Enabling uncertainty quantification for seismic data pre-processing using normalizing flows (NF)—an interpolation example

Rajiv Kumar[1], Maria Kotsi[2,4], Ali Siahkoohi[3], Alison Malcolm[4]
[1]Schlumberger
[2]PanGeo Subsea
[3]Georgia Institute of Technology
[4]Memorial University of Newfoundland

Abstract

Seismic data go through a sequence of pre-processing steps before being made into an image. Although some work has been done to assess the uncertainties in the final images, how the uncertainty in the pre-processing affects the results remains largely unexplored. We use Normalizing Flows (NF), a type of deep neural network, to interpolate seismic data and quantify the associated uncertainty. A big advantage of NFs, over the more commonly used Markov Chain Monte Carlo methods, is that they can successfully sample a complex and high-dimensional probability space with fewer assumptions. We present the first application of NF in interpolating (synthetic) seismic data. The statistical measurements retrieved from the network can be used to better characterize the data as it is passed to the post-processing phase.

Introduction

Acquired seismic data typically pass through a series of carefully designed pre-processing steps, including interpolation, before they are used to estimate Earth properties using imaging or inversion methods. Seismic interpolation is the process through which irregularly subsampled data are transformed into equally and fully–sampled data [Jia and Ma, 2017]. There are a variety of methods for seismic interpolation ranging from wave-equation methods (Stolt [2002]; Fomel [2003]) to domain transform methods (Zwartjes and Sacchi [2007], Hennenfent and Herrmann [2008], Trad [2009], Naghizadeh [2012], Gao et al. [2015], Liu et al. [2017]) and more recently using deep learning methods (Siahkoohi et al. [2018]; Siahkoohi et al. [2019]; Oliveira et al. [2018]; Harpreet et al. [2021]). Although we evaluate the output of each pre–processing step qualitatively, we rarely quantify the associated uncertainty at each step. Quantifying this uncertainty would enable a more informed assessment of the data quality and its impact on the final image. For example, we can use the statistical properties, such as the posterior standard deviation, as weights to distinguish between well and poorly constrained parts of the processed data during interpretation, imaging or inversion.

Uncertainty quantification is usually expressed through Bayesian inferences [Bayes and Price, 1763] and typically calculated with Markov Chain Monte Carlo (MCMC) methods (Metropolis et al. [1953]; Haario et al. [2001]; Neal [2011]; Hoffman and Gelman [2014]). These algorithms start by assuming an initial probability distribution, the so–called prior distribution, and after a finite sampling time, they yield the so–called posterior distribution. However, due to the curse of dimensionality, all of these methods become intractably expensive for large–scale problems. As a result, we assume that the output of each pre-processing step has an insignificant and constant uncertainty across all the samples. In other words, we give equal weight to each sample of the seismic data produced by our pre-processing sequence during imaging and inversion.

Rather than using MCMC, an alternative approach to estimating uncertainties is to use Neural Networks (NN), which can learn a map between input and output data [Bishop, 2006]. NN have successfully been used to solve both inverse (Röth and Tarantola [1994]; Kong et al. [2019]) and inference problems (Devilee et al. [1999]; Earp and Curtis [2020]). Even though NN can provide fast predictions, they have to be robustly trained over the entire prior probability space, which is infeasible when dealing with large data sets. A proposed solution to the limitations of both MCMC and NN is Normalizing Flows (NF) [Rezende and Mohamed, 2015]. NF can convert a simple distribution (e.g. prior distribution) to a complex distribution (e.g. posterior distribution) by passing it through a chain of invertible transforms (the so–called flows). A big advantage of NF is that they provide an approximation of the posterior distribution for a given data set, meaning that they don't need to sample the entire prior probability space, and hence are more computationally efficient.

Recent work in the seismic community shows the advantages of using NF for uncertainty quantification for seismic inverse problems, where the computational cost is dominated by the expensive numerical solution of partial differential equations (PDEs) (Siahkoohi et al. [2020]; Rizzuti et al. [2020]; Zhao et al. [2020]; Siahkoohi et al. [2021]). These authors show various formulations of NF and that the cost of uncertainty quantification is much lower than when using standard MCMC. These past studies focus on the inverse problem. In this study, our focus is on performing the uncertainty quantification for seismic data pre-processing. We provide a proof of concept that NF can be successfully applied to the seismic data pre-processing steps such as the interpolation problem. The NF does not only pre-process the seismic data but also provides the associated posterior distribution, from which we can extract statistical parameters. To our knowledge, this is the first instance where NF are used for seismic pre-processing uncertainty quantification.

Methods

We begin from a standard Bayesian framework in which the posterior, $p(m|d)$, is estimated from the prior, $p(m)$, the evidence, $p(d)$, and the likelihood, $p(d|m)$, via Bayes' theorem,

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

where m are the model parameters and d the data. Although estimating $p(m|d)$ is commonly done with MCMC, there are two key problems with this approach. The first is the time required for sampling. Particularly in higher dimensions at least millions of samples are required for a convergent MCMC algorithm. This leads naturally to the second problem, which is that most techniques for evaluating convergence of an MCMC algorithm are somewhat ad-hoc and are true only in a statistical sense. This means that it is quite difficult to assess whether the algorithm has converged or not.

There are several methods that have been proposed to attempt to alleviate these problems, but here we focus on transport methods. These methods, introduced by El Moselhy and Marzouk [2012] and explained in more detail in Marzouk et al. [2016], replace the sampling of MCMC with a method of mapping one distribution (in this case the prior) to another (in this case the posterior). More specifically, they solve an optimization problem to find a vector mapping T that morphs one distribution into another. To maintain some generality and to make the notation easier for the NF, we rename them to the target (posterior, τ) and reference (prior, ρ) distributions. Note that the reference distribution is often called the latent space in NF. Marzouk et al. [2016] setup this problem as finding a function T such that

$$\tau = \rho \circ T^{-1} |\det \nabla T^{-1}|\,, \tag{2}$$

where $\circ$ indicates functional composition and det is the determinant. The determinant term follows directly from the change of variables formula. Note that T is a vector function (i.e. returns a vector) of vector parameters (i.e. takes a vector as input).

The idea is then to solve an optimization problem to minimize the Kullback-Leibler (K-L) divergence between the target τ and the actual posterior $p(m|d)$. The K-L divergence measures the similarity between two distributions and is defined as

$$\mathcal{D}_{KL}(\tau_1 \| \tau_2) = \mathbb{E}_{\tau_1}\left(\log \frac{\tau_1}{\tau_2}\right), \tag{3}$$

where $\mathbb{E}_{\tau_1}$ is the expectation over τ_1. It is not too difficult to see that this approach is also not computationally free. The computation of the gradient requires significant calculation and is not practical for large problems with large correlations between parameters. This leads us to two observations/advances. The first is the idea of how to set up the function and the second is the solving of the optimization problem. For the former, it is possible to setup the problem so that the gradient is lower-triangular. This is done by enforcing that each component of T, T_i, depends only on the variables $x_{1\cdots i-1}$. This is of key importance because it reduces the determinant calculation to the product of the diagonal elements of the ∇T matrix. The second is where NF come in. It is via NF that we learn these transformations in an efficient way.

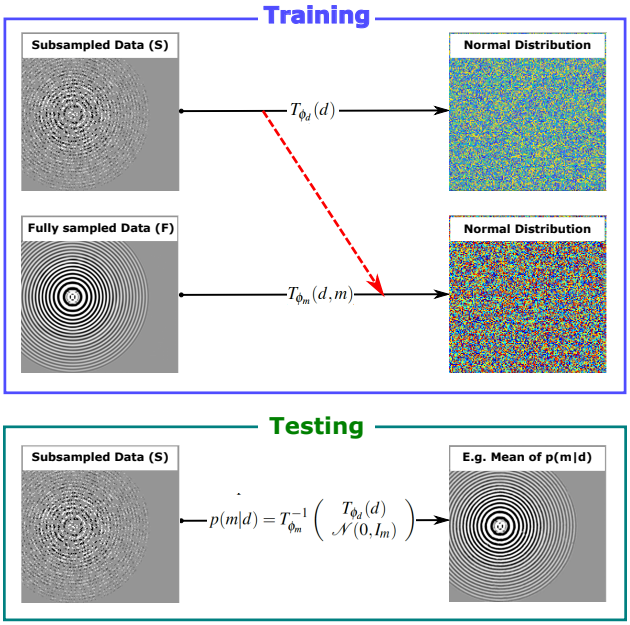

Figure 1: Simple illustration of the HINT architecture for the interpolation of seismic data.

Having established what we are trying to do, namely to map one distribution to another, we now move forward into the question of how we do this in a computationally efficient way. This is where NF enter our discussion. NF are a machine-learning-based method that allows us to learn a mapping from one distribution to another. More specifically, we are going to setup a way to learn the T mapping, but do it in a way that allows for us to train a network (i.e. to fit a set of parameters). We do this by setting up a set of mapping functions, which are both easy to compute, and result in an invertible network so that we can go either from data to model or from model to data. To understand how to make these functions easy to compute, we start from equation 2 and extend it to a sequence of transformations. To that end, we begin with a random variable, d and transform it through a series of changes of variable via

$$d_k = f_k \circ \cdots \circ f_2 \circ f_1(d)\,, \tag{4}$$

by basic probability definitions, this transforms the distribution $\rho(d)$ via

$$\log \rho_k(d_k) = \log \rho_0(z_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial_{d_{k-1}}} \right| . \tag{5}$$

We need to define f_k (where $f_k \circ \cdots \circ f_2 \circ f_1 = T$ above) in a way that allows us to learn it and such that we can easily compute the Jacobian $\left|\det \frac{\partial f_k}{\partial_{d_{k-1}}}\right|$. To do this, we first define a specific form for f_i, which we will refine slightly later. This form is

$$f(x) = \begin{pmatrix} x_1 \\ C(x_2|x_1) \end{pmatrix}, \tag{6}$$

where $x_1 = x_{0:N/2}$ and $x_2 = x_{N/2:N}$ split the input vector x into two parts. For the function C, we use

$$C(x_2|x_1) = x_2 \odot \exp(s(x_1)) + t(x_1)\,, \tag{7}$$

where $\odot$ represents an element-wise multiplication (Hadamard product) and s and t are the parameters that we wish to learn. Dinh et al. [2016] show that this C function is easily invertible and that neither its inverse nor its Jacobian depend on s or t in addition to showing that its Jacobian is lower triangular. This satisfies our requirements that the function be invertible and that it have an easily computed Jacobian. (Because the Jacobian is lower-triangular, its determinant is the product of its diagonal elements, meaning that we need only compute those diagonal elements.) In addition, because neither the Jacobian nor the inverse depend on s and t, we can make them arbitrarily complicated. Continuing to follow Dinh et al. [2016], we use deep neural networks for both s and t. Note that s stands for scale and t for translation, so we can think of this transformation as scaling and shifting our distribution. The s-network uses the first half of the vector x_1 as its inputs and outputs a set of weights of size x_2 that are then multiplied by x_2, which scales each element of x_2 by a different amount, allowing us to change a symmetric distribution into an asymmetric one for example. The translation network, t, works in a similar manner.

A key limitation thus far is that we allow the x_1-variables to scale the x_2-variables but there is no way for the x_2 variables to scale or adapt the x_1-variables. This is what makes the transformation lower-triangular but it does remove some potentially useful scaling functions. This issue is solved by the Hierarchical Invertible Neural Transport (HINT) method proposed by Kruse et al. [2019]. What they suggest is that rather than splitting the input dimensions in two, we instead recursively split the variables into smaller and smaller sets, and allow for permutations of those sets between layers. It is this framework that we use here.

Any NF consists of a sequence of blocks, each of which contains both a scaling and translation network as described above. These blocks taken together make up a sequence of neural networks that successfully map one distribution into another, as illustrated in Figure 1). But there is still a key component missing, which is that as we have described it thus far, the network allows us to recover one joint distribution for x_1 and x_2, or in terms of the notation used above we can recover the joint distribution of the data and model $p(m, d)$. Kruse et al. [2019] also solve this problem, allowing us to recover not just $p(m, d)$ but also the posterior we are looking for, $p(m|d)$. To approximate the posterior distribution, Kruse et al. [2019] solves the following optimization problem with block-triangular conditional NF, $T_\phi : d \times m \to z_d \times z_m$:

$$\min_{\phi} \mathbb{E}_{d,m\sim p_{d,m}(d,m)} \left[\frac{1}{2} \|T_\phi(d, m)\|^2 - \log \left| \det \nabla_{d,m} T_\phi(d, m) \right| \right], \tag{8}$$

where $T_\phi(d, m) = \begin{bmatrix} T_{\phi_d}(d) \\ T_{\phi_m}(d, m) \end{bmatrix}, \phi = \{\phi_d, \phi_m\}$. The NF T_ϕ is parameterized with ϕ and $z_d \times z_m$ and represents the latent space of the prior distribution. To solve the above expression, Kruse et al. [2019] first split the input data into two parts (data and model in this case, but it could also be data and labels or any other pair), and then allow a one-way connection between the data and model. In other words, they find two flows, one which maps between a simple latent space (e.g. a normal distribution) and the data and one which maps between a second latent space and the model. They then connect (via a third set of scale/translate networks,

red dashed arrow in Figure 1) the data network to the model network but not vice-versa. This allows one to fix the data and sample the model space via

$$p(m|d) = T_{\phi_m}^{-1} \begin{pmatrix} T_{\phi_d}(d) \\ \mathcal{N}(0, I_m) \end{pmatrix}, \tag{9}$$

where $T_{\phi_m}^{-1}$ is the pull-back of the latent density, in other words it takes samples from the latent space of both model and data and maps them to the actual data/model spaces. What equation 9 says is that we can apply the following sequence of steps:

1. For a given dataset (subsampled data), find the associated point in the latent space (typically a simple normal distribution).
2. Fix the data components (i.e. the x_1) in the latent space, and sample the rest of the parameters (i.e. the model parameters, or x_2) in the latent space. This is straightforward as we are simply drawing parameters from a normal distribution.
3. Map the drawn samples for the model to the true model space, giving the posterior and our distribution of interpolated data.

In this way, we have setup a framework for computing the uncertainty in a standard processing step. By chaining NF together, we could then potentially map the uncertainty of an entire processing sequence, resulting in a range of datasets that capture the range of possibilities from the applied processing steps.

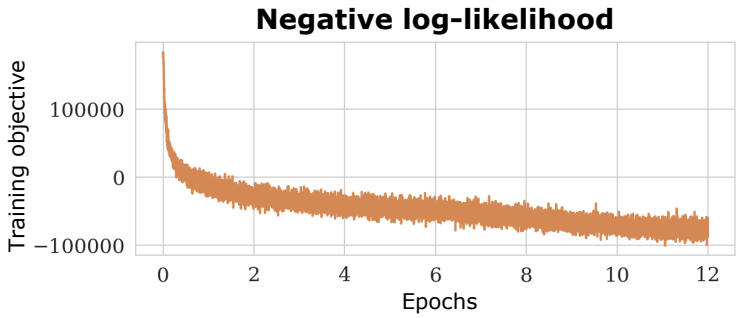

Figure 2: The negative log–likelihood of the training data set. The data set was trained for 12 epochs, where each epoch takes approximately one hour.

Numerical Example

We use a seismic interpolation example to demonstrate the efficacy of the NF for uncertainty quantification (UQ) during seismic pre-processing. The synthetic data are generated using finite difference on a 3D geologically complex BG COMPASS model (Herrmann et al. [2013]). The data contain 172×172 sources where each source is recorded by 172×172 receivers on a 25m grid. For computational efficiency, we first apply the Fourier transform along the time-dimension and then perform all of our tests on a 4-dimensional monochromatic frequency dataset at 10Hz. For the training dataset we randomly selected 75% of the sources and the other 25% of the sources are used for testing. To generate the subsampled dataset (Figure 3), we randomly remove 75% of the receivers from a common source gather (Figure 3). Since the size of a fully sampled common source gather is 172×172, this leads to a total of 29,584 degrees of freedom while

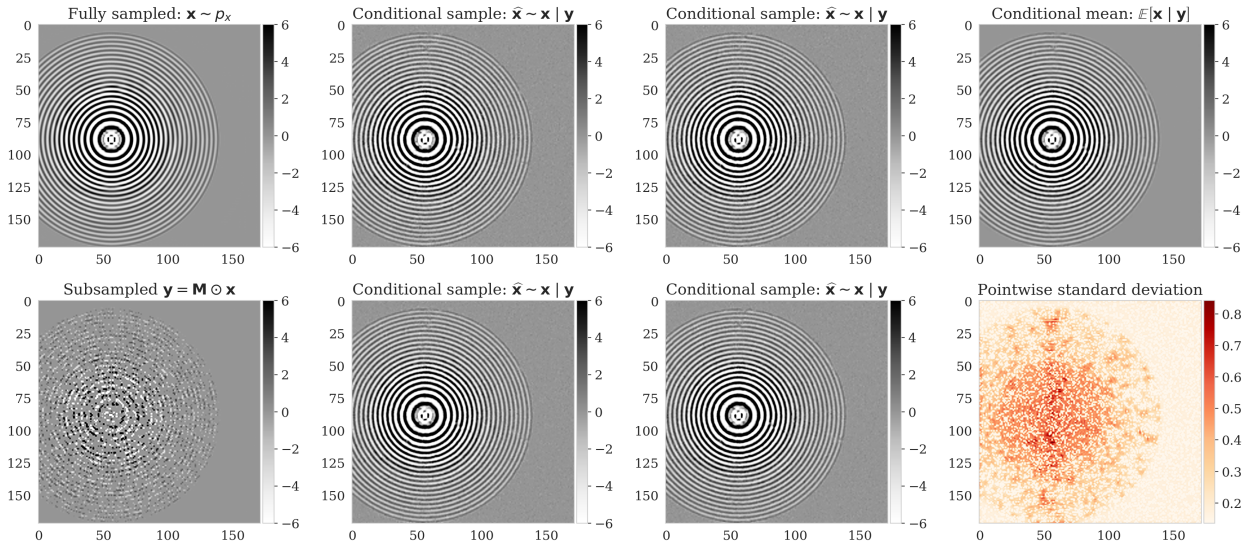

Figure 3: From left to right: The first column top panel shows the fully–sampled data from the test dataset, which serves as a benchmark to quantify the efficacy of NF for data interpolation. The bottom panel shows the subsampled data used to perform the interpolation using the trained NF. The second and third column show different models that they were drawn from the posterior distribution. The fourth column top panel displays the mean of the approximated posterior distribution, while the bottom panel displays the standard deviation related to this approximated posterior distribution. The signal-to-noise ratio in this example is 15.5 dB.

performing the UQ for a common source gather. Note that code to reproduce our results are made available on GitHub. Also, our implementation relies on InvertibleNetworks.jl Witte et al. [2020], a recently-developed memory-efficient framework for training invertible networks in the Julia programming language.

As described in the methods section, we use recursive coupling blocks as a baseline architecture for the NF Kruse et al. [2019], which use invertible coupling layers Dinh et al. [2016] in a hierarchical way. The dense architecture of the NF combines several of these hierarchical coupling blocks to improve the representative power of NF Kruse et al. [2019] (i.e. to improve the number and type of correlations between parameters that can be successfully mapped to and from the latent space). One of the key parts of the architecture of each of the hierarchical coupling blocks is the affine coupling block. This block is composed of a residual block, where the first and last layer of the residual block require 2 input/output channels whereas the intermediate layer dimensions are 64 input, 128 hidden and 64 output channels. Here we use 8 hierarchical coupling blocks for $T_{\phi_d}(d)$, $T_{\phi_m}(d, m)$ while solving equation 8 for the network parameters ϕ.

For training purpose, we use 20000 training pairs where we do training for 12 epochs with a batch size of 64 and a (starting) learning rate of 0.001. After each epoch we decrease the learning rate by a factor of 0.9. Each epoch on a GPU takes approximately one hour. Figure 2 is the evaluation of the loss-function over 12 epochs during the training phase. Note that the value of the objective function in equation 8 can be negative because we are estimating the negative log likelihood of a probability density function (PDF) and this likelihood value can be larger than 1, resulting in a negative value of objective function. The integral of the PDF over the entire space will still be 1. The outputs of the NF are shown in Figure 3. The second and third column show different samples randomly selected from the approximated posterior distribution. The fourth column shows a measure of the associated uncertainty. Here, by uncertainty we simply mean meaningful statistical measures, such as mean and standard deviation, that describe the variability of the predicted output. As evident from the results, NF are able to successfully perform seismic data interpolation with a recovered

signal-to-noise ratio of 15.5 dB, while simultaneously approximating the posterior distribution.

One can potentially think of NF as a sampling MCMC technique, however, with some key differences from the classical Metropolis–Hastings algorithms. First, NF outperform other MCMC methods when dealing with high–dimensional spaces. For example, in this study we had 29,584 degrees of freedom and this took the NF network 10,000 iterations of training. Such a high dimensional space would be impossible to sample using only 10,000 iterations even with advanced sampling techniques such as Hamiltonian Monte Carlo or Trans–dimensional MCMC. Second, NF are model agnostic, meaning that they do not require any knowledge of the distributions nor on how to best parameterize a model. Both of these aspects are important for a successful MCMC inversion. Third, the standard deviation of an NF prediction is not identical to the standard deviation of a distribution from an MCMC inversion, due to the use of fewer iterations and the fact that NF offer only an approximation of this distribution. Therefore, the standard deviation from Figure 3 should be interpreted more like a reliability measure than as an error bar in the predicted output.

Conclusions

We propose the use of NF for efficient seismic data interpolation and uncertainty quantification. To our knowledge, this is the first application of NF to data pre–processing. Having a reliability measurement associated with the data in the pre–processing stage can be key as data are passed through the next stages. For example, the standard deviation output of the NF could be used as a weighting factor during inversion and migration. Our future work will be focused on firstly understanding how model complexities affect the transformation maps (flows) and their accuracy in the posterior approximation, and secondly how to efficiently tune the network hyper–parameters by using optimization methods.

Acknowledgments

Rajiv Kumar thanks Schlumberger for supporting the preparation of this work. The work of Maria Kotsi and Alison Malcolm was supported by Chevron and with grants from the Natural Sciences and Engineering Research Council of Canada Industrial Research Chair Program and InnovateNL. The work of Ali Siahkoohi was supported by Georgia Research Alliance and partners of the ML4Seismic Center.

References

T. Bayes and R. Price. An essay towards solving a problem in the doctrine of chance. *Phil. Trans. R. Soc.*, 53:370–418, 1763.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

R. Devilee, A. Curtis, and K. Roy-Chowdhury. An efficient, probabilistic neural network approach to solving inverse problems: Inverting surface wave velocities for eurasian crustal thickness. *Journal of Geophysical Research: Solid Earth*, 104(B12):28841–28857, 1999.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

S. Earp and A. Curtis. Probabilistic neural network–based 2D travel–time tomography. *Neural Computing and Applications*, 32(22):177077–17095, 2020.

Tarek A El Moselhy and Youssef M Marzouk. Bayesian inference with optimal maps. *Journal of Computational Physics*, 231(23):7815–7850, 2012.

S. Fomel. Seismic reflection data interpolation with differential offset and shot continuation. *Geophysics*, 68: 733–744, 2003.

J. Gao, A. Stanton, and M. D. Sacchi. Parallel matrix factorization algorithm and its application to 5D seismic reconstruction and denoising. *Geophysics*, 80(6):V173–V187, 2015.

H. Haario, E. Saksman, and J. Tamminen. An adaptive metropolis algorithm. *Bernouli*, 7(2):223–242, 2001.

K. Harpreet, N. Pham, and S. Fomel. Seismic data interpolation using deep learning with Generative Adverarial Networks. *Geophysical Prospecting*, 69(2):307–326, 2021.

Gilles Hennenfent and Felix J Herrmann. Simply denoise: Wavefield reconstruction via jittered undersampling. *Geophysics*, 73(3):V19–V28, 2008.

Felix J. Herrmann, Andrew J. Calvert, Ian Hanlon, Mostafa Javanmehri, Rajiv Kumar, Tristan van Leeuwen, Xiang Li, Brendan R. Smithyman, Eric Takam Takougang, and Haneet Wason. Frugal full-waveform inversion: from theory to a practical algorithm. *The Leading Edge*, 32(9):1082–1092, 09 2013. doi: 10.1190/tle32091082.1. URL https://slim.gatech.edu/Publications/Public/Journals/TheLeadingEdge/2013/herrmann2013ffwi/herrmann2013ffwi.html.

M. D. Hoffman and A. Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1593–1623), 2014.

Y. Jia and J. Ma. What can machine learning do for seismic data processing? a machine learning application. *Geophysics*, 82(3):V163–V177, 2017.

Q. Kong, D. T. Trugman, Z. E. Ross, M. J. Bianco, B. J. Meade, and P. Gerstoft. Machine learning in seismology: Turning data into insights. *Seismological Research Letters*, 90(1):3–14, 2019.

Jakob Kruse, Gianluca Detommaso, Robert Scheichl, and Ullrich Köthe. Hint: Hierarchical invertible neural transport for density estimation and bayesian inference. *arXiv preprint arXiv:1905.10687*, 2019.

L. Liu, G. Plonka, and J. Ma. Seismic data interpolation and denoising by learning a tensor tight frame. *Inverse Problems*, 33, 2017.

Youssef Marzouk, Tarek Moselhy, Matthew Parno, and Alessio Spantini. An introduction to sampling via measure transport. *arXiv preprint arXiv:1602.05023*, 2016.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Mostafa Naghizadeh. Seismic data interpolation and denoising in the frequency-wavenumber domain. *Geophysics*, 77(2):V71–V80, 2012.

R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, page Chapter 5, 2011.

D. A. Oliveira, R. S. Ferreira, R. Silva, and E. V. Brazil. Interpoloating seismic data with conditional generative adversarial networks. *IEEE Geoscience and Remote Sensing Letters*, pages 1–5, 2018.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Gabrio Rizzuti, Ali Siahkoohi, Philipp A. Witte, and Felix J. Herrmann. Parameterizing uncertainty by deep invertible networks, an application to reservoir characterization. In *SEG Technical Program Expanded Abstracts*, pages 1541–1545, 09 2020. doi: 10.1190/segam2020-3428150.1. URL https://slim.gatech.edu/Publications/Public/Conferences/SEG/2020/rizzuti2020SEGuqavp/rizzuti2020SEGuqavp.html. (SEG, virtual).

G. Röth and A. Tarantola. Neural networks and inversion of seismic data. *Journal of Geophysical Research: Solid Earth*, 99(B4):6753–6768, 1994.

Ali Siahkoohi, Rajiv Kumar, and F Herrmann. Seismic data reconstruction with generative adversarial networks. In *80th EAGE Conference and Exhibition 2018*, volume 2018, pages 1–5. European Association of Geoscientists & Engineers, 2018.

Ali Siahkoohi, Rajiv Kumar, and Felix J. Herrmann. Deep-learning based ocean bottom seismic wavefield recovery. In *SEG Technical Program Expanded Abstracts*, pages 2232–2237, 09 2019. doi: 10.1190/segam2019-3216632.1. URL https://slim.gatech.edu/Publications/Public/Conferences/SEG/2019/siahkoohi2019SEGdlb/siahkoohi2019SEGdlb.html. (SEG, San Antonio).

Ali Siahkoohi, Gabrio Rizzuti, Philipp A. Witte, and Felix J. Herrmann. Faster uncertainty quantification for inverse problems with conditional normalizing flows. Technical Report TR-CSE-2020-2, Georgia Institute of Technology, 07 2020. URL https://slim.gatech.edu/Publications/Public/TechReport/2020/siahkoohi2020TRfuqf/siahkoohi2020TRfuqf.html.

Ali Siahkoohi, Gabrio Rizzuti, Mathias Louboutin, Philipp A. Witte, and Felix J. Herrmann. Preconditioned training of normalizing flows for variational inference in inverse problems. In *3rd Symposium on Advances in Approximate Bayesian Inference*, 01 2021. URL https://slim.gatech.edu/Publications/Public/Conferences/AABI/2021/siahkoohi2021AABIpto/siahkoohi2020ABIfab.html. AABI.

R. H. Stolt. Seismic data mapping and reconstruction. *Geophysics*, 67:890–908, 2002.

Daniel Trad. Five-dimensional interpolation: Recovering from acquisition constraints. *Geophysics*, 74(6): V123–V132, 2009.

Philipp Witte, Gabriel Rizzuti, Mathias Louboutin, Ali Siahkoohi, and Felix Herrmann. Invertiblenetworks.jl: A julia framework for invertible neural networks, November 2020. URL https://doi.org/10.5281/zenodo.4298853.

Xuebin Zhao, Andrew Curtis, and Xin Zhang. Bayesian seismic tomography using normalizing flows. 2020.

PM Zwartjes and MD Sacchi. Fourier reconstruction of nonuniformly sampled, aliased seismic data. *Geophysics*, 72(1):V21–V32, 2007.