible?, *Comput. Geosci.,* **133,** doi:10.1016/j.cageo.2019.104333.

Li, Z., Meier, M.-A., Hauksson, E., Zhan, Z. & Andrews, J., 2018. Machine learning seismic wave discrimination: application to earthquake early warning, *Geophys. Res. Lett.,* **45**(10), 4773–4779.

Lopez-Alvis, J., Laloy, E., Nguyen, F. & Hermans, T., 2021. Deep generative models in inversion: the impact of the generator's nonlinearity and development of a new approach based on a variational autoencoder, *Comput. Geosci.,* **152,** doi:10.1016/j.cageo.2021.104762.

Mather, B. & Fullea, J., 2019. Constraining the geotherm beneath the British Isles from Bayesian inversion of Curie depth: integrated modelling of magnetic, geothermal, and seismic data, *Solid Earth,* **10**(3), 839–850.

Monge, G., 1781. Mémoire sur la théorie des déblais et des remblais, *Histoire de l'Académie Royale des Sciences de Paris,* pp. 666–704.

Mosser, L., Kimman, W., Dramsch, J., Purves, S., De la Fuente Briceño, A. & Ganssle, G., 2018. Rapid seismic domain transfer: seismic velocity inversion and modeling using deep generative neural networks, in *Proceedings of the 80th EAGE Conference and Exhibition 2018,* pp. 1–5, European Association of Geoscientists & Engineers.

Mosser, L., Dubrule, O. & Blunt, M.J., 2020. Stochastic seismic waveform inversion using generative adversarial networks as a geological prior, *Math. Geosci.,* **52**(1), 53–79.

Mousavi, S., Tkalčić, H., Hawkins, R. & Sambridge, M., 2021. Lowermost mantle shear-velocity structure from hierarchical trans-dimensional Bayesian tomography, *J. geophys. Res.,* **126,** e2020JB021557.

Nash, J.F., 1950. Equilibrium points in n-person games, *Proc. Natl. Acad. Sci. U.S.A.,* **36**(1), 48–49.

Oliveira, B., Afonso, J. & Klöcking, M., 2021. Melting conditions and mantle source composition from probabilistic joint inversion of major and rare earth element concentrations, *Geochim. Cosmochim. Acta,* **315,** 251–275.

Oliveira, D.A., Ferreira, R.S., Silva, R. & Brazil, E.V., 2018. Interpolating seismic data with conditional generative adversarial networks, *IEEE Geosci. Remote Sens. Lett.,* **15**(12), 1952–1956.

Olugboji, T., Lekić, V. & McDonough, W., 2017. A statistical assessment of seismic models of the US continental crust using Bayesian inversion of ambient noise surface wave dispersion data, *Tectonics,* **36**(7), 1232–1253.

Radford, A., Metz, L. & Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks, preprint (arXiv:1511.06434).

Rezende, D.J. & Mohamed, S., 2016. Variational inference with normalizing flows, preprint (arXiv:1505.05770).

Roy, V., 2020. Convergence diagnostics for Markov Chain Monte Carlo, *Annu. Rev. Stat. Appl.,* **7**(1), 387–412.

Rubner, Y., Tomasi, C. & Guibas, L.J., 2000. The earth mover's distance as a metric for image retrieval, *Int. J. Comput. Vis.,* **40**(2), 99–121.

Rudolph, M., Moulik, P. & Lekić, V., 2020. Bayesian inference of mantle viscosity from whole-mantle density models, *Geochem. Geophys. Geosyst.,* **21**(11), e2020GC009335.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. & Chen, X., 2016. Improved techniques for training GANs, *Adv. Neural Inf. Process. Syst.,* **29,** 2234–2242.

Sambridge, M. & Mosegaard, K., 2002. Monte Carlo methods in geophysical inverse problems, *Rev. Geophys.,* **40**(3), 3–1-3-29.

Siahkoohi, A. & Herrmann, F.J., 2021. Learning by example: fast reliability-aware seismic imaging with normalizing flows, in *First International Meeting for Applied Geoscience & Energy,* pp. 1580–1585, Society of Exploration Geophysicists.

Siahkoohi, A., Kumar, R. & Herrmann, F., 2018. Seismic data reconstruction with generative adversarial networks, in *Proceedings of the 80th EAGE Conference and Exhibition 2018,* pp. 1–5, European Association of Geoscientists & Engineers.

Sohl-Dickstein, J., Weiss, E.A., Maheswaranathan, N. & Ganguli, S., 2015. Deep unsupervised learning using nonequilibrium thermodynamics, preprint (arXiv:1503.03585).

Steinberg, A., Vasyura-Bathke, H., Gaebler, P., Ohrnberger, M. & Ceranna, L., 2021. Estimation of seismic moment tensors using variational inference machine learning, *J. geophys. Res.,* **126**(10), e2021JB022685.

Tieleman, T. & Hinton, G., 2012. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning*.

Westwater, R. & Furht, B., 1997. The problem of video compression, in *Real-Time Video Compression: Techniques and Algorithms,* Chap. 1, pp. 1–13, Springer US.

Wieczorek, M.A. & Simons, F.J., 2005. Localized spectral analysis on the sphere, *Geophys. J. Int.,* **162**(3), 655–675.

Yang, Y., Mandt, S. & Theis, L., 2022. An introduction to neural data compression, preprint (arXiv:2202.06533).

Zhang, J. & Sheng, G., 2020. First arrival picking of microseismic signals based on nested U-Net and Wasserstein Generative Adversarial Network, *J. Petrol. Sci. Eng.,* **195,** doi:10.1016/j.petrol.2020.107527.

Zhang, X. & Curtis, A., 2021. Bayesian geophysical inversion using invertible neural networks, *J. geophys. Res.,* **126**(7), e2021JB022320.

Zhao, X., Curtis, A. & Zhang, X., 2022. Bayesian seismic tomography using normalizing flows, *Geophys. J. Int.,* **228**(1), 213–239.

APPENDIX A: GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) were first described by Goodfellow *et al.* (2014) and have since then become a popular and powerful machine learning technique. GANs consist of two neural networks, the generator and the discriminator, both of which have a particular task (Fig. A1). The generator takes a set of random numbers as input and aims to produce samples ('fake data') with the same characteristics as those drawn from a given distribution ('real data'), while the discriminator learns to distinguish between both distributions and assigns a probability that describes its estimate of a sample being drawn from the real data distribution. Both are trained at the same time in a competitive manner until reaching a Nash equilibrium, which in game theory describes the optimal strategy of both players in a two-player minimax game (Nash 1950).

The loss functions that the networks minimize during the training process are

$$L_D = -\frac{1}{N}\sum_{i=1}^{N}[\log D(x_i) + \log(1 - D(G(z_i)))] \tag{A1}$$

for the discriminator D, and

$$L_G = -\frac{1}{N}\sum_{i=1}^{N}\log D(G(z_i)) \tag{A2}$$

for the generator G. Here, x_i are samples from the real data distribution, and z_i are samples drawn from the noise prior. N denotes the batch size, that is how many samples from each distribution are drawn in each iteration. Consequently, the generator minimizes

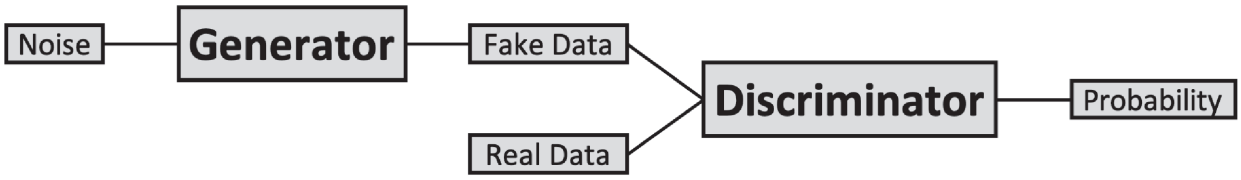

Figure A1. Architecture of a generative adversarial network. A generator learns to mimic samples from a given distribution, and a discriminator learns to identify samples from the distribution. Both networks are trained at the same time in competition with each other.

the negative average output of the samples it creates, that is tries to push the discriminator to evaluate its samples as real samples with probability as close to 1 as possible. The discriminator attempts to maximize two values (minimize the corresponding negative values): its evaluation of real samples and the distance between 1 and its evaluation of fake samples (i.e. it minimizes its evaluation of fake samples).

Based on the original GAN idea, many different architectures, loss functions, and training procedures have been proposed. Among the most important GAN variants are deep convolutional GANs (DCGANs) which use convolutional layers, making them more suitable for computer vision tasks and higher-dimensional data sets (Radford *et al.* 2015). Another variant are Wasserstein GANs (WGANs) which use metrics from the theory of optimal transport and lead to a significant improvement of stability during training (Arjovsky *et al.* 2017). Both these variants and the original formulation were used in different parts in this study, as described in the main text and Appendix B.

APPENDIX B: TECHNICAL CONSIDERATIONS

Generative models, and especially GANs, have become increasingly popular in the machine learning literature and beyond, and many thousands of different architectures and training procedures have been proposed. This vast amount of literature puts the interested user into the dilemma of having to choose between all these options. In this study, we have made use of three different GAN types, each with its own characteristics. In the following, we explain our heuristics for each of these choices, hoping to provide a guide for new potential users. We note that these observations are based on personal experience and thus inherently subjective. Details on architecture and training of all GANs used in this study can be found in Appendix B1.

In the synthetic example in Sections 3.1 and 3.3, we used a standard GAN after Goodfellow *et al.* (2014) with relatively few and thin layers, and therefore a comparably small amount of trainable parameters (see Table A1 for details). This example shows that small networks might be a suitable choice when the structure in the training data is not too complex and the problem is low-dimensional. Our experiments suggest that a standard GAN can lead to good performance up to dimensions of around 50–100, from where the well-known GAN issues of 'non-convergence' and 'mode collapse' start to make training more challenging (e.g. Salimans *et al.* 2016). Mode collapse is a well-known problem in the GAN literature, referring to an ill-trained GAN whose outputs only cover a very limited region (and often only a single point) of the underlying distribution, failing to recover the full diversity of samples contained in the training data set.

The example of the regional reconstruction of shear wave velocity beneath Australia (Section 4.2) deals with a higher-dimensional model ensemble with 529 parameters. Here, we use a deep convolutional GAN (DCGAN; Radford *et al.* 2015) with five layers and a larger amount of trainable parameters, leading to a higher capacity of the networks to store information (see Table A3 for full details). While training these networks takes significantly longer, the Monte Carlo ensemble is recovered with a high quality (Figs 5 and 6). This makes DCGANs a good choice in cases where an accurate version of the model ensemble is required for sampling in subsequent studies.

When learning the global velocity model ensemble (Section 4.4), we use the training procedure from Wasserstein GANs (WGAN; Arjovsky *et al.* 2017) with larger layers than in the initial synthetic example (Table A4). Our experiments confirm the general notion that WGANs have better convergence performance than standard GANs. To further increase performance, we split the model ensemble across 16 different WGANs with around 500 model parameters each. Due to the spherical geometry and the fact that DCGANs require inputs in rectangular form, DCGANs are less appropriate in this case. Compared to DCGANs, WGANs recover less details, but are preferable in terms of both training time and compression rate.

These trade-offs make the choice of the best suitable architecture a non-trivial endeavour, and it needs to be decided on a case-to-case basis. There is no 'perfect architecture' that suits all possible requirements. When deciding on an architecture, different options need to be tested. While the above heuristics are suitable for our needs, they are likely particular to this study and care must be taken in translating them to other applications. In any case, the architecture should merely be seen as a tool that helps achieving the intended goals in the best possible manner.

The issues of network architecture also hold for the other hyperparameters. An extensive search of the hyperparameter space may be necessary in order to find the best set of values. However, we note that from our perspective the most important choices seem to be with regards to the optimizer and its parameters such as learning rate and momentum parameters. Also the length of training needs to be chosen with care, as GANs are prone to diverge again after previous convergence. It is therefore essential to find the right moment to stop training.

In this study, we tested three different GAN architectures that are suitable for different needs. However, these options are by no means an exhaustive list, and many other architectures might work. The same holds for other generative models such as variational autoencoders, diffusion models, and flow-based models, which have not been investigated in this study. Almost certainly, there will be cases where these models outperform GANs, not least because of more stable training. Consequently, other generative models may be good alternatives in order to make use of the concepts that were presented and discussed here.

Table A1. Network architectures for the GANs used in the synthetic example in Sections 3.1 and 3.3. Dropout is used in both training and evaluation stages. FC: fully connected layer.

	Generator				Discriminator		
	n_{in}	n_{out}	Activation	Dropout	n_{in}	n_{out}	Activation
Layer 1 (FC)	100	50	LeakyReLU(0.1)	0.4	2..10	50	LeakyReLU(0.1)
Layer 2 (FC)	50	50	LeakyReLU(0.1)	0.4	50	50	LeakyReLU(0.1)
Layer 3 (FC)	50	50	LeakyReLU(0.1)	0.4	50	50	LeakyReLU(0.1)
Layer 4 (FC)	50	2..10	–	–	50	1	Sigmoid
No. parameters	10 252 (2-D) to 10 660 (10-D)				5301 (2-D) to 5701 (10-D)		

Table A2. Training settings for the GANs used in the synthetic example in Sections 3.1 and 3.3. Original GAN: Goodfellow *et al.* (2014); Adam: Kingma & Ba (2017). Training time is indicated in CPU minutes on a 3.7 GHz Intel Core i5 processor with 16 GB 2667 MHz DDR4 RAM.

	Section 3.1	Section 3.3
GAN type	Original GAN	Original GAN
Optimizer	Adam ($\beta_1 = 0.5$, $\beta_2 = 0.999$)	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Learning rate	10^{-4}	10^{-4}
Batch size	50	100
Data dimension	2	2–10
No. samples	200/600/1000/100 000	300 000
No. epochs	100 000/33 333/20 000/200	500
Gen./Disc. updates	1/1	1/1
Training time	$\sim$ 15 min	$\sim$ 60 min

Table A3. Network architectures and training settings for the DCGANs used in the reconstruction of the Australian patch in Sections 4.2 and 4.3. DCGAN: Radford *et al.* (2015); Adam: Kingma & Ba (2017); 2-D Conv. = 2-D convolutional layer; Transp. 2-D Conv. = Transposed 2-D convolutional layer; in = no. input channels; out = no. output channels; k = kernel size; s = stride; p = padding. In both networks, all weights were initialized according to a normal distribution of mean 0 and standard deviation 0.02, and all biases were set to zero. Training time is indicated in CPU hours on a 3.7 GHz Intel Core i5 processor with 16 GB 2667 MHz DDR4 RAM.

Generator	n_{in}	n_{out}	k	s	p	BatchNorm	Activation
Layer 1 (Transp. 2D Conv.)	100	512	4	2	1	yes	ReLU
Layer 2 (Transp. 2D Conv.)	512	256	4	2	1	yes	ReLU
Layer 3 (Transp. 2D Conv.)	256	128	4	2	1	yes	ReLU
Layer 4 (Transp. 2D Conv.)	128	64	4	2	1	yes	ReLU
Layer 5 (Transp. 2D Conv.)	64	1	3	2	5	no	Tanh
No. parameters	3 574 208						
Discriminator	n_{in}	n_{out}	k	s	p	BatchNorm	Activation
Layer 1 (2D Conv.)	1	64	6	1	0	no	LeakyReLU(0.2)
Layer 2 (2D Conv.)	64	128	4	2	1	yes	LeakyReLU(0.2)
Layer 3 (2D Conv.)	128	256	4	2	1	yes	LeakyReLU(0.2)
Layer 4 (2D Conv.)	256	512	4	2	1	yes	LeakyReLU(0.2)
Layer 5 (2D Conv.)	512	1	4	2	1	no	Sigmoid
No. parameters	2 764 800						
Training settings							
GAN type	Deep convolutional GAN (DCGAN)						
Optimizer	Adam ($\beta_1 = 0.5$, $\beta_2 = 0.999$)						
Learning rate	10^{-4}						
Batch size	128						
Data dimension	529 (23×23)						
No. samples	180 000 (100%); 90 000 (50%); 72 000 (40%)						
No. epochs	5 (100%); 10 (50%); 13 (40%)						
Gen./Disc. updates	1/1						
Training time	$\sim$ 5 hr						

Table A4. Network architectures and training settings for the 16 GANs used in the global reconstruction in Section 4.4. Dropout is used in both training and evaluation stages. WGAN: Arjovsky *et al.* (2017); RMSProp: Tieleman & Hinton (2012); FC = fully connected layer. Training time is indicated in CPU hours on a 3.7 GHz Intel Core i5 processor with 16 GB 2667 MHz DDR4 RAM.

Generator	n_{in}	n_{out}	Activation	Dropout
Layer 1 (FC)	100	200	LeakyReLU(0.1)	0.4
Layer 2 (FC)	200	200	LeakyReLU(0.1)	0.4
Layer 3 (FC)	200	200	LeakyReLU(0.1)	0.4
Layer 4 (FC)	200	∼500	–	–
No. parameters			∼201 100	
Discriminator	n_{in}	n_{out}	Activation	Dropout
Layer 1 (FC)	∼500	200	LeakyReLU(0.1)	–
Layer 2 (FC)	200	200	LeakyReLU(0.1)	–
Layer 3 (FC)	200	200	LeakyReLU(0.1)	–
Layer 4 (FC)	200	1	–	–
No. parameters			∼180 801	
Training settings				
GAN type			Wasserstein GAN (WGAN)	
Clip value			0.01	
Optimizer			RMSProp	
Learning rate			10^{-4}	
Batch size			100	
Data dimension			469–532	
No. samples			180 000	
No. epochs			200	
Gen./Disc. updates			1/1	
Training time			∼ 1 hr	

B1 Network architectures and training settings

Tables A1–A4 show details of architecture and training settings of all GANs used in this study.