DEEP NEURAL NETWORK SURROGATES FOR INVERSE PROBLEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ENERGY RESOURCES ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Zitong Zhou
July 2021

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daniel Tartakovsky, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Roland Horne

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Peter Kitanidis

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Inverse problems in subsurface flow are generally challenging due to the need for a large number of expensive numerical solutions to partial differential equations (PDEs). Inverse modeling typically consists of generating realizations of the unknown model parameters and matching the corresponding model's prediction to the measurements. Model errors, measurement errors, and data scarcity require one to quantify predictive uncertainty in model predictions, which further exacerbates the computational cost of inverse modeling. The latter can be ameliorated either by devising inversion frameworks that require fewer forward model runs to converge, or by constructing a much more efficient forward surrogate model that replaces the PDE solver. In this dissertation, we pursue these two strategies and apply them to three inverse problems of practical importance in subsurface applications. The common thread in this investigation is the use of deep neural network (DNN) surrogates that accelerate forward modeling by several orders of magnitude.

We first present a study on identification of the statistical parameters of a discrete fracture network (DFN). These parameters are field-scale properties of fractured rocks, which play a crucial role in many subsurface problems. In this study, we develop an inversion technique to infer two such parameters, fracture density and fractal dimension of a DFN, from cross-borehole thermal experiments data. Our approach relies on a particle-based heat-transfer model, whose evaluation is accelerated with a DNN surrogate that is integrated into a grid search. The DNN is trained on a small number of heat-transfer model runs, and predicts the cumulative density function of the thermal breakthrough time. The latter is used to compute fine posterior distributions of the (to-be-estimated) parameters. Our synthetic experiments reveal that fracture density is well constrained by data, while fractal dimension is harder to determine. Adding non-uniform prior information related to the DFN connectivity improves the inference of this parameter.

We further focus on a higher-dimensional inverse problem on contaminant source identification. The reconstruction of contaminant release history from sparse observations of solute concentration is a key component of the design of an efficient subsurface remediation strategy. Markov chain Monte Carlo (MCMC), the most general method for this task, is rarely used in practice because of its high computational cost associated with multiple solves of contaminant transport equations. We investigate two MCMC variants: delayed rejection adaptive Metropolis (DRAM) and Hamiltonian Monte

Carlo (HMC), in which a transport model is replaced with a fast and accurate surrogate model in the form of a deep convolutional neural network (CNN). The CNN-based surrogate is trained on a small number of the transport model runs based on the prior knowledge of the unknown release history. Thus reduced computational cost allows one to diminish the sampling error associated with construction of the approximate likelihood function. As all MCMC strategies for source identification, our method has an added advantage of quantifying predictive uncertainty and accounting for measurement errors. Our numerical experiments demonstrate that our method's accuracy is comparable to that of MCMC with the forward transport model, but carries a fraction of the computational cost of the latter.

Finally, we expand the unknown parameter dimension and tackle a realistic three-dimensional inverse problem, in which both a heterogeneous conductivity field and the contaminant release history are identified from sparse observations. Achieving these two goals with limited and noisy hydraulic head and concentration measurements is notoriously challenging. The obstacles include the large dimensionality of the parameter space of such inverse problems and the high computational cost of a numerical solution to the PDEs describing fluid flow and solute transport in porous media. In this study, we use a convolutional adversarial autoencoders (CAAE) to parameterize heterogeneous non-Gaussian conductivity fields with a low-dimensional latent representation. Additionally, we train a three-dimensional dense convolutional encoder-decoder (DenseED) network to perform as the forward surrogate for the flow and transport processes. Combining the CAAE and DenseED forward surrogates, ensemble smoother with the multiple data assimilation (ESMDA) algorithm is used to sample from the Bayesian posterior distribution of the unknown parameters, forming our CAAE–DenseED–ESMDA inversion framework. We compare the inversion results of the CAAE-ESMDA with the physical flow and transport simulator with those of the CAAE–DenseED–ESMDA to demonstrate that the latter yields accurate reconstruction results at the small fraction of the cost of the former.

Acknowledgments

I would like to express my sincere gratitude to my advisor Professor Daniel Tartakovsky, for guiding me through the past five years of my life and research and for his strong support during the pandemics, when I was also facing challenges outside of school. The sentence he said 'We'll do what is the best for you' is something that I will look back to in my whole life. I would like to thank Professor Roland Horne and Professor Peter Kitanidis for kindly reviewing my thesis and providing suggestions. Due to the time zone difference, Professor Horne has to attend the defense at 7:00AM, to which I owe my appreciation. Thank you to Professor Eric Darve for serving as my nonreading committee member and the chair. Thank you to Professor Adam Brandt to reply to my urgent email requesting him to be the evaluator of my defense, it's "the charcoal in the snow" (an old saying in Chinese) to me. Their precious time is highly appreciated.

I want to thank Professor Koby Peng Wang who connected me with Daniel when I was in college and encouraged me to pursue a PhD career in the US, which changed the whole path of my life later on. I want to thank my collaborator Dr. Delphine Roubinet for the tremendous amount of help and effort on one of my research projects, and my former teammate Dr. David A. Barajas-Solano for providing instructions and suggestions. I want to thank Professor Nicolas Zabaras for the inspirations I got from our discussions on exploring new research topics and applications. I have had many research discussions and made very good friends with my teammates: Hannah, Francesca, Weiyu, Livia, Lama, Dong, Dimitrios, Anthony, Jo, Jun. My labmates in the ERE "fish tank": EJ, Dante, Jeff, Wennan, Rita, Ziyan have had many joyful small talks or research discussions with me that made many of my days. I have spent most of my spare time with my friends in Feiyu Chinese theater group directing and acting in plays, and got valuable career development help from some of them. There would have been a huge reduction of my happiness without them.

I want to thank my friends who were with me (virtually) during the nine-month self quarantine in 2020, without whom I don't know what life would be like. Thank Jake for packing my whole apartment and moving all my belongings around. Along this five-year journey of my graduate study, one year in San Diego and fours years in Stanford, every single time I flew back home (about ten times in total), my parents would be waiting in the airport to hand me a bouquet. I am still learning from them to be a more optimistic and stronger person. I feel lucky to be their child.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Inverse Problems and their Challenges

The number of unknown parameters in a given problem determine the dimensionality of the inverse model. The higher the model's dimensionality, the harder it is to solve. Many, if not all, inversion strategies scale poorly with the dimension. Inversion frameworks fall into two categories, deterministic and probabilistic. Regardless of whether physical/chemical/biological phenomena of interest are deterministic or stochastic, and regardless of which inversion framework is adopted, inverse modeling unavoidably requires a large number of forward model solves. In each solve, a realization of unknown parameters is used to obtain the model response, and the latter is compared with the measurements. The computational cost of an individual forward run can be so high as to render the cost of an accurate solution to a real-world inverse problem prohibitive.

As challenging as they might be, inverse problems are still tractable if two complementary strategies are integrated. The first aims to reduce the number of forward model runs that are required for an inversion algorithm to converge. The second aims to reduce the computational cost of each forward run by constructing an efficient surrogate model and/or to reparameterize the high-dimensional inputs.

1.2 Inversion Frameworks

Inverse problems are often ill-posed, as the available observations are sparse and sensitive to random measurement errors. The goal of reducing the number of forward simulations needed relies on the development of advanced inversion algorithms. Among them, deterministic or optimization-type methods (e.g., maximum likelihood estimators) look for a "best" estimate of the parameters, without attempting to quantify the predictive uncertainty inherent in such estimations. Rather than seeking to obtain a unique solution, probabilistic methods provide a distribution of such solutions, so that

each estimate comes with the associated probability of being correct or a confidence interval. This thesis deals with the latter type of inversion algorithms.

Among probabilistic inversion frameworks, Markov chain Monte Carlo (MCMC) and ensemble-based methods are two of the most prevalent alternatives in subsurface inverse modeling [105]. Both frameworks provide an approximation of the Bayesian posterior distribution of unknown parameters. A very beneficial feature of MCMC is its ability to handle general prior distributions of the parameters. However, even the mathematical properties of MCMC guarantee the convergence of a Markov chain of samples to the posterior distribution after a "sufficient" number of samples is assimilated, this number can easily exceed 10^5 in groundwater applications [143, 138]. Parallelization of MCMC [55] does not vastly improve its efficiency, because each Markov chain needs to be long enough for the "burn in" stage samples to be discarded. Search for MCMC variants with improved efficiency has led to the development of the delayed rejection (DR) sampling and the adaptive Metropolis (AM) sampling, which slightly increase the sample acceptance rate; the Hamiltonian Monte Carlo (HMC) sampling based on the gradient, which integrates a Hamiltonian physical system. In theory, HMC is applicable to high-dimensional problems, because the acceptance rate would reach 100% if the Hamiltonian dynamics are simulated properly. Yet, the use of the gradients required in this sampling method preclude its use in many high-dimensional applications. The ensemble-based methods include ensemble Kalman filter (EnKF) [31, 32], restart EnKF (rEnKf) [134], iterative local updating ensemble smoother (ILUES) [140], and ensemble smoother with multiple data assimilation (ESMDA). Such methods use a two-stage prediction/correction procedure to update the ensemble of the parameter realizations drawn from a prior distribution. ESMDA and ILUES have shown the ability to deal with nonlinear state-space models and high-dimensional problems [54, 94]. The "rule of thumb" in ESMDA or ILUES is to use an ensemble that consists of $\mathcal{O}(10^3)$ samples, and to update the ensemble for $\mathcal{O}(10)$ iterations. Because the forward simulations in each ensemble are uncorrelated, they are trivially parallelizable. This property, which is often referred to as "embarrassingly parallel" or "pleasingly parallel" [48], is a great advantage of these methods over MCMC. A geostatistical parameter estimation framework: quasilinear inversion [117, 62, 63] also falls in the type of inversion frameworks that require the gradient information, or referred to as sensitivity of measurements to unknowns, and hence, suffers from the same curse-of-dimensionality[37]. Adjoint-state sensitivity analysis enables the implementation of this method, but loses the attractiveness since the forward model can not be used as a "black-box". Eigenspectrum-based compression developed for geostatictical inversion methods [64, 75] could preserve the feasibility of a "black-box" forward model, and largely reduce the effort of the computation of these gradients with a matrix-free approach, yet is still constrained to the best linear unbiased estimation.

1.3 Surrogate Models

There are at least two ways to reduce the computational cost of a single forward model run. The first seeks to replace expensive forward PDE-based simulations of flow and transport processes with their cheap surrogate, emulator, or reduced-order model [19, 85, 84]. Examples of such surrogates include polynomial chaos expansions [138, 19] and Gaussian processes [29, 139]. These two approaches, and many others, suffer from the so-called curse of dimensionality, which refers to the degradation of their performance as the number of random inputs becomes large. Various flavors of deep neural networks (DNNs) have attracted attention, in part, because they remain robust for large numbers of inputs and outputs [95, 58]. Another benefit of DNNs is that their implementation in open-source software is portable and can be accelerated with advanced computer architectures, such as graphics processing units (GPUs) and tensor processing units (TPUs), without significant coding effort from the user. On the downside, unlike some other surrogate models (e.g., polynomial chaos, which can predict a solution at any time), most DNN surrogates of flow and transport models predict only the model states for a short period, and do not generalize to further predictions.

The second way of using surrogates, which is not directly related to a forward model, is to obtain an efficient parameterization of high-dimensional variables. This technique is often referred to as reduced-order modeling (ROM). DNN-based based ROM methods include autoencoders such as variational autoencoders [61] and adversarial autoencoders based on a generative adversarial network (GAN) [40]. In the context of subsurface modeling the resulting DNN learns the two-way mapping between a training conductivity field and the random latent variable. A realization of the latter can be decoded to a realization of the conductivity field that is similar to those from the training data set. Besides significantly reducing the number of the unknown parameters, variational autoencoders enable one to tackle the latent variable distribution. That is because the latter is usually formed as a standard normal distribution. This simplicity further facilitates the inverse problem solving with ensemble methods.

1.4 Overview of the Dissertation

This dissertation is organized as follows. In Chapter 2, we present a Bayesian inference strategy to estimate characteristics of a Discrete Fracture Network (DFN) from thermal experiments. A DNN surrogate is used to accelerate simulations of heat tracer migration, facilitating exploration of the parameter space. We show that prior knowledge about DFN properties sharpens their estimation, yielding a parameter-space region wherein they lie with high probability. This work was submitted to *Water Resources Research* and is available as a preprint on ArXiv [144].

In Chapter 3, a two-dimensional contaminant source identification problem is tackled with two MCMC variants, DRAM sampling and HMC sampling. We integrate a CNN surrogate model with the DRAM/HMC sampling to identify the contaminant release locations and strengths. DRAM

sampling of the forward flow and transport model, implemented in black-box simulators MODFLOW [47] and MT3DMS [141], is performed to obtain the benchmark results. These are used to test the performance of DRAM sampling and HMC sampling with the CNN surrogate. The comparison of the inversion accuracy and the computational cost is reported. Part of this work (DRAM sampling with physics-based model and CNN surrogate model) is published in *Stochastic Environmental Research and Risk Assessment* [143].

In Chapter 4, we tackle a three-dimensional problem of contaminant source identification in a heterogeneous subsurface environment whose hydraulic conductivity is unknown. The latter is parameterized with a convolutional adversarial autoencoder (CAAE), which reduces the dimensionality of the inverse problem, and maps its complex (non-Gaussian, possibly multimodal) distribution onto a standard normal distribution that is much easier to handle. A forward surrogate model with a convolutional dense encoder-decoder (DenseED) neural network architecture is used in place of the PDE-based model of subsurface flow and transport. Ensemble smoother with multiple data assimilation (ESMDA) is used to approximate the posterior distribution of 931 unknown parameters, including the conductivity field parameterized with CAAE, and the contaminant release history. We compare the accuracy and computational efficiency of the PDE-based CAAE–ESMDA inversion with that of the CAAE–DenseED–ESMDA inversion. This work is under preparation to be submitted to a peer-reviewed journal.

Chapter 2

Fractured Rock Delineation via Thermal Experiments

In this chapter, we aim to estimate two Discrete Fracture Network (DFN) characteristics, the fracture density and fractal dimension, from thermal experiments. Bayesian inference is conducted together with a DNN surrogate to alleviate the computational burden of the forward model of fluid flow and heat transfer.

2.1 Introduction

Characterization of fractured rock is a critical challenge in a wide variety of research fields and applications, such as extraction, management and protection of water resources. In fractured-rock aquifers, fractures can act as preferential flow paths that increase the risk of rapid contaminant migration over large distances. While the resource is generally stored in the surrounding matrix, fractures often determine the spatial extent of the extraction area (the cone of depression or well capture zone). Similar considerations play an important role in (oil/gas and geothermal) reservoir engineering, carbon sequestration, etc.

Various characterization techniques provide complementary information about fractured rocks. These typically rely on direct observation data, surface and borehole data acquired with geophysical techniques, and borehole data collected during hydraulic and tracer experiments [10, 24, 25, 23, 109]. We focus on the latter because they provide information that is directly related to the hydrogeological structures that drive flow and transport processes. For example, measurements of vertical flow velocities in a borehole under ambient and forced hydraulic conditions are used to estimate the properties of individual fractures that intersect the borehole [67, 100, 108], and piezometric data collected in observation boreholes allow one to evaluate features of complex fracture configurations [33, 73, 83].

Chemical tracer experiments, typically comprising the interpretation of breakthrough curves, yield information on the short and long paths in the fractured rock; these characterize the discrete fracture network (DFN) and matrix block properties, respectively [110, 45].

Heat has also been utilized to identify the presence of fractures intersecting boreholes [103, 106], to estimate their properties [65], and to study flow channeling and fracture-matrix exchange at the fracture scale [22, 66]. Most of these thermal experiments employ advanced equipment, which deploys the active line source (ALS) to uniformly modify water temperature in a borehole [102] and the distributed temperature sensing (DTS) to simultaneously monitor the resulting temperature changes in observation boreholes [106]. Thermal tracer experiments offer several advantages over their chemical counterparts. They do rely on neither localized multilevel sampling techniques nor localized tracer injection in boreholes; they interrogate larger area because heat conduction covers larger area than solute diffusion; and they are not restricted by environmental constraints whereas chemical tracers may remain in the environment for a long time [2, 104].

Without exception, the interpretation of hydraulic and tracer experiments involves inverse modeling. The choice of a strategy for the latter depends on the properties of interest, the data considered, the models available to reproduce the data, and the prior information about the studied environment. For fracture configurations between two boreholes, (semi-)analytical and numerical models can be applied to the cross-borehole flow-meter experiments mentioned above to evaluate the transmissivity and storativity of the fractures that intersect the boreholes at known depths [67, 100, 108]; the inversion consists of the gradient-based minimization of a discrepancy between the model's predictions and the collected data. Large-scale systems with complex fracture configurations require the use of sophisticated inversion strategies designed for high volumes of data. Most of such studies generate data via hydraulic and/or tracer tomography experiments, and use the inversion to identify the geometrical and hydraulic properties of a fracture network [33, 73, 119]. Studies attempting to infer the statistical characteristics of a DFN, such as fracture density and scaling exponents in distributions of length, orientation and aperture [52, 53] are limited. Electric potential and electrical resistivity data were used to identify the fractional connected area(FCA) in [88, 87], where the fractal dimension was also characterized, showing relatively high uncertainty of this parameter; yet fractal density was not identified due to the freedom of spatial distribution of the fractures.

Yet, more detailed statistics are necessary to quantify uncertainty in predictions of hydraulic and transport processes in fractured rocks. Their identification rests on ensemble-based computation, which involves repeated solves of a forward model. Two complementary strategies for making the inversion feasible for large, complex problems are: i) to reduce the number of forward solves that are necessary for the inversion algorithm to converge, and ii) to reduce the computational cost of an individual forward solve. The former strategy includes the development of accelerated Markov chain samplers, Hamiltonian Monte Carlo sampling, iterative local updating ensemble smoother, ensemble Kalman filters, and learning on statistical manifolds [7, 12, 11, 58, 143]. The latter strategy

aims to replace an expensive forward model with its cheap surrogate/emulator/reduced-order model [19, 85, 84]. Among these techniques, various flavors of DNNs have attracted attention, in part, because they remain robust for large numbers of inputs and outputs [143, 95, 58]. Another benefit of DNNs is that their implementation in open-source software is portable to advanced computer architectures, such as graphics processing units and tensor processing units, without significant coding effort from the user.

We combine these two strategies for ensemble-based computation to develop an inversion method, which makes it possible to infer the statistical properties of a fracture network from cross-borehole thermal experiments (CBTEs). To alleviate the high cost of a forward model of hydrothermal experiments, we use a meshless, particle-based method to solve the two-dimensional governing equations for fluid flow and heat transfer in DFNs (Section 2.2). These solutions, obtained for several realizations of the DFN parameters, are used in Section 2.3 to train a DNN-based surrogate. The latter's cost is so negligible as to enable us to deploy a fully Bayesian inversion (Section 2.4) that, unlike ensemble Kalman filter, does not require our quantity of interest to be (approximately) Gaussian. To do so, we generate two-dimensional synthetic fractal fracture network models in which hydrothermal experiments are simulated with fluid flow and heat transfer physically-based models. The computational burden of the forward models is overcome by determining neural network surrogate models that are used for inversion analysis. The inverse problem aiming at evaluating the fracture density and fractal dimension of the fracture networks from this data, is conducted with Bayesian inference, which is well suited for low-dimensional problems. The structural and physically-based forward models are presented in Section 2.2, the neural network model formulation in Section 2.3 and the inversion strategy in Section 2.4. Numerical experiments are conducted in Section 2.5, showing that our approach is four orders of magnitude faster than the equivalent inversion based on the physics-based model. These synthetic experiments also reveal that the CBTE data allow one to obtain accurate estimates of fracture density, while the inference of a DFN's fractal dimension is less robust. Main conclusions of this study are summarized in Section 2.6, together with a discussion of alternative strategies to improve the estimation of fractal dimension.

2.2 Models of Fracture Networks and Transport Phenomena

A forward model of CBTEs consists of a fracture network model and those of fluid flow and heat transfer. These models are described in Sections 2.2.1, 2.2.2, and 2.2.3, respectively.

2.2.1 Model of Fracture Networks

To be specific, we conceptualize a DFN via the fractal model of [50],

$$N_r = Cr^{-D}, \tag{2.1}$$

that postulates a power-law relationship between the number of fractures, N_r, and their dimensionless relative length r (normalized by smallest fracture length r_0), in a domain of characteristic length L. The parameters C and D denote fracture density and fractal dimension, respectively. If a network's smallest fracture has length r_0, then the number of classes in the fracture model is $N_f = \text{int}(C/r_0^D)$ and the relative length of fractures in the ith class is $r_i = (C/i)^{1/D}$ $(i = 1, \ldots, N_f)$. This formulation is equivalent to the model [21] that expresses fracture density $n(l, L) = \alpha L^{\mathcal{D}} l^{-a}$ in terms of fracture length l and domain size L, if one sets $\alpha = CD/N_f$, $\mathcal{D} = D$, and $a = D + 1$. The latter model reproduces self-similar structures observed in numerous studies [113, Chapter 6.6.8], allowing one to represent realistic fracture networks with the minimal number of parameters.

To generate a synthetic data set, we consider fractures arranged at two preferred angles $\theta_1 = 25°$ and $\theta_2 = 145°$ in a 100×100 m^2 domain. Fracture centers are randomly distributed over the whole domain, and their aperture is set to 5×10^{-4} m, as in [38]. The resulting DFN is simplified by removing the fractures that are not, directly or indirectly through other fractures, connected to the domain's perimeter. An example of such a DFN is shown in Figure 2.1. Fluid flow and heat transfer are modeled on this fracture network.

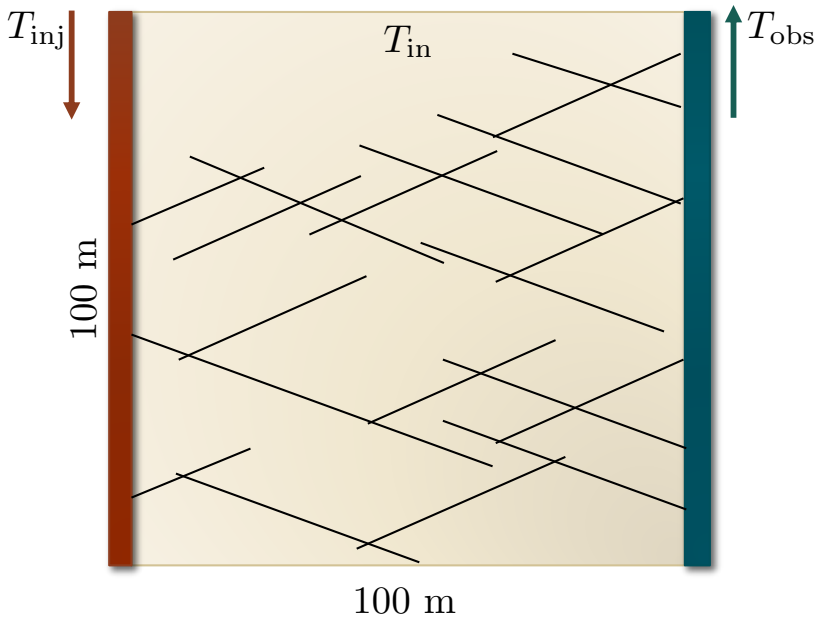

Figure 2.1: Fracture networks example. In a geothermal system, the red rectangle on the left of the domain represents the injection well, the right rectangle represents the extraction well. The injecting, extracting, and initial temperature are $T_{\text{inj}}, T_{\text{obs}}, T_{\text{in}}$ respectively.

2.2.2 Model of Fluid Flow in Fracture Networks

We deploy a standard model of single-phase steady-state laminar flow in a DFN, which assumes the ambient rock matrix to be impervious to fluid. The flow of an incompressible fluid is driven by a hydraulic head gradient, J, due to constant hydraulic heads imposed on the left and right boundaries, the top and bottom boundaries are impermeable.

The fracture extremities and intersections of the DFN, whose construction is detailed above, form the network nodes and a fracture connecting two adjacent nodes are referred to as the network edge. Flow rate in each edge is computed as the cross-sectional average of the Poiseuille velocity profile. Thus, the flow rate, u_{ij}, of flow from node i to node j is $u_{ij} = -b_{ij}^2/(12\nu)J_{ij}$, where ν is the fluid's kinematic viscosity, b_{ij} is the aperture of the fracture connecting the nodes i and j, and $J_{ij} = (h_j - h_i)/l_{ij}$ is the hydraulic head gradient between these nodes with l_{ij} the distance between these nodes. The hydraulic heads at the DFN nodes, h_i $(i = 1, 2, \dots)$, are computed as the solution of a linear system built by enforcing mass conservation at each node: $\sum_{k \in \mathcal{N}_i} b_{ki} u_{ki} = 0$, where $\mathcal{N}_i$ is the set of the neighboring nodes of node i (see, e.g., [38, 145] for details).

2.2.3 Model of Heat Transfer in Fractured Rock

The DFN constructed in Section 2.2.1 is further pruned by removing the edges representing the fractures with negligible flow velocities, e.g., $u_{ij} \leq 10^{-10}$ m/s used in the subsequent numerical experiments. This leads to removing the "dead-ends" of the fracture networks, which correspond to fracture segments for which one of the extremities has only one neighboring node. Convection in the resulting fracture network and conduction in the host matrix rock are modeled via the particle-based approach [38]. The computational cost of this method is significantly lower than that of its mesh-based alternatives because it discretizes only the fracture segments, while the matrix is not meshed. The particle displacement is associated with conduction and convection times in the fracture and the matrix, respectively. The latter time is defined from analytical solutions to a transport equation for fracture-matrix systems, and truncated according to the probability p_{lim} for the particle to reach a neighboring fracture by conduction through the matrix. The fracture segment discretization is defined through the parameter p_{lim} that denotes this probability. Complete mixing is assumed at the fracture intersections, implying that the probability for a particle to enter into a fracture depends only on the flow rate arriving at the considered node.

CBTEs are simulated by uniformly injecting N_{part} particles on the left side of the domain and recording their arrival times on the right side. The cumulative distribution functions (CDFs) of these arrival times describe the changes in the relative temperature T^* observed at distance L from the heat source, assuming complete mixing in the vertical direction at the observation position. The relative temperature is defined as $T^* = (T_{\text{obs}} - T_{\text{in}})/(T_{\text{inj}} - T_{\text{in}})$, where T_{in} is the initial (at $t = 0$) fluid temperature in the system, and T_{inj} and T_{obs} the temperature at the injection and observation positions, respectively [38].

2.3 Neural Network Model Formulation

2.3.1 Forward surrogate model

We define a NN surrogate for the physics-based model described in Section 2.2 with a map,

$$\mathbf{f} : (C, D) \to F(x), \quad F(x) = \mathbb{P}(X \leq x), \quad x \in \mathbb{R}, \tag{2.2}$$

where (C, D) are the fracture network parameters, and $F(x)$ is the CDF of a particle's arrival time X, i.e., the probability that X does not exceed a certain value x. Since the nonzero probability space of $F(x)$ varies for different simulations [38, 112, and Section 2.5 below], we find it convenient to work with the inverse CDF (iCDF) F^{-1}. Because any CDF is a continuous monotonically increasing function, the iCDF (or quantile CDF) is defined as:

$$\text{iCDF} : Q(p) = F^{-1}(p) = \min\{x \in \mathbb{R} : F(x) \geq p\}, \quad p \in (0, 1). \tag{2.3}$$

If $Q(p)$ is discretized into a set of N_k quantiles $\{p_1, \ldots, p_{N_k} : 0 < p_1 < \cdots < p_{N_k} < 1\}$, then

$$\text{iCDF} = \{Q(p_1), \ldots, Q(p_{N_k})\}, \qquad Q(p_1) < \cdots < Q(p_{N_k}). \tag{2.4}$$

2.3.2 Fully connected NNs

Consider a fully connected neural network (FCNN),

$$\mathbf{NN} : \mathbf{m} \xrightarrow{\text{FCNN}} \hat{\mathbf{d}} \tag{2.5}$$

that describes the forward surrogate model (2.2)–(2.4). The vector $\mathbf{m}$, of length N_m, contains the parameters to be estimated (in our problems, these parameters are C and D, so that $N_m = 2$); and the vector $\hat{\mathbf{d}}$, of length N_d, contains the discretized values of the iCDF computed with the model $\mathbf{NN}$. This model is built by defining an $N_d \times N_m$ matrix of weights $\mathbf{W}$, whose values are obtained by minimizing the discrepancy between the vectors $\hat{\mathbf{d}}$ and the vector $\mathbf{d}$ comprising the output of physics-based model from Section 2.2. Because the relationship between $\mathbf{m}$ and $\mathbf{d}$ is likely to be highly nonlinear, we relate $\mathbf{m}$ and $\hat{\mathbf{d}}$ via a nonlinear model $\hat{\mathbf{d}} = \sigma(\mathbf{W}\mathbf{m})$, in which the prescribed "activation" function $\sigma(\cdot)$ operates on each element of the vector $\mathbf{W}\mathbf{m}$. Commonly used activation functions include sigmoid functions (e.g., tanh) and the rectified linear unit (ReLU). The latter, $\sigma(s) = \max(0, s)$, is used in this study due to its proven performance in similar applications [1, 143, 94].

The nonlinear regression model $\hat{\mathbf{d}} = \sigma(\mathbf{W}\mathbf{m}) \equiv (\sigma \circ \mathbf{W})(\mathbf{m})$ constitutes a single layer in a NN. A (deep) FCNN model with N_l layers is constructed by a repeated application of the activation

function to the input,

$$\hat{\mathbf{d}} = \mathbf{NN}(\mathbf{m}; \boldsymbol{\Theta}) \equiv (\sigma_{N_l} \circ \mathbf{W}_{N_l-1}) \circ \ldots \circ (\sigma_2 \circ \mathbf{W}_1)(\mathbf{m}). \tag{2.6a}$$

The parameter set $\boldsymbol{\Theta} = \{\mathbf{W}_1, \ldots, \mathbf{W}_{N_l-1}\}$ consists of the weights $\mathbf{W}_n$ connecting the nth and $(n+1)$st layers with the recursive relationships:

$$\begin{cases} \mathbf{s}_1 = (\sigma_2 \circ \mathbf{W}_1)(\mathbf{m}) \equiv \sigma_2(\mathbf{W}_1\mathbf{m}), \\ \mathbf{s}_i = (\sigma_{i+1} \circ \mathbf{W}_i)(\mathbf{s}_{i-1}) \equiv \sigma_{i+1}(\mathbf{W}_i\mathbf{s}_{i-1}), \quad i = 2, \ldots, N_l - 2 \\ \hat{\mathbf{d}} = (\sigma_{N_l} \circ \mathbf{W}_{N_l-1})(\mathbf{s}_{N_l-2}) \equiv \sigma_{N_l}(\mathbf{W}_{N_l-1}\mathbf{s}_{N_l-2}). \end{cases} \tag{2.6b}$$

Here, $\mathbf{s}_i$ is the vector of data estimated in the ith layer; $\mathbf{W}_1$, $\mathbf{W}_i$ $(i = 2, \ldots, N_l - 2)$ and $\mathbf{W}_{N_l-1}$ are the matrices of size $d_1 \times N_m$, $d_i \times d_{i-1}$ and $N_d \times d_{N_l-2}$, respectively, and the integers d_i $(i = 1, \ldots, N_l - 2)$ represent the number of neurons in the corresponding inner layers of the NN. Sometime, bias parameters $\mathbf{b_i}$ with dimension $d_i \times d_{i-1}$ is added to $\mathbf{s_i}$ too, these $\mathbf{b}$ are included into $\times$ as fitting parameters. The fitting parameters $\boldsymbol{\Theta}$ are obtained by minimizing the discrepancy (or "loss function") $\mathcal{L}(\mathbf{d}_i, \hat{\mathbf{d}}_i)$ between $\hat{\mathbf{d}}$ and $\mathbf{d}$,

$$\boldsymbol{\Theta} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \sum_{i=1}^{N_{\text{data}}} \mathcal{L}(\mathbf{d}_i, \hat{\mathbf{d}}_i), \qquad \hat{\mathbf{d}}_i = \mathbf{NN}(\mathbf{m}_i; \boldsymbol{\Theta}), \tag{2.7}$$

where N_{data} is the number of forward runs of the physics-based model. We use the stochastic gradient descent optimizer [111] to carry out this step, which is commonly referred to as "network training".

2.3.3 Neural network model setting

A choice of the functional form of the loss function $\mathcal{L}$ affects a NN's performance. Measuring the goodness of the fitting for the fracture forward model requires a good choice of the loss function L used in Equation (2.7). This function computes the divergence between the predicted and computed TICDFs. Studies on measuring quantile divergence, especially for discrete inverse distribution, are scarce. Measures of the "difference" between probability distributions, such as the Kullback-Leibler (KL) divergence [69] and the Hellinger distance [72], might or might not be appropriate for inverse distributions. Thus, while the KL divergence is a popular metric in Bayesian inference [12] and generative NNs [61, 40], its asymmetry precludes its use in (2.7) as a distance. Consequently, we quantify the distance between two discrete distributions $P = (p_1, \ldots, p_{N_k})$ and $P' = (p'_1, \ldots, p'_{N_k})$

in terms of the Hellinger distance,

$$\mathcal{L}_{\mathrm{H}}(P,P') = \frac{1}{\sqrt{2}}\|\sqrt{P}-\sqrt{P'}\|_2 = \left(\frac{1}{2}\sum_{i=1}^{N_k}(\sqrt{p_i}-\sqrt{p_i'})^2\right)^{1/2}, \tag{2.8}$$

i.e., solve the minimization problem (2.7) with $\mathcal{L} \equiv \mathcal{L}_{\mathrm{H}}(Q,\hat{Q})$.

To reduce the training cost and improve the NN's performance, we specify additional features to refine the initial guess of input parameters. The relationships between the fractal DFN parameters in Section 2.2.1, suggest the choice of $C^{1/D}$, C^{-D} and CD (which are equal to $r_i i^{1/D}$, r_0/N_f^D and αN_f, respectively) and $1/D$ as extra input features. Given the pair of initial parameters (C,D), the resulting full set of parameters for the NN is:

$$\mathbf{m}_{\mathrm{NN}} = (C, D, C^{1/D}, C^{-D}, CD, 1/D)^\top. \tag{2.9}$$

2.4 Inversion via Bayesian Update

2.4.1 Theoretical background

According to the Bayes rule, the posterior probability density function (PDF) $f_{\mathbf{m}|\mathbf{d}}$ of the parameter vector $\mathbf{m}$ is computed as:

$$f_{\mathbf{m}|\mathbf{d}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}}) = \frac{f_{\mathbf{m}}(\tilde{\mathbf{m}})f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}})}{f_{\mathbf{d}}(\tilde{\mathbf{d}})}, \qquad f_{\mathbf{d}}(\tilde{\mathbf{d}}) = \int f_{\mathbf{m}}(\tilde{\mathbf{m}})f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}})\mathrm{d}\tilde{\mathbf{m}}, \tag{2.10}$$

where $\tilde{\mathbf{d}}$ and $\tilde{\mathbf{m}}$ are the deterministic coordinates of random variable $\mathbf{d}$ and $\mathbf{m}$, respectively; $f_{\mathbf{m}}$ is the prior PDF of $\mathbf{m}$; $f_{\mathbf{d}|\mathbf{m}}$ is the likelihood function (i.e., the joint PDF of the measurements conditioned on the model predictions, which is treated as a function of $\mathbf{m}$); and the normalizing factor $f_{\mathbf{d}}$ ensures that $f_{\mathbf{m}|\mathbf{d}}$ integrates to 1.

We take the likelihood function $f_{\mathbf{d}|\mathbf{m}}$ to be Gaussian,

$$f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}}) = \frac{1}{\sigma_{\mathrm{d}}\sqrt{2\pi}}\exp\left[-\frac{1}{2}\frac{L_H(\tilde{\mathbf{d}},\mathbf{g}(\tilde{\mathbf{m}}))}{\sigma_{\mathrm{d}}^2}\right]. \tag{2.11}$$

This PDF has the standard deviation σ_{d} and is centered around the square root of the Hellinger distance between the data $\tilde{\mathbf{d}}$ predicted by the likelihood and the data $\mathbf{g}(\tilde{\mathbf{m}})$ provided by the forward model $\mathbf{g}$. Addition of prior knowledge of $\mathbf{m}$ to the likelihood function is done within the standard Bayesian framework by assuming that the prior PDF is as important as the data. We explore how the posterior PDF can be improved by adjusting the impact of the prior. To do so, we treat the latter as a regularization term with a tunable hyperparameter γ that corresponds to the weight associated with the prior, enabling us to reduce the impact of the prior when its knowledge does not

seem to be persuasive. The resulting posterior PDF is formulated as

$$f_{\mathbf{m}|\mathbf{d}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}}) \propto \mathrm{e}^{-H(\tilde{\mathbf{m}})}, \qquad H(\tilde{\mathbf{m}}) = H_{\mathrm{obs}}(\tilde{\mathbf{m}}) + \gamma H_{\mathrm{reg}}(\tilde{\mathbf{m}}), \tag{2.12}$$

where $H_{\mathrm{obs}}(\tilde{\mathbf{m}}) = -\ln(f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}}))$ and $H_{\mathrm{reg}}(\tilde{\mathbf{m}}) = -\ln(f_{\mathbf{m}}(\tilde{\mathbf{m}}))$ are the negative log-likelihood and log-prior distributions, respectively. This yields:

$$f_{\mathbf{m}|\mathbf{d}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}}) \propto f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}})\,(f_{\mathbf{m}}(\tilde{\mathbf{m}}))^{\gamma}, \quad \gamma \in [0,1]. \tag{2.13}$$

This posterior PDF is computed via the following algorithm.

2.4.2 Numerical Implementation

The efficiency of the neural network model that is previously defined enables us to implement a robust inverse analysis by computing the posterior distribution of a large number of inputs. The corresponding Monte Carlo algorithm is implemented as follows. The posterior distribution previously defined is computed with the following Monte Carlo algorithm:

1. The domains $\mathcal{C}$ and $\mathcal{D}$ of values for the parameters C and D are discretized with N_C and N_D nodes, respectively. The result is a $N_C \times N_D$ regular grid for the parameter pair (C, D) with coordinate vectors $\mathbf{m}_{ij} = (C_i, D_j)^\top$ $(i = 1, \ldots, N_C,\, j = 1, \ldots, N_D)$.

2. The iCDFs (2.4) are computed with the forward model $\mathbf{g}$ for all pairs $\mathbf{m}_{ij}$.

3. The negative log-likelihood $H_{\mathrm{obs}}(\mathbf{m}) = -\ln(f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}}))$ is computed via (2.11), with the data $\mathbf{g}(\mathbf{m})$ provided by model $\mathbf{g}$ in Step 2.

4. The posterior PDF $f_{\mathbf{m}|\mathbf{d}}$ is computed via (2.13) by adjusting the weight γ assigned to the prior knowledge. (The case $\gamma = 0$ corresponds to a uniform prior for $\mathbf{m}$, where the unnormalized posterior PDF is equivalent to the likelihood.)

This brute-force implementation of Bayesian inference is only made possible by the availability of the FCNN surrogate, whose forward runs carry virtually zero computational cost. In its absence, or if the number of unknown parameters were large, one would have to deploy more advanced Bayesian update schemes such as Markov chain Monte Carlo [143, 7] or ensemble updating methods [94, 95].

2.5 Numerical experiments

The synthetic generation of DFNs and breakthrough times, t_{break}, for a heat tracer is described Section 2.5.1. Generation of the data for CNN training is described in Section 2.5.2, with the construction of a CNN surrogate for the PDE-based model (Section 2.2) reported in Section 2.5.3.

In Sections 2.5.4 and 2.5.5, we use this surrogate to accelerate the solution of the inverse problem of identifying the DFN properties from the breakthrough-time data.

2.5.1 Synthetic Heat-Tracer Experiment

Our synthetic heat tracer experiment consists of injected hot water with temperature T_{inj} at the inlet ($x_1 = 0$) and observing temperature changes at the outlet ($x_1 = L$). The goal is to infer the statistical properties of a DFN, C and D, from a resulting breakthrough curve. A fracture network with known values of C and D serves as ground truth, with possible measurement errors neglected. Consistent with [38], we set the externally imposed hydraulic gradient across the simulation domain to $J = 0.01$ and the thermal diffusion coefficient in the matrix to $D_{\text{therm}} = 9.16 \times 10^{-7}$ m^2/s.

2.5.2 Generation and Analysis of Synthetic Data

To generate data for the CNN training and testing, we considered the fracture networks (2.1) with $C \in [2.5, 6.5]$ and $D \in [1.0, 1.3]$. These parameter ranges are both observed experimentally [89, 115] and used in previous numerical studies [38, 126]. The parameter space $[2.5, 6.5] \times [1.0, 1.3]$ was uniformly discretized into $N_{\text{sim}} = 10^4$ nodes, i.e., pairs of the parameters $(C, D)_i$ with $i = 1, \ldots, N_{\text{sim}}$. The number of injected particles, N_{part}, representing the relative temperature of the injected fluid during a CBTE, T_{inj}, varied between 10^2 and 10^4.

Table 2.1 below gives the computational times for estimating the TCDFs of one random fracture network realization over 10^4 pairs of the parameters (C, D). The table shows that the average time required to perform one simulation is smaller than one second when the number of particles is set to 10^2 (*Simu1*, *Simu4*, *Simu7* and *Simu10*) or when the fractures are not discretized (*Simu1*, *Simu2* and *Simu3*).

In addition to N_{part}, the simulation time and accuracy of each forward model run are largely controlled by the number of elements used to discretize a fracture, which is defined by the parameter p_{lim} introduced in Section 2.2.3. The simulation time t_{sim} refers to the time (in seconds) it takes to estimate the CDF of breakthrough times for one random DFN realization and one of the $N_{\text{sim}} = 10^4$ pairs of the parameters (C, D). We found the average t_{sim} not to exceed 1 s if either $N_{\text{part}} = 100$ or the fracture is not discretized (Table 2.1); the average is over 20 random realizations of the DFN obtained with different random seeds for each parameter pair (C, D).

Representative CDFs of breakthrough times of N_{part} particles, in each of these 20 DFN realizations, are displayed in Figure 2.2 and Figure 2.3 for six pairs of the DFN parameters (C, D). The across-realization variability of the CDFs is more pronounced for $N_{\text{part}} = 10^2$ then 10^3 particles, and visually indistinguishable when going from $N_{\text{part}} = 10^3$ to 10^4 particles(shown in Figure 2.4). Likewise, no appreciable differences between the CDFs computed with $p_{\text{lim}} = 0.5$ and 0.2 were observed. Finally, when the random-seed effects are averaged out, the resulting breakthrough-time CDFs for $N_{\text{part}} = 10^2$ and 10^3 are practically identical (Figure 2.5). Based on these findings, in the

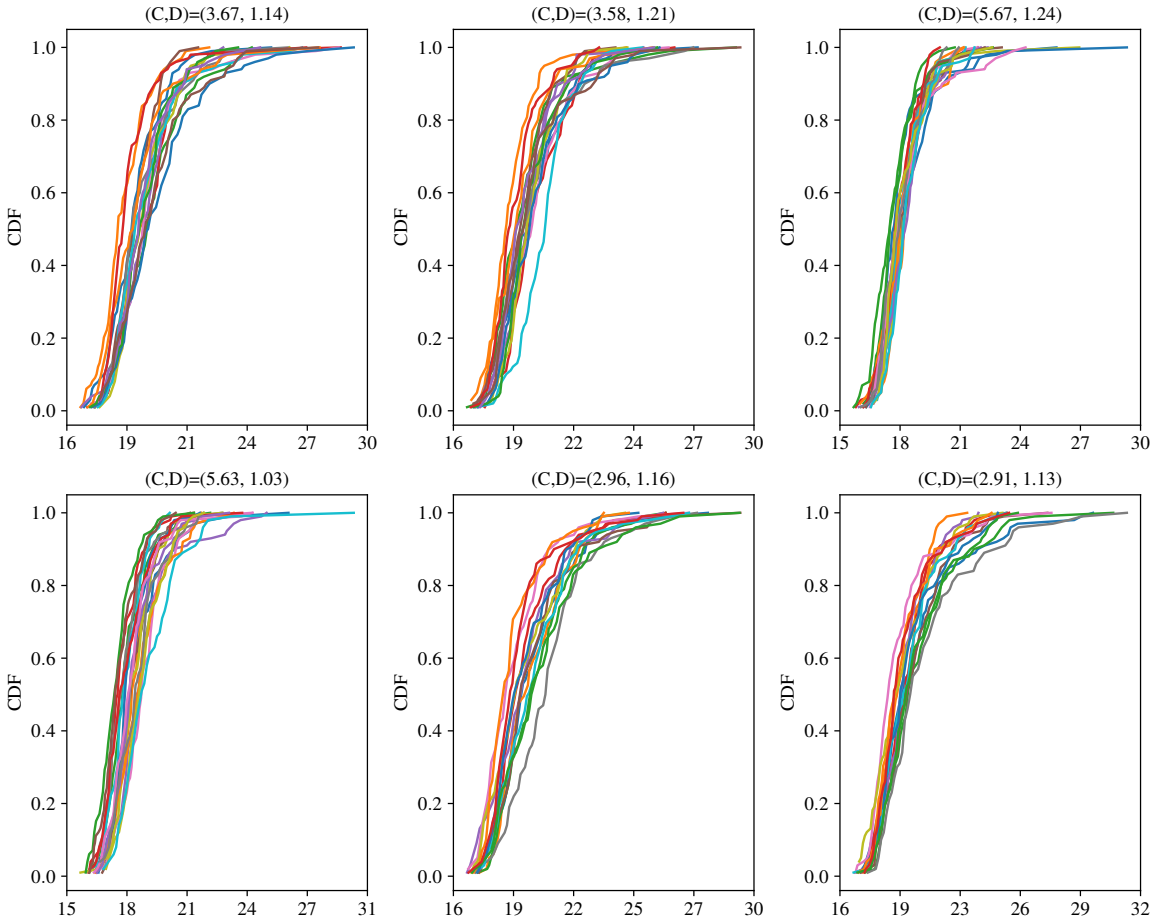

Figure 2.2: Representative CDFs of the logarithm of breakthrough times (in seconds) of $N_{\text{part}} = 100$ particles, $F(\ln t_{\text{break}})$, for 20 realizations of the DFN characterized by a given combination of the DFN parameters (C, D). Each colored curve corresponds to a different random realization, in all simulations, we set $p_{\text{lim}} = 0.5$. Corresponding to *Simu7* in Table 2.1.

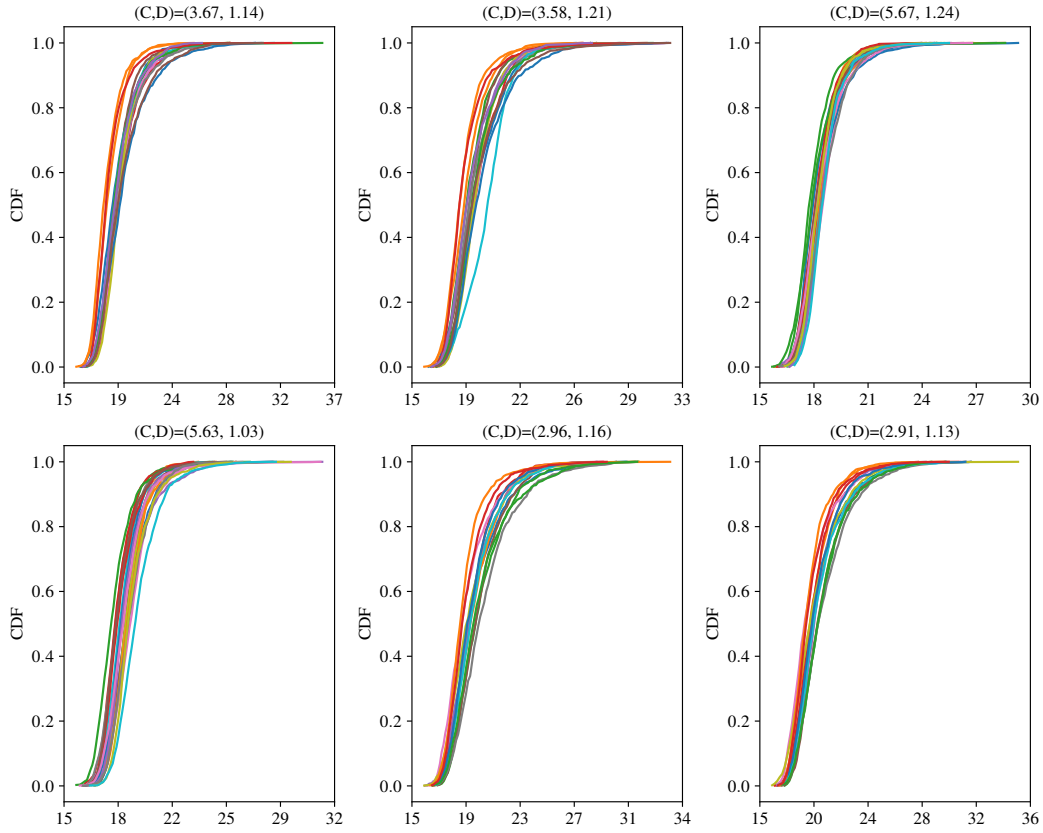

Figure 2.3: Representative CDFs of the logarithm of breakthrough times (in seconds) of $N_{\text{part}} = 1000$ particles, $F(\ln t_{\text{break}})$, for 20 realizations of the DFN characterized by a given combination of the DFN parameters (C, D). Each colored curve corresponds to a different random realization, in all simulations, we set $p_{\text{lim}} = 0.5$.Corresponding to *Simu8* in Table 2.1.

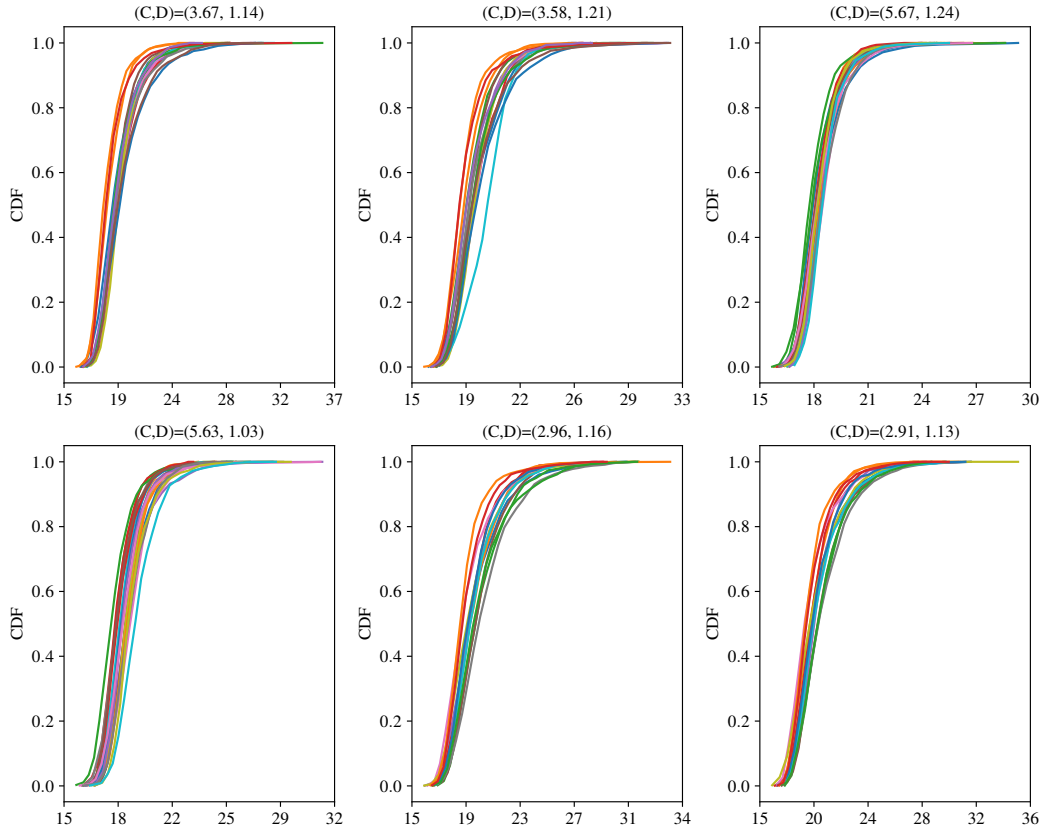

Figure 2.4: Representative CDFs of the logarithm of breakthrough times (in seconds) of $N_{\text{part}} = 10000$ particles, $F(\ln t_{\text{break}})$, for 20 realizations of the DFN characterized by a given combination of the DFN parameters (C, D). Each colored curve corresponds to a different random realization, in all simulations, we set $p_{\text{lim}} = 0.5$. Corresponding to *Simu9* in Table 2.1.

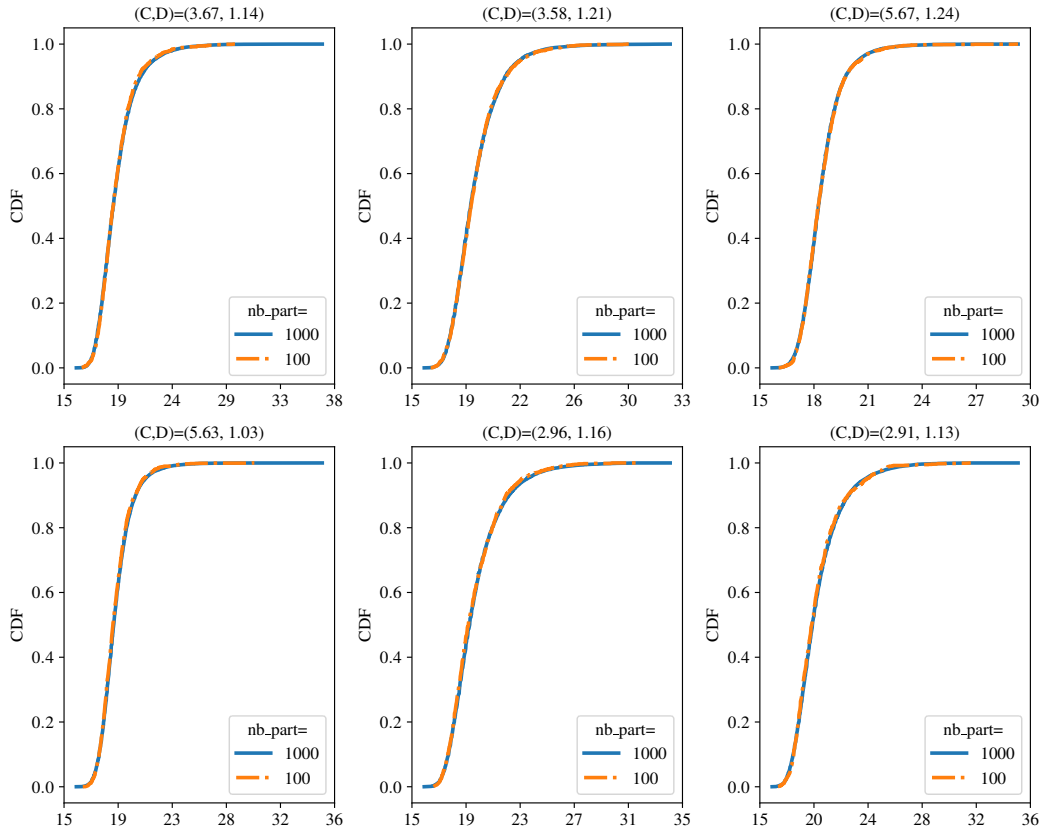

Figure 2.5: Mean CDFs of the logarithm of breakthrough times (in seconds) of N_{part} particles, $F(\ln t_{\text{break}})$, averaged over the corresponding DFN realizations in Figure 2.2 and Figure 2.3.

Simulation name	N_{part}	p_{lim}	Computational time
Simu 1	10^2	-	3.232
Simu 2	10^3	-	3.294
Simu 3	10^4	-	5.702
Simu 4	10^2	0.9	5.301
Simu 5	10^3	0.9	17.365
Simu 6	10^4	0.9	100.573
Simu 7	10^2	0.5	6.560
Simu 8	10^3	0.5	25.676
Simu 9	10^4	0.5	191.739
Simu 10	10^2	0.2	9.804
Simu 11	10^3	0.2	56.483
Simu 12	10^4	0.2	420.883

Table 2.1: Computational times, in seconds, over 10^4 simulations for various values of the number of particle (N_{part}) and the fracture discretization ($p_{\text{lim}} = 1/N_{\text{dis}}$). When p_{lim} is not defined ($N_{\text{dis}} = 0$), the fracture is not discretized and the results rely on the assumption of an infinite surrounding ambient matrix [107, 110]. When p_{lim} is defined, decreasing the value of this parameter results in improving the fracture discretization by reducing the size of the discretized fracture elements.

subsequent simulations, we set $N_{\text{part}} = 100$ and $p_{\text{lim}} = 0.5$ in order to obtain an optimal balance between the computational time and accuracy.

For some parameter pairs (C, D), not every DFN realization (defined by the random seed) hydraulically connects the injection and observation boundaries. The number of displayed CDFs is smaller than 20 because the physically-based transport model could not perform for all the 20 random fracture networks. This is usually due to the presence of not connected fracture networks for which the fluid flow distribution is not defined. Such hydraulically disconnected networks are not suitable for our flow model (see Section 2.2.2). However, in our numerical experiments, there were at least 10—and, in the majority of cases, 20—connected fracture networks for each (C, D) pair shown in Figure 2.7.

The final step in our data generation procedure consists of converting the estimated CDFs F into corresponding iCDFs F^{-1} (Figure 2.6). The latter form the data set $\mathbf{d}$, different parts of which are used to train a CNN and to verify its performance.

Before any data is seen, we perform a qualitative sensitivity analysis to gain some presumption on the inverse modeling results. The sensitivity analysis here are to inspect the injectivity of the model:

$$\begin{gathered} f : X \to Y, \\ \text{injectivity of } f : \forall a, b \in X, \quad f(a) = f(b) \Rightarrow a = b \end{gathered} \tag{2.14}$$

It is observed from the obtained dataset that the function mapping fracture parameters (C, D) and the particle arrival time density function is not injective. Hence performing inverse analysis

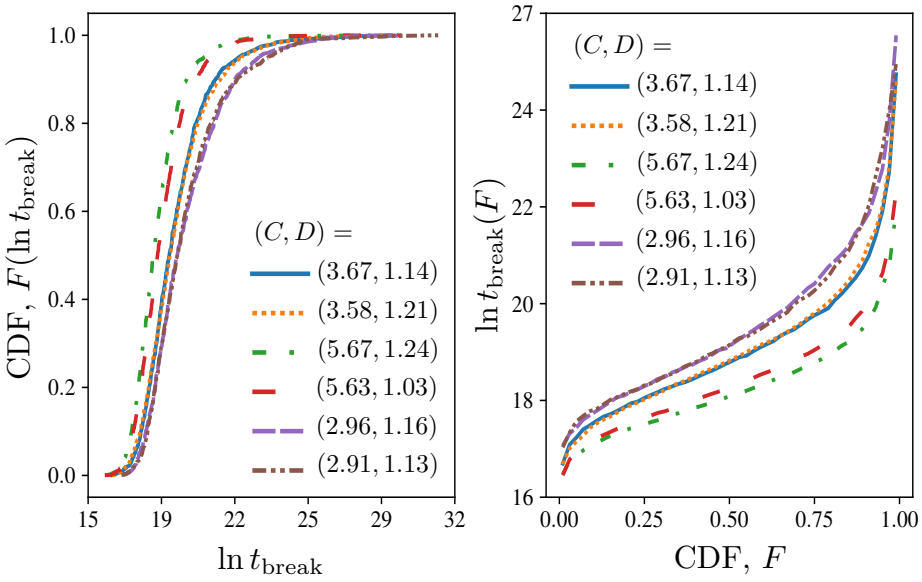

Figure 2.6: CDFs (left) and corresponding iCDFs (right) of the thermal breakthrough times for a single realization of the six DFNs characterized by six pairs of the parameters (C, D).

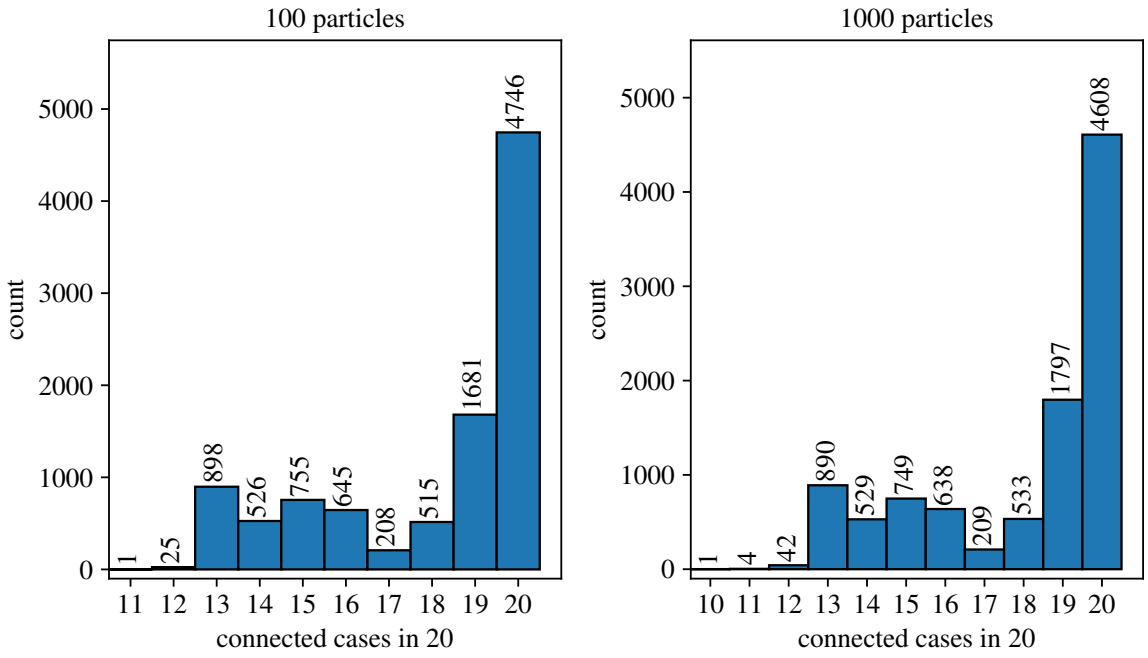

Figure 2.7: Histogram of connected and successfully simulated realizations over 20 realizations. Left: simulations with $N_{\text{par}} = 100$, Right: simulations with $N_{\text{par}} = 100$. This information is used to construct the prior distribution of the parameters (C, D) in Figure 2.15.

would lead to ambiguous results. Figure 2.8 illustrates such non-injectivity, as the cases which result in very similar output are distributed in a big area in the input space.

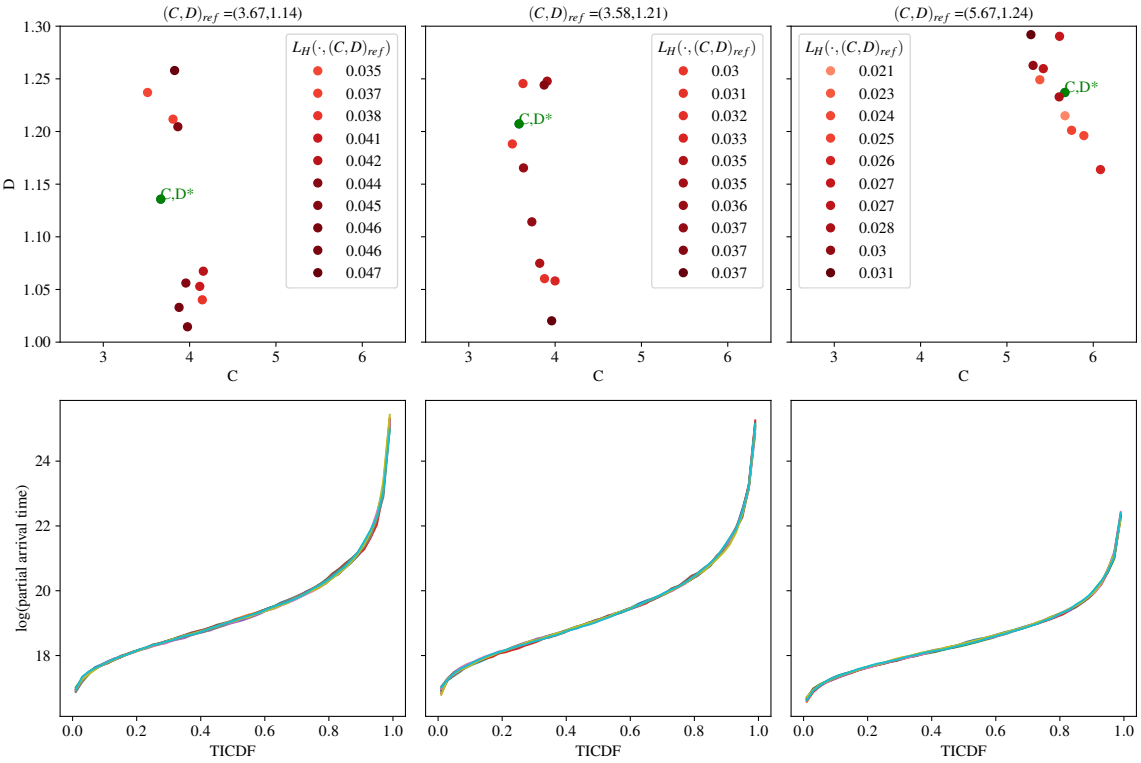

Figure 2.8: Three examples showing that the considered problem is not injective, i.e., different sets of the parameters (C,D) lead to similar CDFs. In the figures in the first row, the green dot indicates the reference cases, the red dots are the C,D pairs which result in the most similar CDFs with the reference case, the lighter the color, the closer the CDFs.

It is interesting to notice from Figure 2.8 that although the forward problem is ill-posed, the output depends on C more than D, i.e. the model is more sensitive to C. This property of the forward model can be later used in a correlated case where C and D are not independent, s.t. the inverse analysis is less ill-posed.

2.5.3 FCNN Training and Testing

The data generated above are arranged in a set $\{\mathbf{m}_{\mathrm{NN}_i}, \mathbf{d}_i\}_{i=1}^{N_{\mathrm{sim}}}$ with $N_{\mathrm{sim}} = 10^4$ and $\mathbf{m}_{\mathrm{NN}}$ defined in (2.9). We randomly select $8 \cdot 10^3$ of these pairs to train the FCNN $\mathbf{NN}$ in (2.5), leaving the remaining $2 \cdot 10^3$ for testing. The output data $\mathbf{d}$ come in the form of iCDFs, i.e., nondecreasing series of numbers. Since a NN model is not guaranteed to reproduce this trend, we use the hyperparameter tuning method [78] to perform the search in the hyperparameter space specified in Table 2.2.

Parameter name	Search region
Number of layers	$\mathcal{U}\{3, 4, 5, 6\}$
Number of neurons	$\mathcal{U}\{2^2, 2^3, ..., 2^9\}$
Optimizer name	$\mathcal{U}\{\texttt{rms}, \texttt{sgd}, \texttt{ada}, \texttt{adam}\}$
Learning rate, l_r	$\log_{10}(l_r) \sim \mathcal{U}[-4, -2]$

Table 2.2: Hyperparameter search space defined by the number of layers, the number of neurons in each layer, the optimizer names, and (logarithm of) the learning rate. These parameters are uniformly sampled from either a discrete set of values, $\mathcal{U}\{\cdot, \cdot, \ldots, \cdot\}$, or an interval, $\mathcal{U}[\cdot, \cdot]$. The RMSprop optimizer [41, 49], `rms`; the stochastic gradient descent optimizer [121], `sgd`; the Adagrad optimizer [27] `ada`; and the Adam optimizer [60], `adam`, slightly differ from each other when performing the parameter gradient descent during the NN training.

The hyperparameter search involved 2500 trials; in each trial, the subset of data $\{\mathbf{m}_{\mathrm{NN}_i}, \mathbf{d}_i\}_{i=1}^{8000}$ were randomly split into a training set consisting of 6400 pairs $\{\mathbf{m}_{\mathrm{NN}_i}, \mathbf{d}_i\}$ and a validation set comprising the remaining 1600 pairs $\{\mathbf{m}_{\mathrm{NN}_i}, \mathbf{d}_i\}$. For each epoch, the 6400 training pairs were used to optimize the NN parameters, and the NN accuracy is evaluated on the validation set. Each trial used one of the optimizers in Table 2.2 for at most 10^3 epochs; the trial was stopped if the validation loss did not decrease for 10^2 epochs. After completion of all the trials with these rules, the trial with the smallest validation loss was saved. The optimal FCNN, described in Table 2.3, has 6 layers between the input and output layers and is obtained using the Adam optimizer with the Adam optimizer coefficients $\beta = (0.9, 0.999)$ to perform gradient descent. This trial is associated with a learning rate $l_r = 0.00403$ and the averaged Hellinger loss of 0.0827 on the validation set. This FCNN was further trained with a learning rate that reduces on plateau of the validation performance to further fine-tune the model parameters for another 10^3 epochs; the ending testing Hellinger loss is 0.0652 and the total training time is 37340 seconds. Figure 2.9 depicts the FCNN predictions of the iCDFs of the particle breakthrough times in DFNs characterized by different parameter-pairs (C, D) not used for training. These predictions are visually indistinguishable from those obtained with the physics-based model $\mathbf{g}(\mathbf{m})$ described in Section 2.2.1.

Layer	Weights	Bias	Layer output
Input	-	-	6
FC_1	$\mathbf{W}_1 : 256 \times 6$	$\mathbf{b}_1 : 256$	$\mathbf{s}_1 : 256$
FC_2	$\mathbf{W}_2 : 64 \times 256$	$\mathbf{b}_2 : 64$	$\mathbf{s}_2 : 64$
FC_3	$\mathbf{W}_3 : 512 \times 64$	$\mathbf{b}_3 : 512$	$\mathbf{s}_3 : 512$
FC_4	$\mathbf{W}_4 : 256 \times 512$	$\mathbf{b}_4 : 256$	$\mathbf{s}_4 : 256$
FC_5	$\mathbf{W}_5 : 32 \times 256$	$\mathbf{b}_5 : 32$	$\mathbf{s}_5 : 32$
FC_6	$\mathbf{W}_6 : 128 \times 32$	$\mathbf{b}_6 : 128$	$\mathbf{s}_6 : 128$
Output	$\mathbf{W}_7 : 50 \times 128$	$\mathbf{b}_7 : 50$	50

Table 2.3: The best-trial NN architecture consists of six hidden layers, FC_i $(i = 1, \ldots, 6)$, with the corresponding weight matrix $\mathbf{W}_i$ and layer output $\mathbf{s}_i$ $(i = 1, \ldots, 6)$ in (2.6). Bias parameters $\mathbf{b_i}$ with dimension $d_i \times d_{i-1}$ is added to $\mathbf{s_i}$ too, these $\mathbf{b}$ are included into $\boldsymbol{\Theta}$ as fitting parameters.

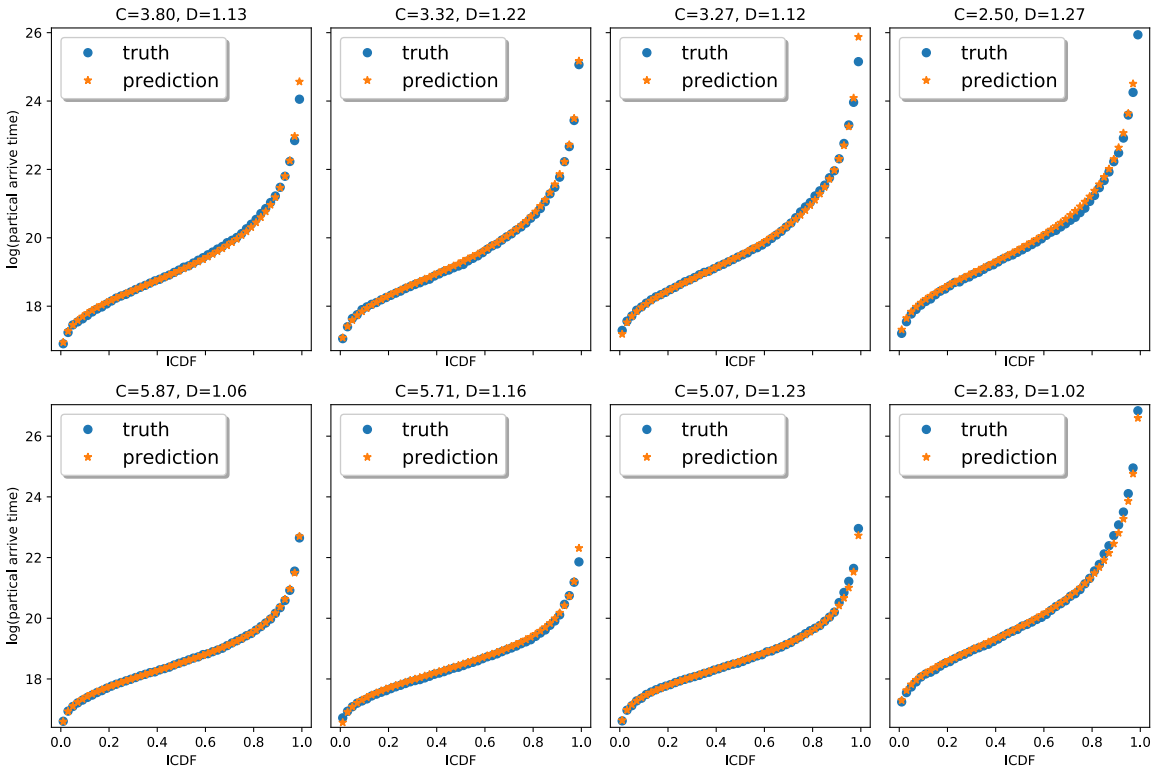

Figure 2.9: Physics-based and FCNN predictions of the iCDFs of the particle breakthrough times in DFNs characterized by different parameter-pairs (C, D) not used for training. 'truth' is obtained with the physics-based iCDFs.

2.5.4 Bayesian Inversion without Prior Information

We start with the Bayesian data assimilation and parameter estimation from Section 2.4. Taking the uniform prior, $\gamma = 0$ in (2.13), and assimilating the $N_{\text{sim}} = 10^4$ candidates provided by the physics-based model $\mathbf{g}$, this procedure yields the posterior PDFs of C and D shown in Figure 2.10 and Figure 2.11. While this noninformative prior indicates that all values of the parameters (C, D) are equally likely, the sharpened posterior correctly assigns higher probability to the region containing the reference (C, D) values. The relatively small number ($N_{\text{sim}} = 10^4$) of the forward solves of the physics-based model $\mathbf{g}$ manifests itself in granularity of the posterior PDF maps.

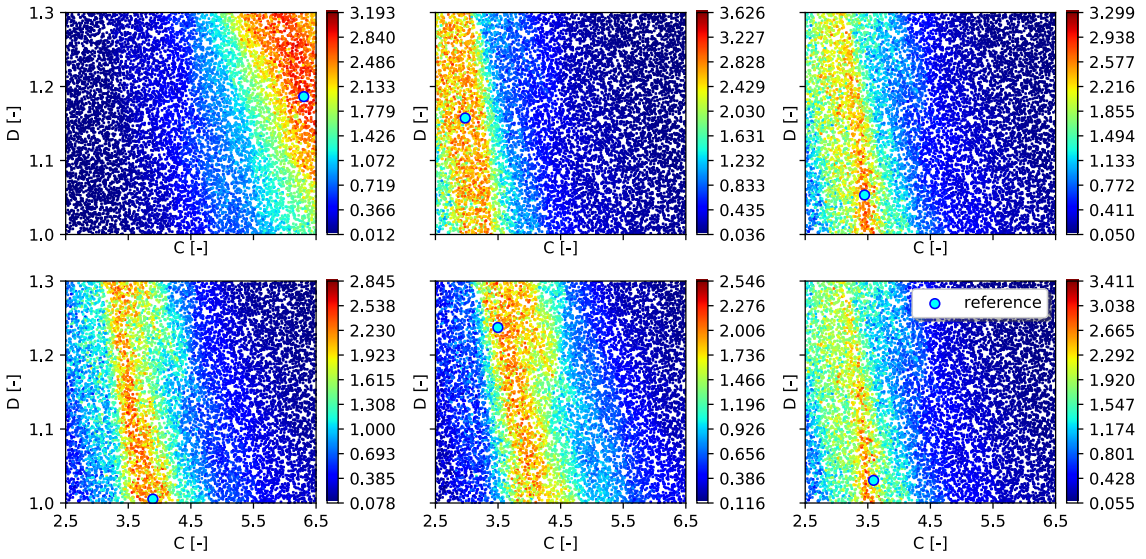

Figure 2.10: Examples of posterior PDFs of the DFN parameters C and D, for six experiments defined by the reference parameter values (blue circles). These PDFs are computed via Bayesian assimilation of either 10^4 runs of the physics-based model

Significantly more forward model runs are needed to further sharpen these posterior PDFs around the true values of (C, D) and to reduce the image pixelation. Generating the significant amounts of such data with the physics-based model is computationally prohibitive. Instead, we use 10^7 additional candidates, corresponding to a $10^4 \times 10^3$ mesh of the parameter space, provided by the FCNN surrogate. Figure 2.10 demonstrates that assimilation of these data (forward runs of the cheap FCNN surrogate) further reduces the band containing the unknown model parameters (C, D) with high probability. These distributions are improved as illustrated in Figure 2.11 by using data points, and resulting in finer distributions that are consistent with the initial distributions shown in Figure 2.13, 2.14. Generation of such large data sets with the physics-based model is four orders of magnitude more expensive than that with the FCNN. The availability of a NN surrogates makes a difference between being able to solve this inverse problem or not.

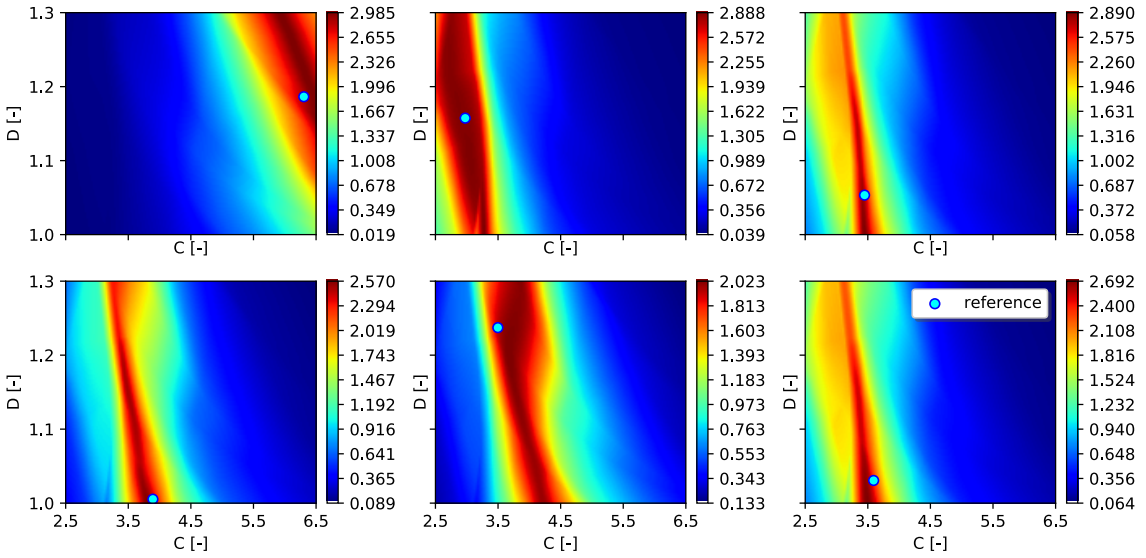

Figure 2.11: Examples of posterior PDFs of the DFN parameters C and D, for six experiments defined by the reference parameter values (blue circles). These PDFs are computed via Bayesian assimilation of additional 10^7 runs of the FCNN surrogate

In addition to improving the quality of the posterior distributions, the computational times reported in Table 2.4 show that using the surrogate **NN** model results in reducing by a factor 4 the inverse analysis time, which is not feasible with the physics-based model **g** .

	N_{sim}	T_{train}	T_{run}	T_{grid}	T_{tot}
$\mathbf{g}(\mathbf{m})$	2×10^8	0	$1.312 \cdot 10^8$	5.47	$1.312 \cdot 10^8$
$\mathbf{NN}(\mathbf{m})$	10^7	37340	1.26	5.47	$3.735 \cdot 10^4$

Table 2.4: Computational cost of the Bayesian inversion using the physics-based model $\mathbf{g}(\mathbf{m})$ or the FCNN surrogate $\mathbf{NN}(\mathbf{m})$. Each inversion requires N_{sim} forward runs and takes time T_{tot}. The latter comprises time to train the model (T_{train}), time to execute the forward runs (T_{run}) and time to define the posterior PDF on the discretized parameter grid (T_{grid}). The running time for $\mathbf{g}(\mathbf{m})$ is a projection based on the simulation time of 6560 seconds that was necessary to run 10^4 simulations. The FCNN was trained and executed on GPUs provided by GoogleColab. All times are in seconds.

The posterior PDFs displayed in Figure 2.10 show that the fracture density C is well constrained and amenable to our Bayesian inversion, whereas the inference of the fractal dimension D is more elusive. Examples of the DFNs in this study are provided in Figure 2 of [38]. They suggest that, for the parameter ranges considered, C impacts the spatial extent of a fracture network, while D affects the fracture-length distribution. Consequently, C has a more significant impact on the overall structures.

2.5.5 Bayesian Inversion with Data-informed Priors

To refine the inference of parameters C and D from the breakthrough-time CDFs, we add some prior information. First, we observe that the field data reported in Section 2.5.6 suggest that C and D are correlated. These data are fitted with a shallow feed-forward NN resulting in the prior PDF of C and D shown in Figure 2.12. These data vary over larger ranges than those used for C and D in the previous section; at the same time, most values correspond to $C < 2$. That is because the field data come from a large number of different sites and from direct outcrop observations. Figure 9 in [126] shows that a network with $C < 2$ would have low connectivity. On the other hand, a DFN with a large D is very dense, requiring large computational times to simulate and, possibly, being amenable to a (stochastic) continuum representation. Driven by these practical considerations, and to ascertain the value of this additional information, we restrict the prior PDF from Figure 2.12 to the same range of parameters as that used in the previous section.

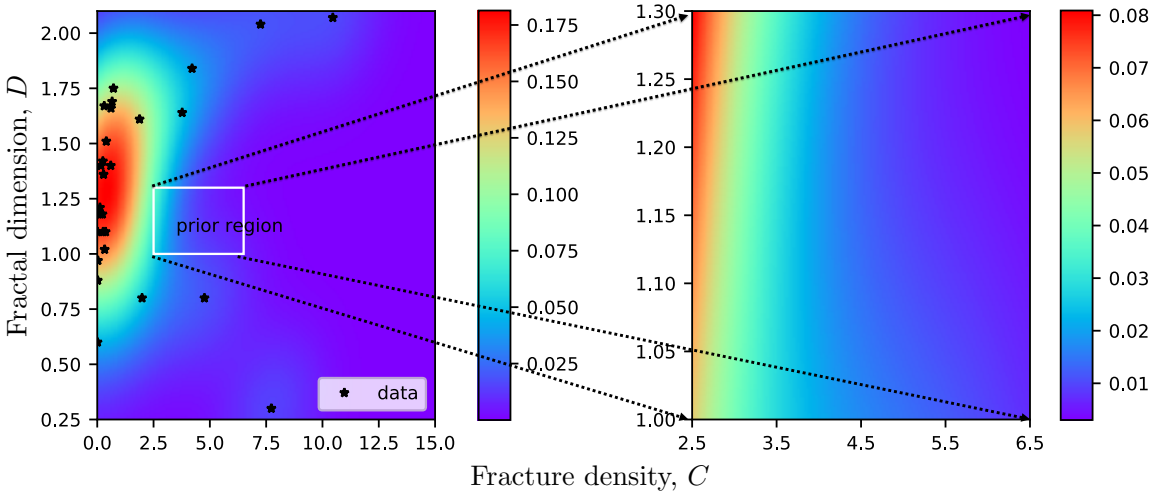

Figure 2.12: Prior joint PDF of C and D inferred from the field-scale data in Section 2.5.6 (left) and its rescaled counterpart over the parameter range used in our study (right).

The relative importance given to the prior information about the DFN properties C and D (Figure 2.12) is controlled by the parameter γ in (2.12). Large values of γ correspond to higher confidence in the quality and relevance of the data reported in Section 2.5.6. Figure 2.13, 2.14 exhibits posterior PDFs of C and D computed via our Bayesian assimilation procedure with $\gamma = 0.5$ and 1. Visual comparison of Figures 2.10, 2.11, 2.13, and 2.14 reveals that the incorporation of the prior information about generic (not site-specific) correlations between C and D sharpens our estimation of these parameters, i.e., decreases the area in the parameter space where they are predicted to lie with high probability. Putting more trust in the prior, i.e., using a higher value of γ, amplifies this trend. However, the increase in certainty might be misplaced, as witnessed by several

examples the reference parameter values fall outside the high probability regions.

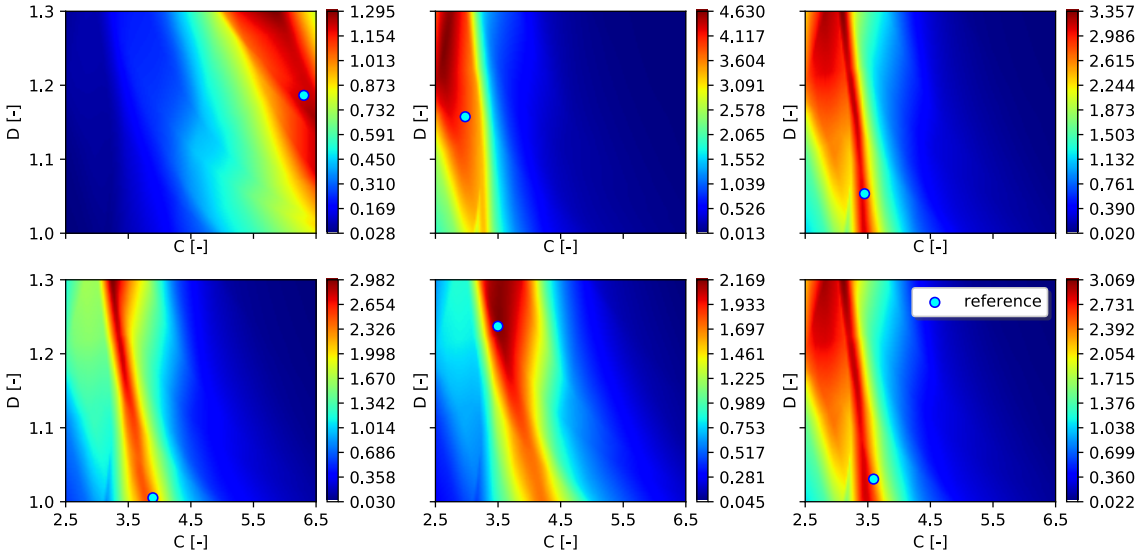

Figure 2.13: Examples of posterior PDFs of the DFN parameters C and D in the presence of prior information, for six experiments defined by the reference parameter values (blue circles). These PDFs are computed via Bayesian assimilation with the informative prior (Figure 2.12), whose relative importance is $\gamma = 0.5$.

Fracture network's connectivity is another potential source of information that can boost one's ability to infer the parameters C and D from CBTEs. Let N_{con_i} denote the number of connected fracture networks among 20 random realizations of a DFN characterized by $(C, D)_i$. Figure 2.15 exhibits N_{con_i} for $N_{\text{sim}} = 10^4$ DFNs characterized by $(C, D)_i$ $(i = 1, \ldots, N_{\text{sim}})$, with the results interpolated to $10^4 \times 10^3$ mesh of the (C, D) space by means of a shallow NN. We define a prior PDF for C and D as:

$$f_{\mathbf{m}}(\tilde{\mathbf{m}}) \propto N^2_{\text{con}}(\tilde{\mathbf{m}}), \quad N_{\text{con}} \in [0, 1, \ldots, 20], \tag{2.15}$$

which is properly normalized to ensure it integrates to one. This prior PDF, shown in Figure 2.15, assigns larger probability to those (C, D) pairs that show higher connectivity in our data set.

The Bayesian inference procedure with this prior yields the posterior joint PDFs of C and D in Figure 2.16. These distributions are sharper than those computed with either uninformative (Figure 2.10) or correlation-based (Figures 2.13 and 2.14) priors, indicating the further increased confidence in the method's predictions of C and D. Adding prior information to our inverse analysis results in general in reducing the extent of the highest probability zones (dark red zones) in comparison with the posterior distributions that are computed without prior information. As before, assigning more weight to the prior, i.e., increasing γ, reduces the area of the high-probability regions in the (C, D) space. This increased confidence in predictions of C and D is more pronounced when

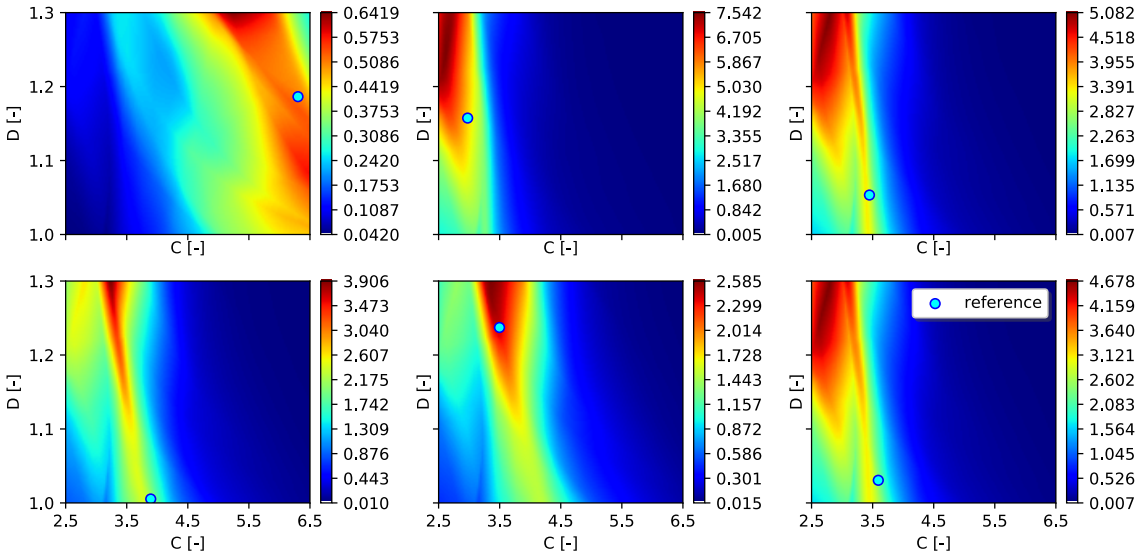

Figure 2.14: Examples of posterior PDFs of the DFN parameters C and D in the presence of prior information, for six experiments defined by the reference parameter values (blue circles). These PDFs are computed via Bayesian assimilation with the informative prior (Figure 2.12), whose relative importance is $\gamma = 1.0$ (bottom).

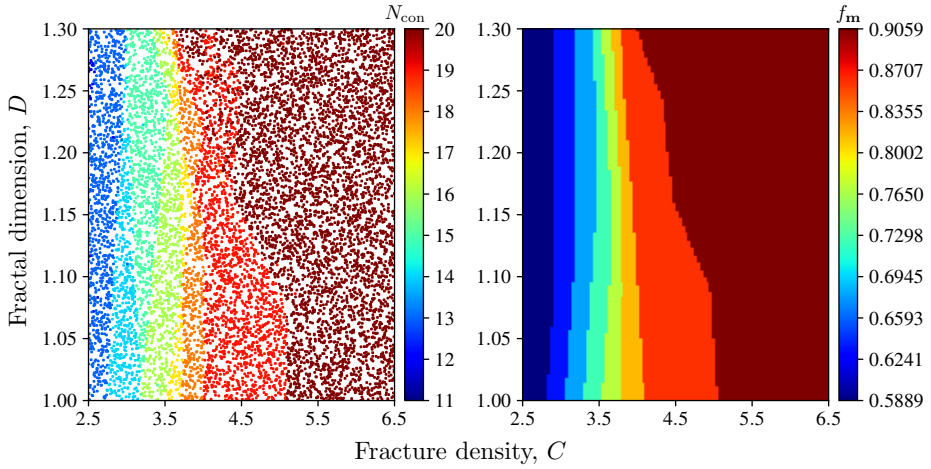

Figure 2.15: Number of connected networks, N_{con}, averaged over 20 random realizations of the DFN model with a given parameter pair $\mathbf{m} = (C, D)^\top$ (left); and corresponding prior PDF $f_{\mathbf{m}}$ in (2.15) (right).

the connectivity-based prior, rather than the correlation-based prior, is used. There is even more important decrease of the high probability zones since the extent of the red zones decreases from Figures 2.13 and 2.16 to Figures 2.14 and 2.17. The connectivity information also ensures that this confidence is not misplaced, i.e., the reference parameter values lie within the high-probability regions. The location of the reference parameter value is better with the prior defined from the connectivity of the system as it is still located in the highest probability zones for most of the cases shown in Figures 2.16 and 2.17.

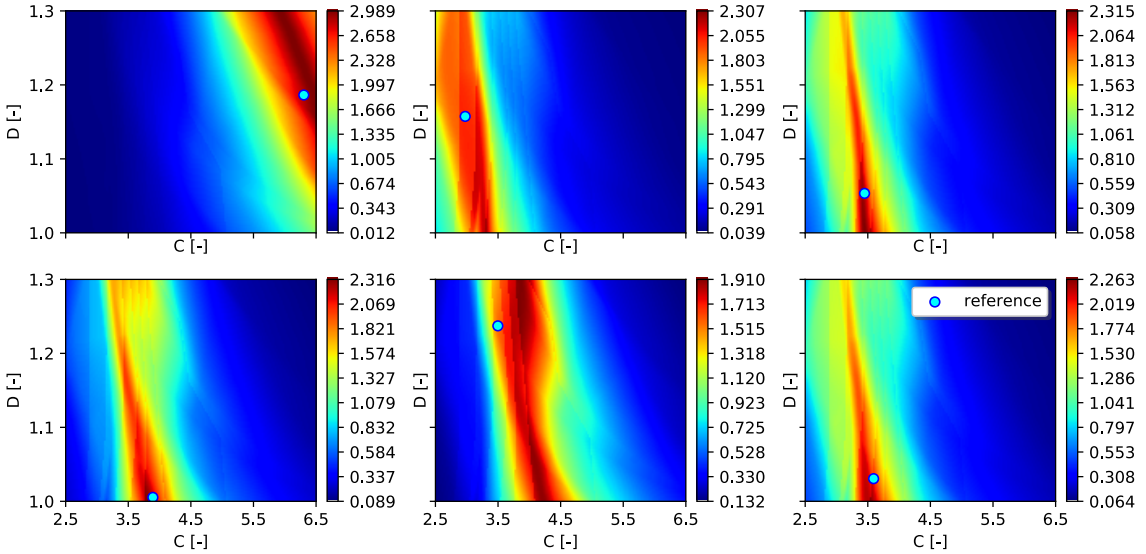

Figure 2.16: Examples of posterior PDFs of the DFN parameters C and D in the presence of prior information, for six experiments defined by the reference parameter values (blue circles). These PDFs are computed via Bayesian assimilation with the informative prior (2.15), whose relative importance increases from $\gamma = 0.5$.

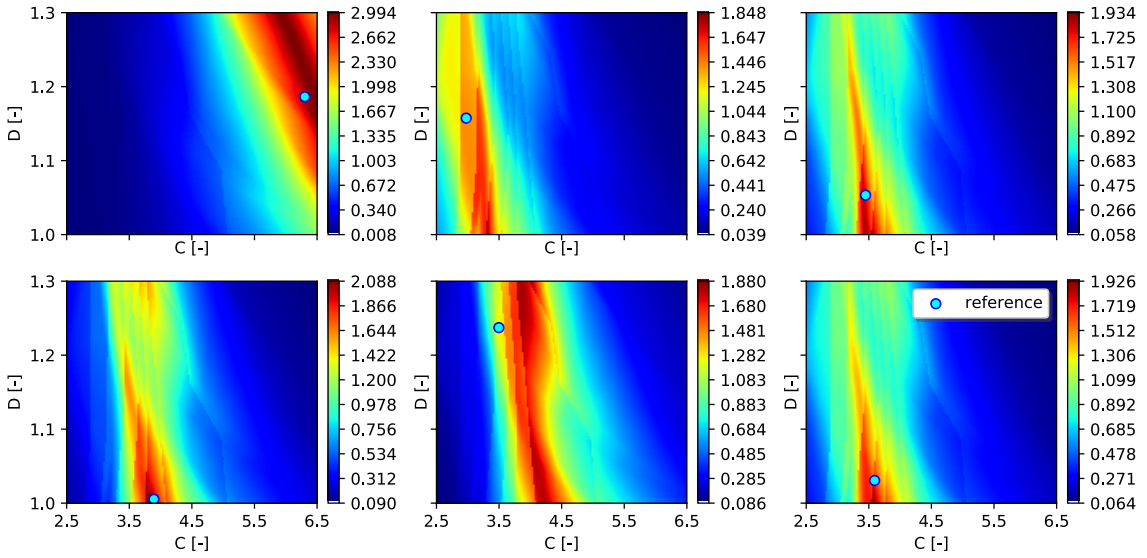

Figure 2.17: Examples of posterior PDFs of the DFN parameters C and D in the presence of prior information, for six experiments defined by the reference parameter values (blue circles). These PDFs are computed via Bayesian assimilation with the informative prior (2.15), whose relative importance increases from $\gamma = 1$.

2.5.6 Field-scale Observations of Fracture Networks

For the sake of completeness, we report in Table 2.5 the field-scale observations of fracture networks from [10]. These are accompanied by our calculation of the corresponding values of parameters C and D in the model of fracture networks. Figure 2.18 shows the paired values of parameters C and D that are used in expression (2.1) and reported in Table 2.5.

2.6 Conclusions

We developed and applied a computationally efficient parameter-estimation method, which makes it possible to infer the statistical properties of a fracture network from cross-borehole thermal experiments (CBTEs). A key component of our method is the construction of a neural network surrogate of the physics-based model of fluid flow and heat transfer in fractured rocks. The negligible computational cost of this surrogate allows for the deployment of a straightforward grid search in the parameter space spanned by fracture density C and fractal dimension D. A neural network is trained with a $\{(C, D)_i, ICDF_i\}_{i=1}^{N_d=10000}$ dataset to replace the forward transport model for computational efficiency. The likelihood of a given pair of (C, D) is computed by comparing the output result of the neural network surrogate model with the ICDF measurement. This likelihood, together with a uniform or nonuniform prior density of (C, D), determines the Bayesian-like posterior density of

N_f [-]	a [-]	S [m^2]	l_{min} [m]	α [-]	D [-]	C [-]
107	1.74	24	0.1	0.60035	0.74	86.80731
121	2.11	25	0.1	0.41703	1.11	45.46014
3499	1.88	$2.70 \cdot 10^{11}$	10^3	$4.97809 \cdot 10^{-6}$	0.88	0.01979
120	0.9	$8.25 \cdot 10^7$	40	$-1.00582 \cdot 10^{-7}$	-0.1	0.00012
101	1	$2.62 \cdot 10^7$	57	0	0	NaN
300	1.76	NP	$7.00 \cdot 10^3$	NaN	0.76	NaN
380	1.9	$3.43 \cdot 10^3$	3	0.26777	0.9	113.05832
350	2.1	$1.26 \cdot 10^8$	220	0.00115	1.1	0.36680
1000	3.2	$1.60 \cdot 10^9$	380	0.65137	2.2	296.07649
1000	2.1	$1.65 \cdot 10^{10}$	$2.00 \cdot 10^3$	0.00028	1.1	0.25921
800	2.2	$2.50 \cdot 10^1$	$6.00 \cdot 10^{-2}$	1.31254	1.2	875.02702
380	2.1	NP	$2.50 \cdot 10^3$	NaN	1.1	NaN
1700	2.02	$1.00 \cdot 10^{10}$	$1.00 \cdot 10^3$	0.0002	1.02	0.33182
260	1.3	$8.75 \cdot 10^3$	1.00	0.00891	0.3	7.72571
100	1.8	$2.10 \cdot 10^3$	1.00	0.03809	0.8	4.76190
873	2.64	$3.40 \cdot 10^1$	$5.00 \cdot 10^{-3}$	0.00709	1.64	3.7745
320	2.61	$2.07 \cdot 10^7$	4.00·10	0.00945	1.61	1.87779
50	1.67	$2.90 \cdot 10^7$	7.00·10	$1.99004 \cdot 10^{-5}$	0.67	0.00148
180	1.97	$2.80 \cdot 10^8$	$3.00 \cdot 10^2$	0.00016	0.97	0.02925
400	2.21	$1.20 \cdot 10^8$	4.00·10	0.00035	1.21	0.11573
250	2.11	$2.50 \cdot 10^{11}$	$4.50 \cdot 10^3$	$1.26005 \cdot 10^{-5}$	1.11	0.00284
400	2.84	$2.90 \cdot 10^{11}$	$5.50 \cdot 10^3$	0.01935	1.84	4.20716
70	2.67	$3.60 \cdot 10^9$	$1.60 \cdot 10^3$	0.00728	1.67	0.30533
150	2.66	$5.10 \cdot 10^9$	$1.25 \cdot 10^3$	0.00675	1.66	0.61021
200	3.07	$6.20 \cdot 10^9$	$1.00 \cdot 10^3$	0.10829	2.07	10.46329
1034	2.51	$8.70 \cdot 10^7$	1.00·10	0.00058	1.51	0.39767
40	1.6	$2.00 \cdot 10^4$	$6.00 \cdot 10^{-2}$	0.00022	0.6	0.01479
318	2.42	$1.69 \cdot 10^8$	7.00·10	0.00111	1.42	0.24946
291	2.69	$1.69 \cdot 10^8$	7.00·10	0.00382	1.69	0.65783
78	2.1	$1.69 \cdot 10^8$	$1.00 \cdot 10^2$	$8.04638 \cdot 10^{-5}$	1.1	0.00570
70	2.67	$3.60 \cdot 10^9$	$1.60 \cdot 10^3$	0.00728	1.67	0.30533
150	2.66	$5.10 \cdot 10^9$	$1.25 \cdot 10^3$	0.00675	1.66	0.61021
200	3.07	$6.20 \cdot 10^9$	$1.00 \cdot 10^3$	0.10829	2.07	10.46329
1034	2.51	$8.70 \cdot 10^7$	1.00·10	0.00058	1.51	0.39767
40	1.6	$2.00 \cdot 10^4$	$6.00 \cdot 10^{-2}$	0.00022	0.6	0.01479
218	2.02	1.00	$2.00 \cdot 10^{-2}$	4.11251	1.02	878.94881

111	3.04	$8.40{\cdot}10^{7}$	$2.00{\cdot}10^{2}$	0.13328	2.04	7.25217
470	1.8	$1.17{\cdot}10^{4}$	$6.00{\cdot}10^{-2}$	0.00338	0.8	1.98852
417	2.18	$6.00{\cdot}10^{7}$	$4.00{\cdot}10$	0.00064	1.18	0.22519
201	2.4	3.00E-01	1.50E-04	0.00416	1.4	0.59676
100	2.4	$6.00{\cdot}10^{8}$	$7.00{\cdot}10^{2}$	0.00224	1.4	0.16032
1034	2.36	$8.70{\cdot}10^{7}$	$1.00{\cdot}10$	0.00037	1.36	0.28153
450	2.18	$2.20{\cdot}10^{8}$	$7.00{\cdot}10$	0.00036	1.18	0.13843
350	2.75	$1.50{\cdot}10^{9}$	$1.80{\cdot}10^{2}$	0.00361	1.75	0.72239
300	2.37	NP	$1.00{\cdot}10^{2}$	NaN	1.37	NaN

Table 2.5: Fracture number (N_{f}), power-law exponent (a), surface area (S), minimum fracture length ($l_{\min}$), and density parameter α for various fracture networks reported in Table 2 in [10]. The corresponding values of fracture density (C) and fractal dimension (D) in the network model (2.1) are determined from the parameter relationships in Section 2.2.1.

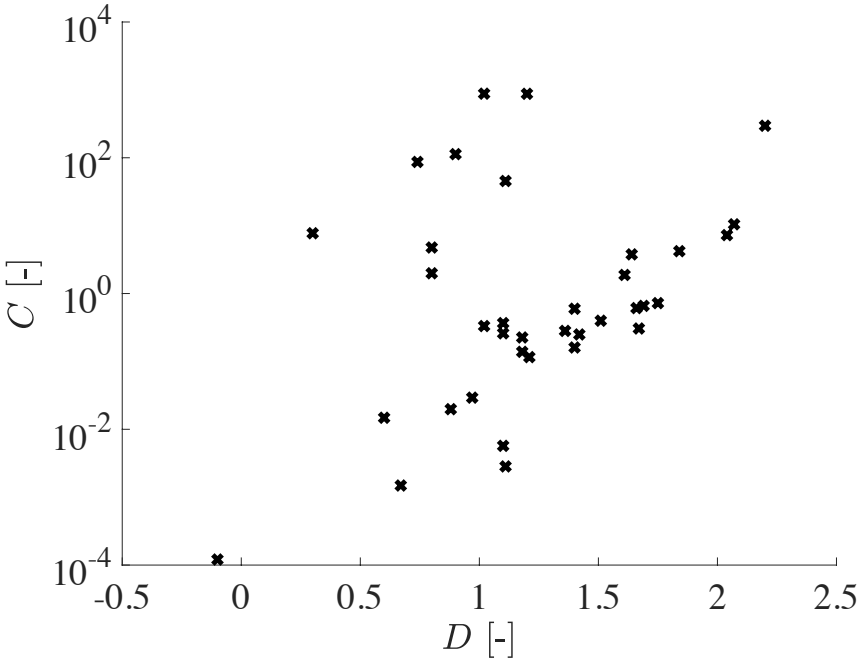

Figure 2.18: Correlation between parameters C and D from the values reported in Table 2.5

(C, D). The posterior distribution on the domain $(C, D) \in [2.5, 6.5] \times [1.0, 1.3]$ is obtained on very fine 10000×1000 mesh grids with the method described above. Our numerical experiments lead to the following major conclusions.

1. The neural network surrogate provides accurate estimates of an average inverse cumulative distribution function (iCDF) of breakthrough times, for the fracture network characterized by given parameters (C, D).

2. In the absence of any expert knowledge about C and D, i.e., when an uninformative prior is used, our method—with the likelihood function defined in terms of the Hellinger distance between the predicted and observed iCDFs—significantly sharpens this prior, correctly identifying parameter regions wherein the true values of (C, D) lie.

3. Incorporation of the prior information about generic (not site-specific) correlations between C and D sharpens our estimation of these parameters, i.e., decreases the area in the parameter space where they are predicted to lie with high probability. Putting more trust in the prior, i.e., using a higher value of γ, amplifies this trend. However, the increase in certainty might be misplaced, as witnessed by several examples the reference parameter values fall outside the high probability regions.

4. Incorporation of the prior information about a fracture network's connectivity yields the posterior joint PDFs of C and D that are sharper than those computed with either uninformative or correlation-based priors, indicating the further increased confidence in the method's predictions of C and D.

5. The increased confidence in predictions of C and D is more pronounced when the connectivity-based prior, rather than the correlation-based prior, is used. The connectivity information also ensures that this confidence is not misplaced, i.e., the reference parameter values lie within the high-probability regions.

Chapter 3

MCMC with CNN Surrogates: Source Identification

In this chapter, we present a DRAM-CNN method to integrate DRAM MCMC sampling with a CNN surrogate forward model. The method is used to reconstruct a contaminant source in a two-dimensional domain. The results are compared with their counterparts obtained via the HMC sampler with the CNN surrogate model.

3.1 Introduction

Identification of contaminant release history in groundwater plays an important role in regulatory efforts and design of remedial actions. Such efforts rely on measurements of solute concentrations collected at a few locations (pumping or observation wells) in an aquifer. Data collection can take place at discrete times and is often plagued by measurement errors. A release history is estimated by matching these data to predictions of a solute transport model, an inverse modeling procedure that is typically ill-posed. Alternative strategies for solving this inverse problem [3, 142, 105, 7] fall into two categories: deterministic and probabilistic. Deterministic methods include least squares regression [128] and hybrid optimization with a genetic algorithm [5, 77]. They provide a "best" estimate of the contaminant release history, without quantifying the uncertainty inevitable in such predictions.

Probabilistic methods, e.g., data assimilation via extended and ensemble Kalman filters [133, 134] and Bayesian inference based on Markov chain Monte Carlo or MCMC [34], overcome this shortcoming. Kalman filters are relatively fast but do not generalize to strongly nonlinear problems, sometimes exhibiting inconsistency between updated parameters and observed states [17]. Particle filters and MCMC are exact even for nonlinear systems but are computationally expensive, and

often prohibitively so. Increased efficiency of MCMC with a Gibbs sampler [92] comes at the cost of generality by requiring the random fields of interest to be Gaussian. MCMC with the delayed rejection adaptive Metropolis (DRAM) sampling [44] is slightly more efficient and does not require the Gaussianity assumption; it has been used in experimental design for source identification [138], and is deployed as part of our algorithm. Gradient-based MCMC methods, such as hybrid Monte Carlo (HMC) sampling [7], increase the slow convergence of these and other MCMC variants. However, the repeated computation of gradients of a Hamiltonian can be prohibitively expensive for high-dimensional transport problems without special treatment.

With an exception of the method of distribution [11, 12], the computational cost of Bayesian methods for data assimilation and statistical inference is dominated by multiple runs of a forward transport model. The computational burden can be significantly reduced by deploying a surrogate model, which provides a low-cost approximation of its expensive physics-based counterpart. Examples of such surrogates include polynomial chaos expansions [138, 19] and Gaussian processes [29, 139]. A possible surrogate-introduced bias can be reduced or eliminated altogether by the use of a two-stage MCMC [139]. Both polynomial chaos expansions and Gaussian processes suffer from the so-called curse of dimensionality, which refers to the degradation of their performance as the number of random inputs becomes large.

Artificial neural networks in general, and deep neural networks in particular, constitute surrogates that remain robust for large numbers of inputs and outputs [96, 94]. Their implementations in open-source software offer an added benefit of being portable to advanced computer architectures, such as graphics processing units and tensor processing units, without significant input from the user. Our algorithm employs a convolutional neural network (CNN) as a surrogate, the role that is related to but distinct from other uses of neural networks in scientific computing, e.g., their use as a numerical method for solving differential equations [74, 70].

In Section 3.2 we formulate the problem of contaminant source identification from sparse and noisy measurements of solute concentrations. Section 3.3 contains a description of our algorithm, which combines MCMC with DRAM sampling (Section 3.3.1) and a CNN-based surrogate of the forward transport model (Section 3.3.2). We also show a gradient-based HMC sampling method with the easily accessible differentiation of the CNN surrogate model. Results of our numerical experiments are reported in Section 3.4; they demonstrate that our DRAM-CNN method is about 20 times faster than DRAM with a physics-based transport model. HMC-CNN is not really faster than DRAM with physics-based transport model in terms of the time obtaining each sample, but the sample chain takes much fewer steps to converge, or "mix well", hence using fewer samples can approximate the posterior distribution better than DRAM samplers. Main conclusions drawn from this study are summarized in Section 3.5.

3.2 Problem Formulation

Vertically averaged hydraulic head distribution $h(\mathbf{x})$ in an aquifer Ω with hydraulic conductivity $K(\mathbf{x})$ and porosity $\theta(\mathbf{x})$ is described by a two-dimensional steady-state groundwater flow equation,

$$\nabla \cdot (K \nabla h) = 0, \qquad \mathbf{x} \in \Omega, \tag{3.1}$$

subject to appropriate boundary conditions on the simulation domain boundary $\partial\Omega$. Once (3.1) is solved, average macroscopic flow velocity $\mathbf{u}(\mathbf{x}) = (u_1, u_2)^\top$ is evaluated as:

$$\mathbf{u} = -\frac{K}{\theta} \nabla h. \tag{3.2}$$

Starting at some unknown time t_0 a contaminant with volumetric concentration c_s enters the aquifer through point-wise or spatially distributed sources $\Omega_\mathrm{s} \subset \Omega$. The contaminant continues to be released for unknown duration T with unknown intensity $q_\mathrm{s}(\mathbf{x}, t)$ (volumetric flow rate per unit source volume), such that $q_\mathrm{s}(\mathbf{x}, t) \neq 0$ for $t_0 \leq t \leq t_0 + T$. The contaminant, whose volumetric concentration is denoted by $c(\mathbf{x}, t)$, migrates through the aquifer and undergoes (bio)geochemical transformations with a rate law $R(c)$. Without loss of generality, we assume that the spatiotemporal evolution of $c(\mathbf{x}, t)$ is adequately described by an advection-dispersion-reaction equation,

$$\frac{\partial \theta c}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c) - \nabla \cdot (\theta \mathbf{u} c) - R(c) + q_\mathrm{s} c_\mathrm{s}, \qquad \mathbf{x} = (x_1, x_2)^\top \in \Omega, \quad t > t_0, \tag{3.3}$$

although other, e.g., non-Fickian, transport models [99, 120, 116] can be considered instead. If the coordinate system is aligned with the mean flow direction, such that $\mathbf{u} = (u \equiv |\mathbf{u}|, 0)^\top$, then the dispersion coefficient tensor $\mathbf{D}$ in (3.3) has components

$$D_{11} = \theta D_\mathrm{m} + \alpha_L u, \qquad D_{22} = \theta D_\mathrm{m} + \alpha_T u, \qquad D_{12} = D_{21} = \theta D_\mathrm{m}, \tag{3.4}$$

where D_m is the contaminant's molecular diffusion coefficient in water; and α_L and α_T are the longitudinal and transverse dispersivities, respectively.

Our goal is to estimate the location and strength of the source of contamination, $r(\mathbf{x}, t) = q_\mathrm{s}(\mathbf{x}, t) c_\mathrm{s}(\mathbf{x}, t)$, by using the transport model (3.1)–(3.4) and concentration measurements $\bar{c}_{m,i} = \bar{c}(\mathbf{x}_m, t_i)$ collected at locations $\{\mathbf{x}_m\}_{m=1}^M$ at times $\{t_i\}_{i=1}^I$. The concentration data are corrupted by random measurement errors, such that:

$$\bar{c}_{m,i} = c(\mathbf{x}_m, t_i) + \epsilon_{mi}, \qquad m = 1, \cdots, M, \quad i = 1, \cdots, I; \tag{3.5}$$

where $c(\mathbf{x}_m, t_i)$ are the model predictions, and the errors ϵ_{mi} are zero-mean Gaussian random variables with covariance $\mathbb{E}[\epsilon_{mi}\epsilon_{nj}] = \delta_{ij} R_{mn}$. Here, $\mathbb{E}[\cdot]$ denotes the ensemble mean; δ_{ij} is the Kronecker

delta function; and R_{mn}, with $m, n \in [1, M]$, are components of the $M \times M$ spatial covariance matrix $\mathbf{R}$ of measurements errors, taken to be the identity matrix multiplied by the standard deviation of the measurement errors. This model assumes both the model (3.1)–(3.4) to be error-free and the measurements errors to be uncorrelated in time but not in space.

3.3 Methods

Our first algorithm comprises MCMC with DRAM sampling, a CNN-based surrogate of the transport model (3.1)–(3.4), and the second algorithm combines HMC-NUTS sampling with CNN-based surrogate model. These three components are described below.

3.3.1 MCMC with DRAM Sampling

Upon a spatiotemporal discretization of the simulation domain, we arrange the uncertain (random) input parameters in (3.1)–(3.4) into a vector $\mathbf{m}$ of length N_m; these inputs may include the spatiotemporally discretized source term $r(\mathbf{x}, t)$, initial concentration $c_{\text{in}}(\mathbf{x})$, hydraulic conductivity $K(\mathbf{x})$, etc. Likewise, we arrange the random measurements $\bar{c}_{m,i}$ into a vector $\mathbf{d}$ of length N_d, and the random measurement noise ϵ_{mi} into a vector $\boldsymbol{\varepsilon}$ of the same length. Then, the error model (3.5) takes the vector form:

$$\mathbf{d} = \mathbf{g}(\mathbf{m}) + \boldsymbol{\varepsilon}, \tag{3.6}$$

where $\mathbf{g}(\cdot)$ is the vector, of length N_d, of the correspondingly arranged stochastic model predictions $c(\mathbf{x}_m, t_i)$ predicated on the model inputs $\mathbf{m}$.

In Bayesian inference, the parameters $\mathbf{m}$ are estimated probabilistically from both model predictions and (noisy) measurements by means of the Bayes theorem,

$$f_{\mathbf{m}|\mathbf{d}}(\tilde{\mathbf{m}}; \tilde{\mathbf{d}}) = \frac{f_{\mathbf{m}}(\tilde{\mathbf{m}}) f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}}; \tilde{\mathbf{d}})}{f_{\mathbf{d}}(\tilde{\mathbf{d}})}, \qquad f_{\mathbf{d}}(\tilde{\mathbf{d}}) = \int f_{\mathbf{m}}(\tilde{\mathbf{m}}) f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}}; \tilde{\mathbf{d}}) \mathrm{d}\tilde{\mathbf{m}}. \tag{3.7}$$

Here, $\tilde{\mathbf{d}}$ is the deterministic coordinate in the phase space of the random variable $\mathbf{d}$; $f_{\mathbf{m}}$ is a prior probability density function (PDF) of the inputs $\mathbf{m}$, which encapsulates the information about the model parameters and contaminant source before any measurements are assimilated; $f_{\mathbf{m}|\mathbf{d}}$ is the posterior PDF of $\mathbf{m}$ that represents refined knowledge about $\mathbf{m}$ gained from the data $\mathbf{d}$; $f_{\mathbf{d}|\mathbf{m}}$ is the likelihood function, i.e., the joint PDF of concentration measurements conditioned on the corresponding model predictions that is treated as a function of $\mathbf{m}$ rather than $\mathbf{d}$; and $f_{\mathbf{d}}$, called "evidence", serves as a normalizing constant that ensures that $f_{\mathbf{m}|\mathbf{d}}(\mathbf{m}; \cdot)$ integrates to 1. Since $\boldsymbol{\varepsilon}$ in (3.5) or (3.6) is multivariate Gaussian, the likelihood function has the form:

$$f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}}; \tilde{\mathbf{d}}) = \frac{1}{(2\pi)^{d/2} |\mathbf{R}|^{1/2}} \exp\left(-\frac{1}{2} \mathbf{v}^\top \mathbf{R}^{-1} \mathbf{v}\right), \qquad \mathbf{v} = \tilde{\mathbf{d}} - \mathbf{g}(\mathbf{m}). \tag{3.8}$$

In high-dimensional nonlinear problems (i.e., problems with large N_m), such as (3.1)–(3.4), the posterior PDF $f_{\mathbf{d}|\mathbf{m}}$ cannot be obtained analytically and computation of the integral in the evidence $f_{\mathbf{d}}$ is prohibitively expensive. Instead, one can use MCMC to draw samples from $f_{\mathbf{m}}(\tilde{\mathbf{m}})f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}})$, without computing the normalizing constant $f_{\mathbf{d}}$. A commonly used MCMC variant relies on the Metropolis–Hastings sampling [34]; this approach uses a zero-mean Gaussian PDF with tunable variance σ^2 to generate proposals near a previous sample, which are accepted with the acceptance rate given by the relative posterior value. The performance of the Metropolis-Hastings sampling depends on the choice of hyperparameters, such as σ^2, and on how well the proposal PDF matches the target PDF. The choice of an inappropriate proposal PDF might cause an extremely slow convergence.

We deploy the DRAM sampling—specifically its numerical implementation in [93]—to accelerate the convergence of MCMC. DRAM sampling differs from the Metropolis–Hasting sampling in two aspects. First, the *delayed rejection* [42] refers to the strategy in which a proposal's rejection in the first attempt is tied to the subsequent proposal that can be accepted with a combined probability for the two proposals; this rejection delay is iterated multiple times in the sampling process. Second, *adaptive Metropolis* [43] uses past sample chains to tune the proposal distribution in order to accelerate the convergence of MCMC. The DRAM sampling is more efficient than other sampling strategies for many problems, including that of source identification [138].

3.3.2 Deep Convolutional Neural Networks

Any MCMC implementation requires many solves of the transport model (3.1)–(3.4) for different realizations of the input parameters $\mathbf{m}$. We use a CNN surrogate model to alleviate the cost of each solve. Several alternative input-output frameworks to construct a surrogate model are shown in Table 3.1. Among these, autoregressive models predict a concentration map only for the next time step. When measurements are collected at multiple times, an autoregressive model has to be repeatedly evaluated, for each realization of the inputs $\mathbf{m}$. If the release time, conductivity field, and porosity are known, then $\mathbf{m}$ represents the initial concentration field $c_{\text{in}}(\mathbf{x})$. Otherwise, $\mathbf{m}$ is the stack of the maps of $c_{\text{in}}(\mathbf{x})$, conductivity field $K(\mathbf{x})$, porosity field $\theta(\mathbf{x})$, etc.

Model	Input	Output	Modeling frequency
PDE model	$\mathbf{m}$	$\{c(\mathbf{x}, t_i)\}$	1
Image-to-image	$\mathbf{m}$	$\{c(\mathbf{x}, t_i)\}$	1
Image-to-sensors	$\mathbf{m}$	$\{c(\mathbf{x}_m, t_i)\}$	1
Autoregressive image-to-image	$c(\mathbf{x}, t)$	$c(\mathbf{x}, t + \Delta t)$	I

Table 3.1: Alternative input-output frameworks for construction of surrogate models. The data are collected at M locations $\mathbf{x}_m$ $(m = 1, \cdots, M)$ at I times t_i $(i = 1, \cdots, I)$.

We choose an image-to-image regression model, rather than the autoregressive surrogate used in [94] to solve a similar source identification problem, for the following reasons. First, it is better at

generalization than image-to-sensors models. Second, although autoregressive surrogates excel at regression tasks [94], they might become computationally expensive when the measurement frequency is high.

Our image-to-image regression model replaces the PDE-based transport model (3.1)–(3.4) or $\mathbf{g}(\mathbf{m})$ with a CNN $\mathbf{N}(\mathbf{m})$ depicted in Figure 3.1, i.e.,

$$\mathbf{g} : \mathbf{m} \xrightarrow{\text{PDEs}} \{c(x_m, t_i)\}_{m,i=1}^{M,I} \text{ is replaced with } \mathbf{N} : \mathbf{m} \xrightarrow{\text{CNN}} \{c(\mathbf{x}, t_i)\}_{i=1}^{I}. \tag{3.9}$$

We start by attempting to demystify neural networks, which are spreading virally throughout the hydrologic community. A simplest way to relate the model output $\mathbf{d}$ to the model input $\mathbf{m}$ without having to run the model $\mathbf{g}$ is to replace the latter with a linear input-output relation $\hat{\mathbf{d}} = \mathbf{W}\mathbf{m}$, where $\mathbf{W}$ is an $N_d \times N_m$ matrix of weights whose numerical values are obtained by minimizing the discrepancy between the $\hat{\mathbf{d}}$ and $\mathbf{d}$ values that are either measured or computed with the model $\mathbf{g}$ or both. The performance of this linear regression, in which the bias parameters are omitted to simplify the presentation, is likely to be suboptimal, because a relationship between the inputs and outputs is likely to be highly nonlinear. Thus, one replaces $\hat{\mathbf{d}} = \mathbf{W}\mathbf{m}$ with a nonlinear model $\hat{\mathbf{d}} = \sigma(\mathbf{W}\mathbf{m})$, in which a prescribed function $\sigma(\cdot)$ operates on each element of the vector $\mathbf{W}\mathbf{m}$. Examples of this so-called activation function include a sigmoidal function (e.g., tanh) and a rectified linear unit (ReLU). The latter is defined as $\sigma(s) = \max(0, s)$, it is used here because of its current popularity in the field. The nonlinear regression model $\hat{\mathbf{d}} = \sigma(\mathbf{W}\mathbf{m}) \equiv (\sigma \circ \mathbf{W})(\mathbf{m})$ constitutes a single "layer" in a network.

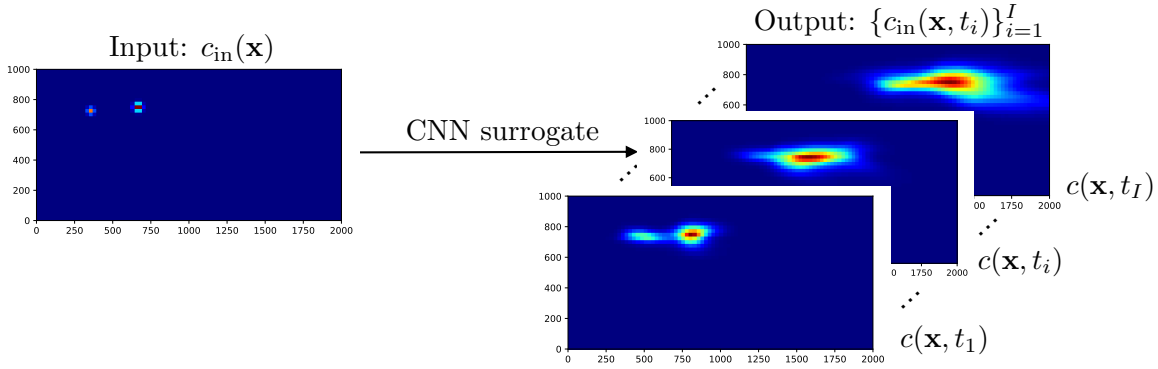

Figure 3.1: A surrogate model constructed with a convolution neural network (CNN). The surrogate takes as input a set of uncertain parameters $\mathbf{m}$, e.g., an initial contaminant concentration field $c_{\text{in}}(\mathbf{x})$ and returns as output temporal snapshots of the solute concentrations $c(\mathbf{x}, t_i)$ in an aquifer.

A (deep) fully connected neural network $\mathbf{N_f}$ comprising N_l "layers" is constructed by a repeated application of the activation function to the input,

$$\mathbf{d} = \mathbf{N_f}(\mathbf{m}; \boldsymbol{\Theta}) \equiv (\sigma_{N_l} \circ \mathbf{W}_{N_l - 1}) \circ \ldots \circ (\sigma_2 \circ \mathbf{W}_1)(\mathbf{m}). \tag{3.10a}$$

In general, different activation functions might be used in one network. The parameter set $\mathbf{\Theta} = \{\mathbf{W}_1, \ldots, \mathbf{W}_{N_l-1}\}$ consists of the weights $\mathbf{W}_n$ connecting the nth and $(n+1)$st layers. In this recursive relation,

$$\begin{cases} \mathbf{s}_1 = (\sigma_2 \circ \mathbf{W}_1)(\mathbf{m}) \equiv \sigma_2(\mathbf{W}_1\mathbf{m}), \\ \mathbf{s}_2 = (\sigma_3 \circ \mathbf{W}_2)(\mathbf{s}_1) \equiv \sigma_3(\mathbf{W}_2\mathbf{s}_1), \\ \vdots \\ \mathbf{d} = (\sigma_{N_l} \circ \mathbf{W}_{N_l-1})(\mathbf{s}_{N_l-2}) \equiv \sigma_{N_l}(\mathbf{W}_{N_l-1}\mathbf{s}_{N_l-2}), \end{cases} \tag{3.10b}$$

the weights $\mathbf{W}_1$ form a $d_1 \times N_m$ matrix, $\mathbf{W}_2$ is a $d_2 \times d_1$ matrix, $\mathbf{W}_3$ is a $d_3 \times d_2$ matrix,$\cdots$, and $\mathbf{W}_{N_l-1}$ is a $N_d \times d_{N_l-2}$ matrix. The integers $d_1, \cdots, d_{N_l-2}$ represent the number of neurons in the corresponding inner layers of the network. The fitting parameters $\mathbf{\Theta}$ are obtained, or the "network is trained", by minimizing the discrepancy between the prediction and the output in the dataset.

The size of the parameter set $\mathbf{\Theta}$ grows rapidly with the number of layers N_l and the number of neurons d_n in each inner layer. When the output layer contains hundreds or thousands of variables (aka "features", such as concentrations at observation wells collected at multiple times), this size can be unreasonably large. By utilizing a convolution-like operator to preserve the spatial correlations in the input(shown in Figure 3.2 and Equation 3.11), CNNs reduce the size of $\mathbf{\Theta}$ and scale much better with the number of parameters than their fully connected counterparts. Given a two-dimensional input $\mathbf{x} \in \mathbb{R}^{H \times W}$, a convolutional layer evolves from $\mathbf{x}$ to the feature value ζ at location (u, v) with the following relation:

$$\zeta^q_{u,v}(x_{u,v}) = \sigma\left(\sum_{i=1}^{k'_1}\sum_{j=1}^{k'_2} \omega^q_{i,j} x_{u+i,v+j}\right), \tag{3.11}$$

here, $\omega^q \in \mathbb{R}^{k'_1 \times k'_2}$ are a series of filters, $q = 1, ..., N_q$. The output feature maps of a convolutional layer are obtained by sliding filters over the whole input image. σ is the activation function mentioned earlier. With this operation, the resulting output of a convolutional layer consists of N_q feature maps. Each feature map has size $H_y \times W_y$ determined by the input $\mathbf{x}$ size $H_x \times W_x$ [28]:

$$\begin{aligned} H_y &= \left[\frac{H_x + 2p_1 - k'_1}{s_1} + 1\right], \\ W_y &= \left[\frac{W_x + 2p_2 - k'_2}{s_2} + 1\right], [\cdot] : \text{floor function.} \end{aligned} \tag{3.12}$$

p_i and s_i are padding of '0' and stride of the kernels respectively in a convolutional operation.

CNNs are widely used to perform image-to-image regression. We refer the interested reader to [39] for an in-depth description of CNNs. In this study, a CNN is trained to predict the concentration map at times when the measurements were obtained.

Specifically, we use a convolutional encoder-decoder network to perform the regression with a

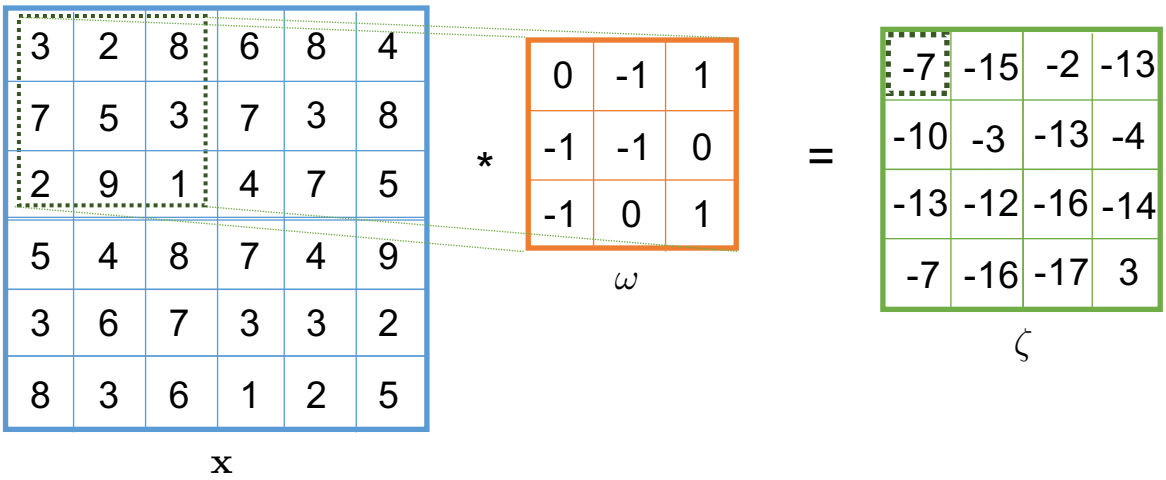

Figure 3.2: A convolutional operation in convolutional neural networks. ω is a filters in a convolutional layer, ζ and $\mathbf{x}$ denotes the input and output of the convolutional operation with this filter ω. The size of the output corresponds to Equation 3.12

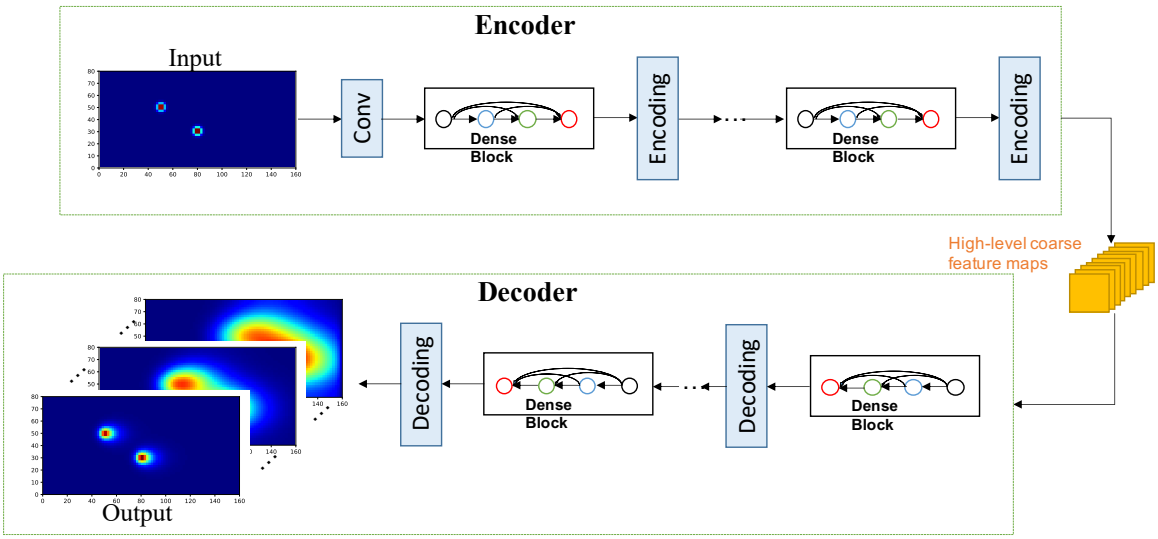

Figure 3.3: Contaminant transport surrogate modeling: input-output illustration and the convolutional encoder-decoder network architecture.

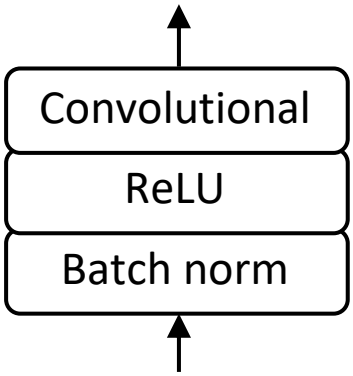

Figure 3.4: "Conv" layer in Figure 3.3. In each "Conv" layer, there exist three consecutive operations, batch normalization, ReLU activation, and a convolutional layer.

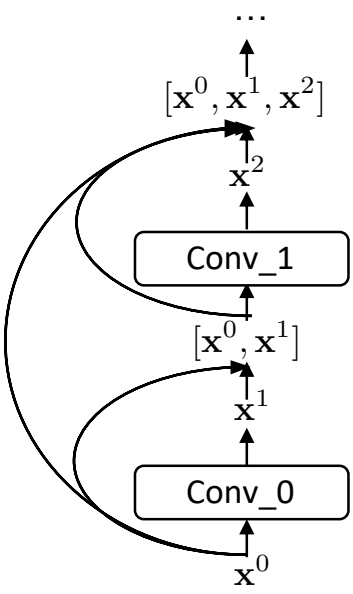

Figure 3.5: A dense block in DenseED neural network in Figure 3.3. A dense block consists of several consecutive small "Conv" layers, with the input of each layer being the concatenation of all previous inputs.

coarse-refine process. In the latter, the encoder extracts the high-level coarse features of the input maps, and the decoder refines the coarse features to the full maps again [94, Fig. 2]. No fully connected layers are used in the architecture to avoid parameter abuse, instead, dense blocks are used to create more connection between the layers to let information propagate, and further more reduce parameters by reusing previous ones. This is done with consecutive convolutional layers with connections between the preceding layers and the lth layer as shown in Figure 3.5. Three operations consist each "Conv" layer referred to in Figure 3.3: batch normalization, rectified linear unit (ReLU activation), and a convolutional layer as shown in Figure 3.4. With all these elements, the whole architecture is shown in Figure 3.3. The L_1-norm loss function, L_2-norm weight regularization, and stochastic gradient descent [15] are used in the parameter estimation process.

It is worthwhile emphasizing that unlike some surrogate models, e.g., polynomial chaos which can predict a solution at any time, the CNN used in this study predicts only concentration maps for a short period. The reason is that for the inverse problem under consideration, only observations at measurement times are of interest and a model's ability to predict concentrations at later times is immaterial.

3.3.3 HMC with NUTS Sampling with CNN surrogate

Following the Bayesian theorem formulated above, we also test Hybrid Monte Carlo sampling. Hybrid Monte Carlo (HMC) sampling, brought up in [26], is a MCMC sampling method uniting MH sampling and Hamiltonian dynamics. This method can be referred to as either Hybrid Monte Carlo or Hamiltonian Monte Carlo.

In Hamiltonian dynamics, the state of particles can be described by a d-dimensional position vector, $\mathbf{q}$, and a d-dimensional momentum vector, $\mathbf{p}$. A function of $\mathbf{q}$ and $\mathbf{p}$ known as the Hamiltonian, $H(\mathbf{q}, \mathbf{p})$ denotes the total energy of the particle with state$(\mathbf{q}, \mathbf{p})$[98].

The partial derivatives of the Hamiltonian determine how the position vector $\mathbf{q}$ and momentum vector $\mathbf{p}$ of particles evolve over time τ. The evolution of the position and momentum of particles can be described with the following Hamilton's equation:

$$\frac{dq_i}{d\tau} = \frac{\partial H}{\partial p_i}, \tag{3.13}$$

$$\frac{dp_i}{d\tau} = -\frac{\partial H}{\partial q_i}. \tag{3.14}$$

Here we use the letter τ instead of t to distinguish from the time in the forward flow and contaminant transport model. τ is the fictitious time in Hamiltonian dynamics.

Conventionally, the Hamiltonian function of $\mathbf{p}$ and $\mathbf{q}$ is written as the sum of potential energy and kinetic energy of the particles at state $(\mathbf{q}, \mathbf{p})$.

$$H(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p}), \tag{3.15}$$

in which $U(\mathbf{q})$ is the potential energy, determined by the position vector $\mathbf{q}$, $K(\mathbf{p})$ is the kinetic energy, determined by the momentum vector $\mathbf{p}$. The kinetic energy is usually defined as:

$$K(\mathbf{p}) = \mathbf{p}^T M^{-1}\mathbf{p}/2 \tag{3.16}$$

Matrix $\mathbf{M}$ is a symmetric, positive-definite matrix representing the mass of the particles. With the kinetic energy defined as such, $K(p)$ will be equivalent to the negative log probability density of a normal distribution with covariance matrix $\mathbf{M}$. This covariance matrix is usually diagonal in many studies [98].

In Hamiltonian dynamics, the time derivative of total energy(Hamiltonian) $H(q,p)$ is denoted as:

$$\frac{\partial H(q,p)}{\partial \tau} = \frac{\partial H(\mathbf{q},\mathbf{p})}{\partial \mathbf{q}}\frac{d\mathbf{q}}{d\tau} + \frac{\partial H(\mathbf{q},\mathbf{p})}{\partial \mathbf{p}}\frac{d\mathbf{p}}{d\tau} \tag{3.17}$$

$$= \frac{dU(\mathbf{q})}{d\mathbf{q}}\frac{d\mathbf{q}}{d\tau} + \frac{dK(\mathbf{p})}{d\mathbf{p}}\frac{d\mathbf{p}}{d\tau}. \tag{3.18}$$

With the kinetic energy $K(\mathbf{p}) = \mathbf{p}^T M^{-1}\mathbf{p}/2$,

$$\frac{dK(\mathbf{p})}{d\mathbf{p}} = \mathbf{M}^{-1}\mathbf{p}, \tag{3.19}$$

$$\frac{d\mathbf{q}}{d\tau} = M^{-1}\mathbf{p}. \tag{3.20}$$

The total Hamiltonian, i.e., the total energy of a particle without perturbation from outside, keeps constant,

$$\frac{\partial H(\mathbf{q},\mathbf{p})}{\partial \tau} = 0, \tag{3.21}$$

$$\frac{dU(\mathbf{q})}{d\mathbf{q}}\frac{d\mathbf{q}}{d\tau} + \frac{dK(\mathbf{p})}{d\mathbf{p}}\frac{d\mathbf{p}}{d\tau} = 0, \tag{3.22}$$

$$\frac{dU(\mathbf{q})}{d\mathbf{q}}\mathbf{M}^{-1}\mathbf{p} + \mathbf{M}^{-1}\mathbf{p}\frac{d\mathbf{p}}{d\tau} = 0, \tag{3.23}$$

leading to the constraint:

$$\frac{dU(\mathbf{q})}{d\mathbf{q}} = -\frac{d\mathbf{p}}{d\tau}. \tag{3.24}$$

In Hamiltonian dynamics, the mapping T_s from the state at time $\tau, (\mathbf{q}(\tau), \mathbf{p}(\tau))$ to the state at $\tau + s, (\mathbf{q}(\tau + s), \mathbf{p}(\tau + s))$ is one-to-one. The inverse mapping exists and is denoted as T_{-s}. This reversibility property ensures MCMC with Hamiltonian dynamics to keep the desired posterior distribution invariant over time.

In computer programs, the evolution of the state vector of particles is implemented by finite difference numerical method. For reversibility in fictitious time τ, leapfrog method is usually a good

choice of the numerical method. With $m_i = \mathbf{M}_{i,i}$ and $\mathbf{M}$ being diagonal,

$$p_i(\tau + \frac{\Delta\tau}{2}) = p_i(\tau) - \frac{\Delta\tau}{2}\frac{\partial U}{\partial q_i}(q(\tau)), \tag{3.25}$$

$$q_i(\tau + \Delta\tau) = q_i(\tau) + \Delta\tau\frac{p_i(\tau + \frac{\Delta\tau}{2})}{m_i} \tag{3.26}$$

$$p_i(\tau + \Delta\tau) = p_i(\tau + \frac{\Delta\tau}{2}) - \frac{\Delta\tau}{2}\frac{\partial U}{\partial q_i}(q(\tau + \Delta\tau)). \tag{3.27}$$

The symmetry of leapfrog method enables the evolution of the states to be reversed, given that the kinetic energy $K(\mathbf{p})$ is the same for particles with state $\mathbf{p}$ and $-\mathbf{p}$, tracing back through $\Delta\tau$ will leads the solution back to its previous state.

The total Hamiltonian $H(\mathbf{q}, \mathbf{p})$ being constant is related to the concept of a canonical distribution. Given an energy function $E(\mathbf{m})$ for a state variable $\mathbf{m}$ in some physical system, the canonical distribution of states has the PDF

$$P(\mathbf{m}) = \frac{1}{Z}\exp(-E(\mathbf{m})/T),$$

In which T is the temperature of the system, Z is the normalizing constant. For a PDF $P(\mathbf{m})$, build a system with the energy of state $\mathbf{m}$ to be $E(\mathbf{m})$, samples generated from this system will then be equivalent to samples drawn from the its canonical distribution $P(\mathbf{m})$.

In a Hamiltonian system, the total energy of the particle is the Hamiltonian $H(\mathbf{q}, \mathbf{p})$, the canonical distribution is of the following form:

$$P(\mathbf{q}, \mathbf{p}) = \frac{1}{Z}\exp(-H(\mathbf{q}, \mathbf{p})/T), \tag{3.28}$$

Because the total Hamiltonian is the sum of two forms of energy,

$$H(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p}),$$

the joint PDF of state $(\mathbf{q}, \mathbf{p})$ is

$$P(\mathbf{q}, \mathbf{p}) = \frac{1}{Z}\exp(-U(\mathbf{q})/T)\exp(-K(\mathbf{p})/T). \tag{3.29}$$

Now we combine Hamiltonian dynamics with MCMC sampling. Usually, state $\mathbf{q}$ represents the state of variable of interest, the potential energy $U(\mathbf{q})$ is not specified and can be customized for different problems. The exponential form of the canonical distribution infers the possibility that the potential energy $U(\mathbf{q})$ is of the form:

$$U(\mathbf{q}) = -\ln[\pi(\mathbf{q})L(\mathbf{q}|\mathbf{d})], \tag{3.30}$$

in which $\pi(\mathbf{q})$ is the prior density of $\mathbf{q}$ and $L(\mathbf{q}|\mathbf{d}) = P(\mathbf{d}|\mathbf{q})$ is the likelihood function given data $\mathbf{d}$.

With the canonical distribution defined such, MH sampling can be applied with Hamiltonian dynamics to build the hybrid or Hamiltonian Monte Carlo sampling. For a Bayesian updating problem, suppose that the desired posterior distribution we want to sample from is $P(\mathbf{m}|\mathbf{d}) \propto P(\mathbf{m})P(\mathbf{d}|\mathbf{m})$, if further assumptions are made for the likelihood function $P(\mathbf{d}|\mathbf{m})$ and the prior density distribution $P(\mathbf{m})$ to be Gaussian, then the likelihood function and the prior in a logarithm form is:

$$P(\mathbf{m}|\mathbf{d}) \propto P(\mathbf{m})P(\mathbf{d}|\mathbf{m}) \tag{3.31}$$

$$\propto \exp(-\frac{\mathbf{m}^T R^{-1}\mathbf{m}}{2}) \exp(-H_{\text{obs}}(\mathbf{m})) \tag{3.32}$$

$$\propto \exp(-\frac{\mathbf{m}^T R^{-1}\mathbf{m}}{2} - H_{\text{obs}}(\mathbf{m})), \tag{3.33}$$

where R is the covariance matrix of the prior of x. H_{obs} corresponds to the log-likelihood function denoted in another form in Equation 3.7.

There are two main stages in HMC method. At the first step, for a given sample $\mathbf{m}$, we treat $\mathbf{m}$ as the position vector $\mathbf{q}$ in Hamiltonian dynamics, $m_i = q_i$ for $i = 1, \ldots, d$. Then, we independently draw a momentum vector from a Gaussian distribution,

$$P(p_i) \propto \exp\left(\frac{p_i^2}{2m_i}\right), \qquad i = 1, \ldots, d,$$

using the time derivatives,

$$\frac{d\mathbf{q}}{d\tau} = M^{-1}\mathbf{p}, \ \frac{d\mathbf{p}}{d\tau} = -\frac{dU(\mathbf{q})}{d\mathbf{q}}, \tag{3.34}$$

where $U(\mathbf{q})$ is obtained from Equation (3.33) as:

$$U(\mathbf{q}) = \frac{\mathbf{q}^T R^{-1}\mathbf{q}}{2} + H_{\text{obs}}(\mathbf{q}). \tag{3.35}$$

With the time derivatives of the position vector and momentum vector, state vector $(\mathbf{q}, \mathbf{p})$ now can be evolved in fictitious time τ with leapfrog numerical method. After "enough" time τ, the Hamiltonian dynamics is well mixed and due to the invariant of $H(\mathbf{q}, \mathbf{p})$, state $(\mathbf{q}, \mathbf{p})$ should follow the canonical distribution in Equation (3.29). At the end of the evolution the states becomes $(\mathbf{q}^*, \mathbf{p}^*)$.

At the second step, we apply a MH updating to the end states of the Hamiltonian dynamics. From last step, a new joint state $(\mathbf{q}^*, \mathbf{p}^*)$ was proposed by the Hamiltonian dynamics. Now, whether the proposed $\mathbf{q}^*$ is accepted as the next sample depends on how preferable it is in the canonical

distribution, the acceptance rate is:

$$r_{\text{accp}} \equiv \min\{1, \exp(-H(\mathbf{q}^*, \mathbf{p}^*) + H(\mathbf{q}, \mathbf{p}))\} \tag{3.36}$$

$$= \min\{1, \exp(-U(\mathbf{q}^*) + U(\mathbf{q}) - K(\mathbf{p}^*) + K(\mathbf{p}))\}. \tag{3.37}$$

The process is to accept the new state $(\mathbf{q}^*, \mathbf{p}^*)$ with the probability r_{accp}, or if it is not accepted, to set $(\mathbf{q}, \mathbf{p})$ to be the next state as in MH sampling. After updating the joint states, simply neglect the momentum vector p to get the desired new q. Note that in the canonical distribution (3.29), p and q are independent, neglecting $\mathbf{p}$ will keeps $\mathbf{q}$ still in the canonical distribution with energy $U(\mathbf{q})$.

Conceptually, evolution of the state $(\mathbf{q}, \mathbf{p})$ should keep $H(\mathbf{q}, \mathbf{p})$ exactly invariant over τ, so the acceptance rate should be 1, however, due to numerical errors of leapfrog method, $H(\mathbf{q}, \mathbf{p})$ is not guaranteed to be constant. Therefore, a MH updating mechanism is needed to restrain the samples in the desired posterior distribution.

Although HMC is conceptually more advanced dealing with complicated distributions than MH sampling, the fictitious time step $\Delta\tau$ and the total time steps in the Hamiltonian dynamics are two hyperparameters, requiring careful tuning. This can be accomplished with a No-U-Turn-Sampler (NUTS) [51], which adjusts these two hyperparameters with the two states $(\mathbf{q}, \mathbf{p})$ during the evolution of the Hamiltonian dynamics. HMC with NUTS requires the gradient, $\frac{\partial U}{\partial q_i}$, which is often intractable in forward simulations with PDE solvers, especially for physical processes with nonlinear terms. Auto-differentiation [68] of neural network surrogates conveniently provides the solution to this problem: the analytical differentiation $\frac{\partial c(\mathbf{x},t)}{\partial \mathbf{m}}$ is easy to be extracted from the neural networks.

We use a `python` package `pyro`[9] for the HMC NUTS sampling with the CNN surrogate model, which provides the gradients $\frac{\partial c(\mathbf{x},t)}{\partial \mathbf{m}}$ needed in Hamiltonian dynamics.

3.4 Numerical Experiments

We use both the CNN-based MCMC with DRAM sampling and CNN-based HMC sampling to identify a contamination source from sparse concentration measurements. A PDE-based transport model used to generate synthetic data is formulated in Section 3.4.1. The CNN-based surrogate for the transport model is developed and analyzed in Section 3.4.2. The performance of our approaches in terms of the accuracy and efficiency vis-à-vis the PDE-based MCMC with DRAM sampling and HMC sampling is discussed in Section 3.4.3.

3.4.1 Contaminant Transport Model

Our solute transport model consists of (3.1)–(3.4) with $R(c) = 0$. A spatially varying hydraulic conductivity field $K(\mathbf{x})$ is shown in Figure 3.6 for a 1000 m by 2000 m rectangular simulation domain discretized into 41×81 cells. We use the fast Fourier transform (see Algorithm 3 in [71]) to

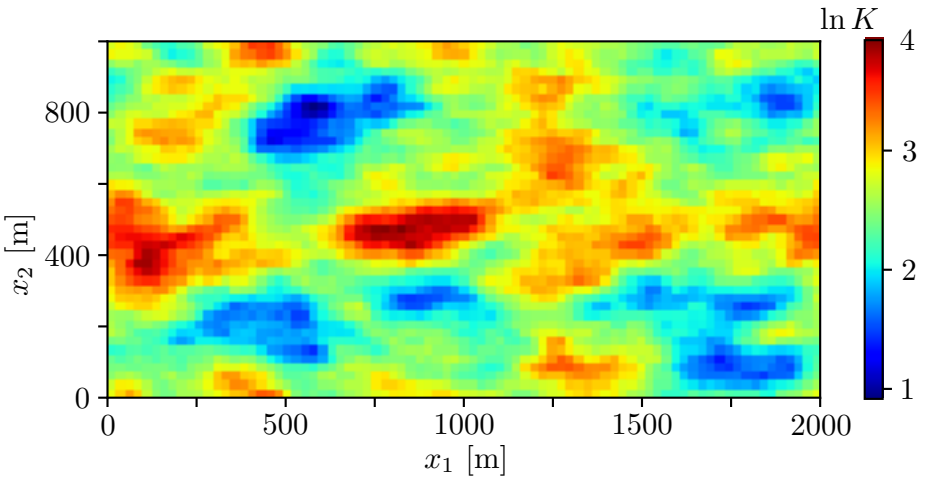

Figure 3.6: Hydraulic conductivity $K(\mathbf{x})$ [m/d], in logarithm scale.

generate $K(\mathbf{x})$ as a rescaled realization of the zero-mean multivariate Gaussian random field with the two-point covariance function:

$$C(\mathbf{x}, \mathbf{y}) = \int_{\mathbb{R}^2} \mathrm{e}^{-2\pi i \langle \mathbf{p}, \mathbf{x}-\mathbf{y} \rangle} |\mathbf{p}|^{-7/4} \mathrm{d}p_1 \mathrm{d}p_2,$$

where $\langle \cdot, \cdot \rangle$ represents the Euclidean inner product on $\mathbb{R}^2$, and $\mathbf{p} = (p_1, p_2)^\top$.

Porosity θ and dispersivities λ_L and λ_T are constant. The values of these and other flow and transport parameters, which are representative of a sandy alluvial aquifer in Southern California [80, 79], are summarized in Table 3.2. Equation (3.4) is used to obtain the dispersion coefficients.

We consider an instantaneous, spatially distributed contaminant release taking place at time $t_0 = 0$. This replaces the source term $r(\mathbf{x}, t) = q_\mathrm{s}(\mathbf{x}, t) c_\mathrm{s}(\mathbf{x}, t)$ in (3.3) with the Dirac-delta source $r(\mathbf{x}, t) = r(\mathbf{x})\delta(t)$ or, equivalently, with an unknown initial contaminant distribution $c_\mathrm{in}(\mathbf{x})$. Our goal is to reconstruct the latter from the noisy concentration data $\bar{c}_{m,i}$ collected at $M = 20$ locations $\{\mathbf{x}_m\}_{m=1}^M$ at $\{t_i\}_{i=1}^I = \{3, 4, \ldots, 18)$ years after the contaminant release ($I = 16$).

Parameter	Value	Units
Porosity, θ	0.3	–
Molecular diffusion, D_m	10^{-9}	m^2/d
Longitudinal dispersivity, α_L	10	m
Dispersivity ratio, α_L/α_T	10	–

Table 3.2: Values of hydraulic and transport parameters, which are representative of sandy alluvial aquifers in Southern California [80, 79].

We used `Flopy` [6], a `Python` implementation of MODFLOW [47] and MT3DMS [8], to solve the flow (3.1) and transport (3.3) equations, respectively. With constant hydraulic head values on

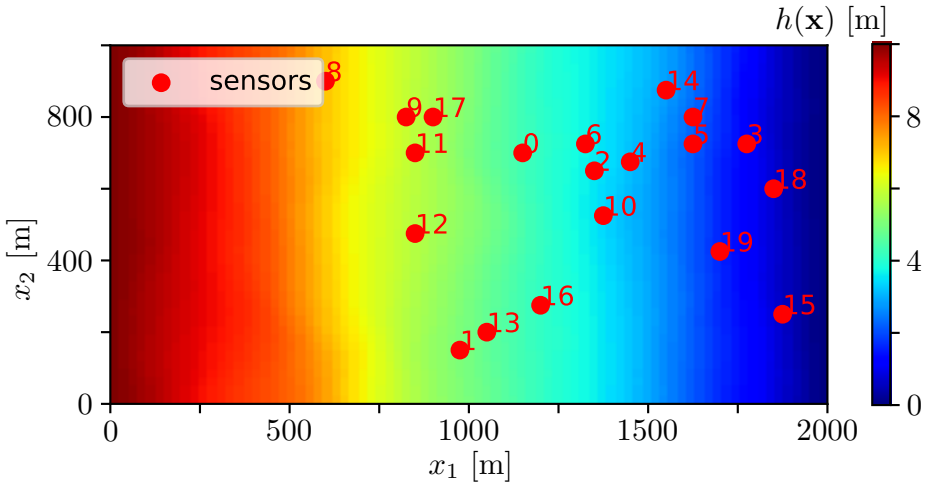

Figure 3.7: Hydraulic head distribution $h(\mathbf{x})$ [m] and locations of 20 observational wells. The flow is driven by constant heads $h_L = 10$ m and $h_R = 0$ maintained at the left and right boundaries, respectively; no-flow boundary conditions are assigned to the upper and lower boundaries.

	$x_{1,1}$	$x_{2,1}$	$x_{1,2}$	$x_{2,2}$	S_1	σ_1	S_2	σ_2
Interval	[0,700]	[50,900]	[0,700]	[50,900]	[0,100]	[13,20]	[0,100]	[13,20]
Truth	325	325	562.5	625	30	15	50	17

Table 3.3: Prior uniform distributions for the meta-parameters $\mathbf{m}$ characterizing the initial contaminant plume (3.38), and the true, yet unknown, values of these parameters.

the left and right boundaries, the head distribution $h(\mathbf{x})$ is shown in Figure 3.7, together with the locations of 20 observational wells.

The initial contaminant distribution consists of N_p co-mingling Gaussian plumes,

$$c_{\text{in}}(x_1, x_2) = \sum_{i=1}^{N_p} S_i \exp\left[-\frac{(x_1 - x_{1,i})^2 + (x_2 - x_{2,i})^2}{2\sigma_i^2}\right], \tag{3.38}$$

each of which has the strength S_i and the width σ_i, and is centered at the point $(x_{1,i}, x_{2,i})$. The true, yet unknown, values of these parameters are collated in Table 3.3 for $N_p = 2$; they are used to generate the measurements $\bar{c}_{m,i}$ by adding the zero-mean Gaussian noise with standard deviation $\sigma_\epsilon = 0.001$. The initial concentration map is shown in Figure 3.8. These data form the 20 breakthrough curves shown in Figure 3.9.

The lack of knowledge about the initial contaminant distribution $c_{\text{in}}(\mathbf{x})$ is modeled by treating these parameters, $\mathbf{m} = (x_{1,i}, x_{2,i}, \sigma_i, S_i)$ with $i = 1$ and 2, as random variables distributed uniformly on the intervals specified in Table 3.3. These uninformative priors are refined as the measurements are assimilated into the model predictions.

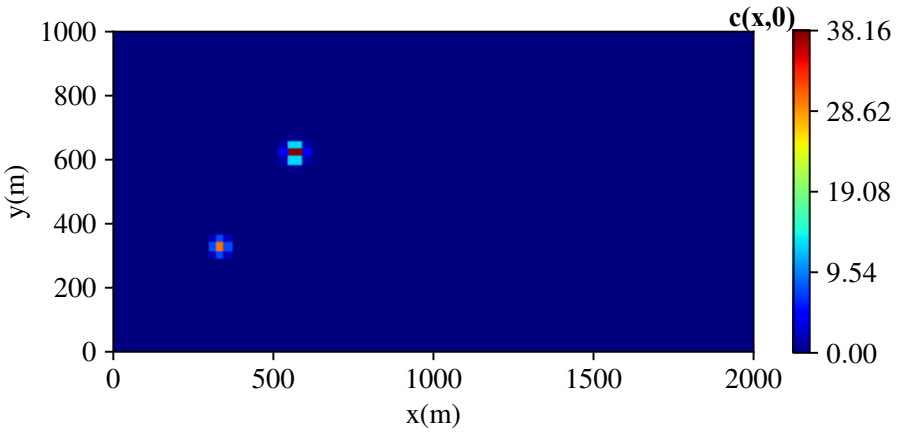

Figure 3.8: True initial concentration c_{in}, the two peaks corresponds to the two Gaussian plumes constructed with the true parameters in Table 3.3.

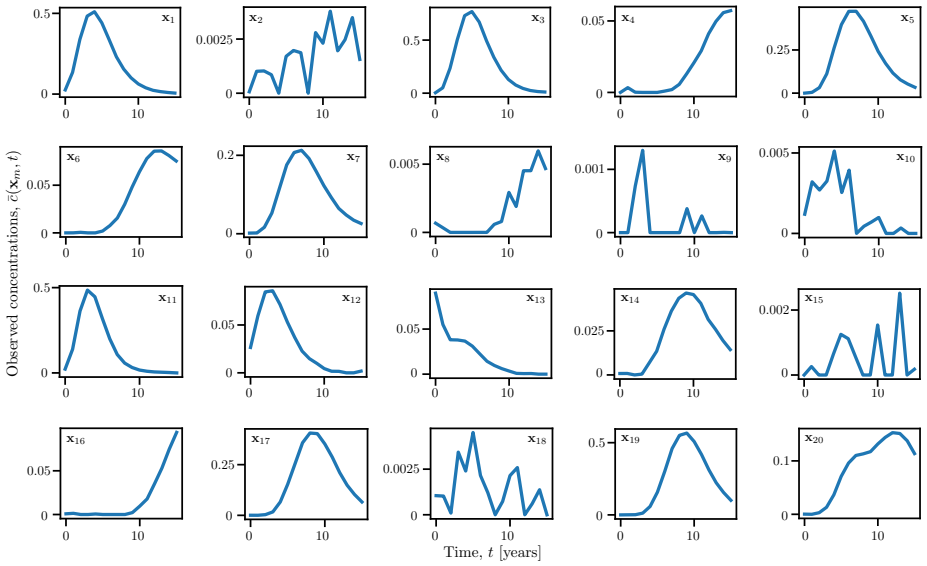

Figure 3.9: Contaminant breakthrough curves $c(\mathbf{x}_m, t)$ observed in the wells whose locations $\mathbf{x}_m$ $(m = 1, \ldots, 20)$ are shown in Figure 3.7.

3.4.2 Construction and Accuracy of CNN Surrogate

As discussed in Section 3.3, although only model predictions at 20 wells are strictly necessary for the inversion, the use of full concentration distributions $c(\mathbf{x}, t_i)$ as output of the CNN-based surrogate has better generalization properties. We used $N = 1600$ solutions (Monte Carlo realizations) of the PDE-based transport model (3.3) for different realizations of the initial condition $c_{\text{in}}(\mathbf{x})$ to train the CNN; another $N_{\text{test}} = 400$ realizations were retained for testing. These 2000 realizations of the initial concentration $c_{\text{in}}(\mathbf{x})$ in (3.38) were generated with Latin hyper-cube sampling of the uniformly distributed input parameters $\mathbf{m}$ from Table 3.3. The CNN contains three dense blocks with $N_l = 6, 12, 6$ internal layers, uses a growth rate $R_g = 40$, and has $N_{\text{in}} = 64$ initial features; it was trained for 300 epochs. The CNN's output is 16 stacked maps of the solute concentration $c(\mathbf{x}, t_i)$ at $t_i = (3, 4, \ldots, 18)$ years after the contaminant release.

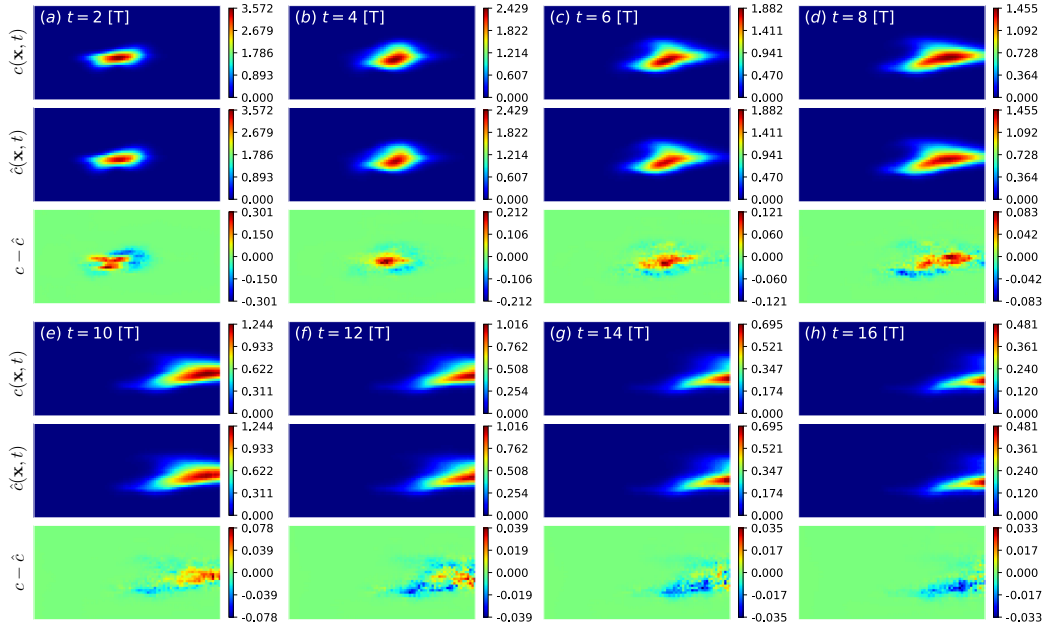

Figure 3.10: Temporal snapshots of the solute concentration alternatively predicted with the transport model (c, top row) and the CNN surrogate ($\hat{c}$, second row) for a given realization of the initial concentration $c_{\text{in}}(\mathbf{x})$. The bottom row exhibits the corresponding errors of the CNN surrogate, $(c - \hat{c})$. The times in the upper left corner correspond to the number of years after contaminant release.

Figure 3.10 exhibits temporal snapshots of the solute concentrations alternatively predicted with the transport model, $c(\mathbf{x}, t_i)$, and the CNN surrogate, $\hat{c}(\mathbf{x}, t_i)$, for a given realization of the initial concentration $c_{\text{in}}(\mathbf{x})$ at eight different times t_i. The root mean square error of the CNN surrogate, $\|c(\mathbf{x}, t_i) - \hat{c}(\mathbf{x}, t_i)\|_2$, falls to 0.023 at the end of the training process. It is worthwhile emphasizing here that the $N = 1600$ Monte Carlo realizations used to train our CNN surrogate are but a small

fraction of the number of forward solves needed by MCMC.

3.4.3 DRAM/HMC Reconstruction of Contaminant Source

We start by analyzing the performance of MCMC with DRAM sampler of $\mathbf{m}$ when the PDE-based transport model (3.3) is used to generate realizations of $c(\mathbf{x}, t_i)$. Because the model is treated as exact, this step allows us to establish the best plume reconstruction provided by our implementation of DRAM. The latter relied on 100000 samples of $\mathbf{m}$, the first half of which was used in the "burn-in" stage and, hence, are not included into the estimation sample set. Figure 3.11 exhibits sample chains for each of the six parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$. Visual inspection of these plots reveals that DRAM does a good job identifying the centers of mass of the two co-mingling plumes, $(x_{1,i}, x_{2,i})$ with $i = 1$ and 2; identification of the spatial extent, σ_i, and strength, S_i, of these plumes is less accurate. However, with 100000 samples, these trace plots still imply the need for more DRAM samples, which is practically prohibitive, shown in the following sections with the indication for the CPU time of this experiment.

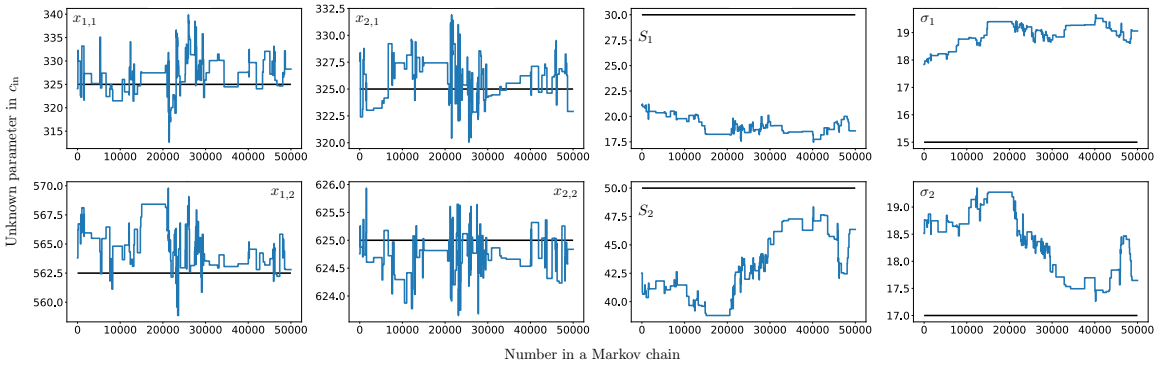

Figure 3.11: DRAM chains of the parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$ obtained by sampling from the transport model (3.3). Each Markov chain represents a parameter value plotted as function of the number of iterations (links in the chain). The black horizontal lines are the true values of each parameter.

Table 3.4 provides a more quantitative assessment of the performance of the PDE-based DRAM. The standard deviations of the DRAM estimates of the plumes' centers of mass, $(x_{1,i}, x_{2,i})$, is no more than 1% of their respective means, indicating high confidence in the estimation of these key parameters. The standard deviations of the other parameter estimates, relative to their respective means, are appreciably higher. Also shown in Table 3.4 are Sokal's adaptive truncated periodogram estimator of the integrated autocorrelation time τ [118], and the Geweke convergence diagnostic p [36]. These quantities are routinely used to diagnose the convergence of Markov chains. The former provides an average number of dependent samples in a chain that contain the same information as one independent sample; the latter quantifies the similarity between the first 10% samples and the

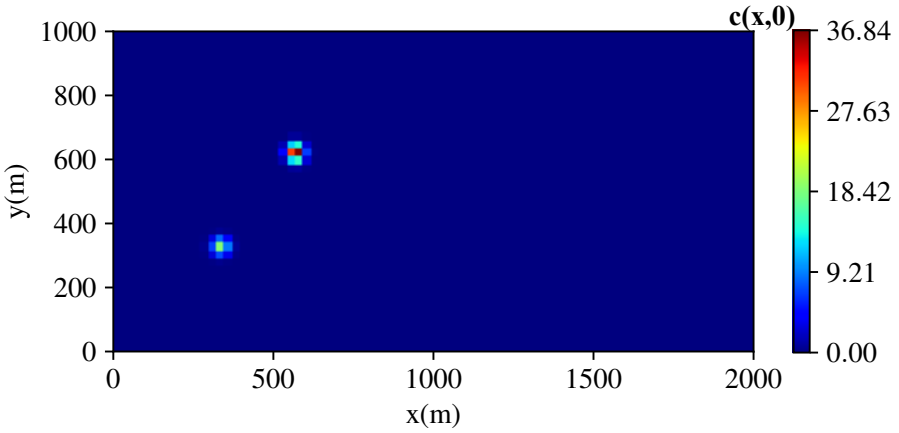

Figure 3.12: The initial concentration c_{in}, constructed with the two Gaussian plumes computed with the DRAM MCMC chain(second half) obtained by sampling from the PDE model. The mean values of the parameters are shown in Table 3.4.

Parameter	True value	Mean	Std	τ	p
$x_{1,1}$	325	327.5836	3.3924	1046.3394	0.9991
$x_{2,1}$	325	325.7773	1.6108	1289.5577	0.9929
$x_{1,2}$	562.5	564.3320	1.9967	2218.9018	0.9881
$x_{2,2}$	625	624.7743	0.3203	402.0658	0.9998
S_1	30	18.6853	0.5007	1713.8339	0.9699
σ_1	15	19.1371	0.2365	2172.9087	0.9837
S_2	50	44.3071	2.8493	4441.9589	0.7632
σ_2	17	18.0939	0.5932	4409.0626	0.8832
M_1	20.4244	20.6709	–	–	–
M_1	43.5802	43.74	–	–	–

Table 3.4: DRAM chain statistics—mean, standard deviation, integrated autocorrelation time τ, and Geweke convergence diagnostic p—of the parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$ obtained by sampling from the PDE model. Also shown is the total contaminant mass of the two co-mingling plumes, M_1 and M_2.

last 50% samples. The initial concentration map constructed with the two Gaussian plumes with the parameter in this MCMC chain is shown in Figure 3.12.

Although somewhat less accurate, the estimates of the spatial extent, σ_i, and strength, S_i, of the co-mingling plumes is more than adequate for field applications. Their estimation errors cannot be eliminated with more computations, as suggested by a very large number of samples used in our DRAM. Instead, they reflect the relative dearth of information provided by a few sampling locations.

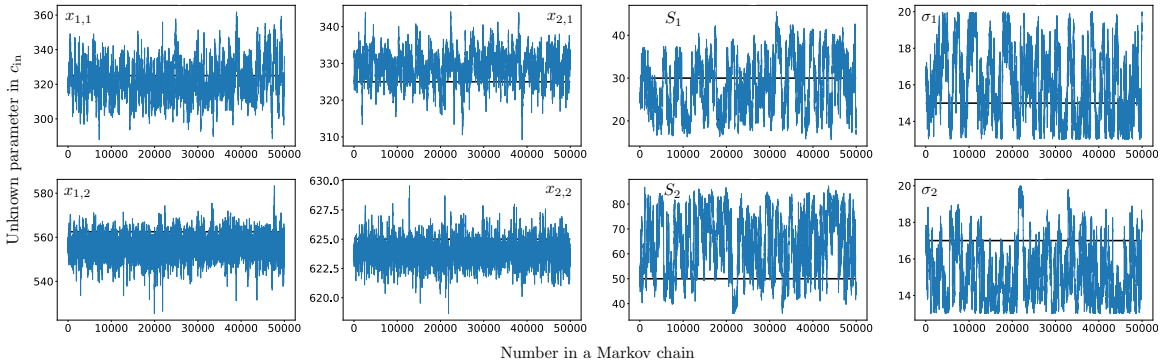

Figure 3.13: DRAM chains of the parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$ obtained by sampling from the CNN surrogate (3.10). Each Markov chain represents a parameter value plotted as function of the number of iterations (links in the chain). The black horizontal lines are the true values of each parameter.

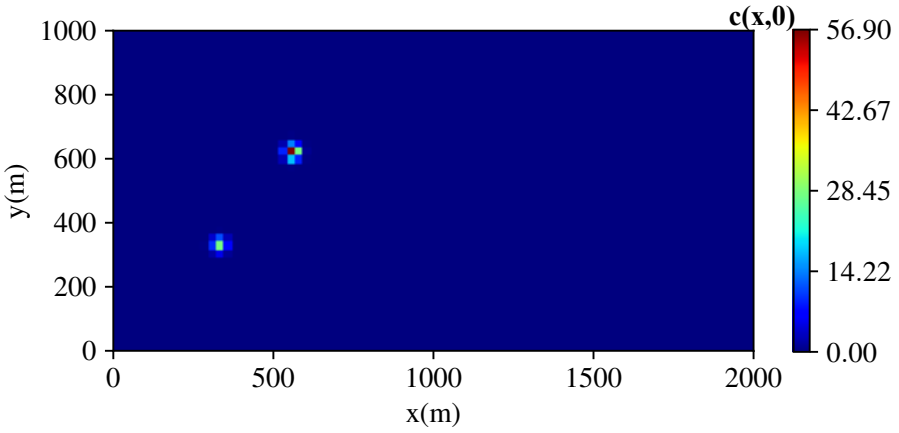

Figure 3.14: The initial concentration c_{in}, constructed with the two Gaussian plumes computed with the DRAM MCMC chain(second half) obtained by sampling from the CNN surrogate model. The mean values of the parameters are shown in Table 3.5.

Next, we repeat the MCMC procedure but using the CNN surrogate to generate samples. Figure 3.13 exhibits the resulting MCMC chains of the parameters $\mathbf{m}$, i.e., the parameter values plotted

Parameter	True value	Mean	Std	τ	p
$x_{1,1}$	325	322.3274	9.8827	189.8946	0.9944
$x_{2,1}$	325	328.8859	3.9956	231.9033	0.9992
$x_{1,2}$	562.5	555.4074	4.3167	35.8577	0.9983
$x_{2,2}$	625	623.8933	0.8944	43.2115	0.9999
S_1	30	28.4441	6.4531	514.4594	0.8100
σ_1	15	15.9822	1.9291	537.7868	0.9094
S_2	50	64.6830	12.1613	540.6132	0.9962
σ_2	17	15.1550	1.6076	543.3779	0.9964
M_1	20.4244	21.9306	–	–	–
M_2	43.5802	44.8789	–	–	–

Table 3.5: DRAM chain statistics—mean, standard deviation, integrated autocorrelation time τ, and Geweke convergence diagnostic p—of the parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$ obtained by sampling from the CNN surrogate. Also shown is the total contaminant mass of the two co-mingling plumes, M_1 and M_2.

as function of the number of samples N (excluding the first 50000 samples used in the burn-in stage). Because of the prediction error of the CNN surrogate, the chains differ significantly from their PDE-based counterparts in Fig. 3.11. They are visibly better mixed, an observation that is further confirmed by the fact that the integrated autocorrelation times τ in Table 3.5 are much smaller than those reported in Table 3.4. The initial concentration map constructed with the two Gaussian plumes with the parameter in this MCMC chain is shown in Figure 3.14. However, the standard deviations (Std) for the parameter estimators are slightly larger than those obtained with the PDE-based MCMC; this implies that the CNN prediction error undermines the ability of MCMC to narrow down the posterior distributions. The posterior PDFs for the centers of mass of the two commingling plumes, $(x_{1,i}, x_{2,i})$, are shown in Figure 3.15 and Figure 3.16. The discrepancy between the actual and reconstructed (as the means of these PDFs) locations is within 7 m; it is of negligible practical significance.

Comparison of Table 3.4 and Table 3.5 reveal that, similar to the PDE-based sampler, the CNN-based sampler provides more accurate estimates of the source location $(x_{1,i}, x_{2,i})$ than of its spread (σ_i) and strength (S_i). However, in practice, one is more interested in the total mass of the released contaminant (M) rather than its spatial configuration (characterized by σ_i and S_i). The mass of each of the commingling plumes in (3.38), M_1 and M_2, is:

$$M_i = \theta \int_{\Omega_i} c_{\text{in}}(\mathbf{x}) \mathrm{d}\mathbf{x}, \qquad \Omega_i : [x_{1,i} \pm 100] \times [x_{2,i} \pm 100], \qquad i = 1, 2. \tag{3.39}$$

Both the PDE- and CNN-based MCMC strategies yield accurate estimates of M_1 and M_2 (Tables 3.4 and 3.5).

We also present the HMC sampling using the CNN surrogate to generate samples and compute the gradient. Results with only 2000 samples are shown in Figure 3.17, exhibiting the resulting HMC

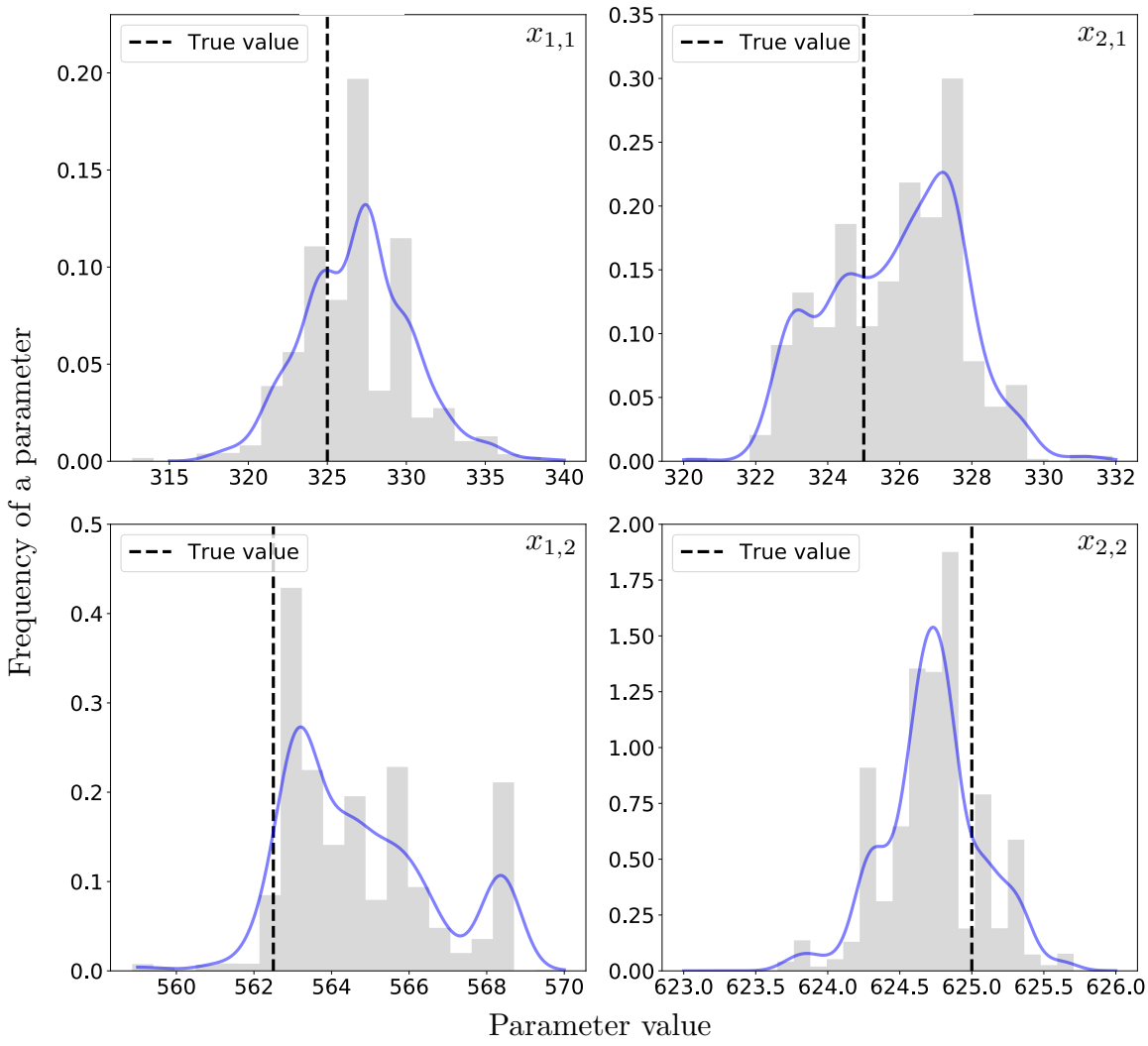

Figure 3.15: Probability density functions (solid lines) and histograms (gray bars) of the centers of mass of the two commingling spills, $(x_{1,1}, x_{2,1})$ and $(x_{1,2}, x_{2,2})$, computed with MCMC drawing samples from the PDE-based transport model. Vertical dashed lines represent the true locations.

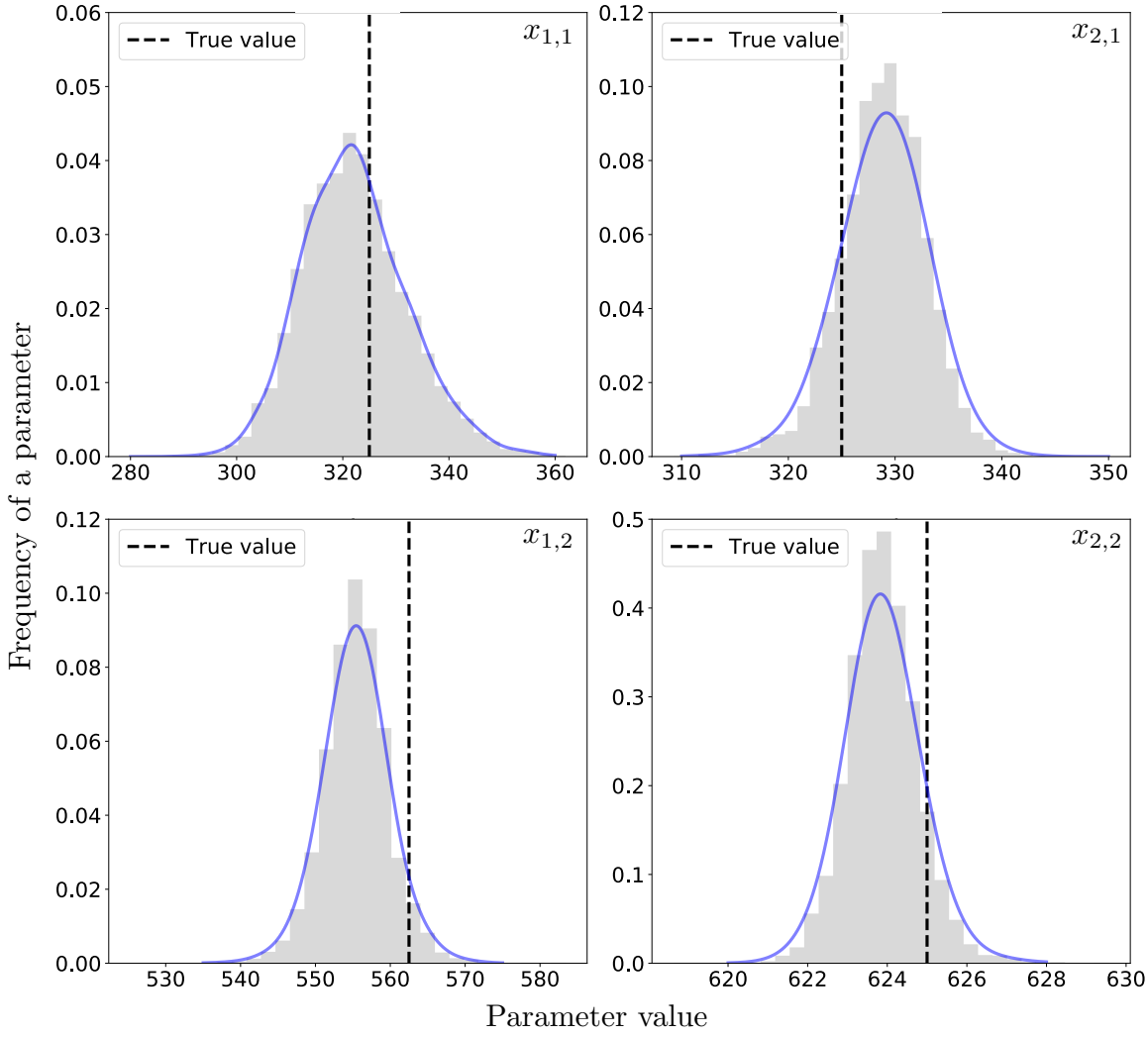

Figure 3.16: Probability density functions (solid lines) and histograms (gray bars) of the centers of mass of the two commingling spills, $(x_{1,1}, x_{2,1})$ and $(x_{1,2}, x_{2,2})$, computed with DRAM drawing samples from the CNN surrogate. Vertical dashed lines represent the true locations.

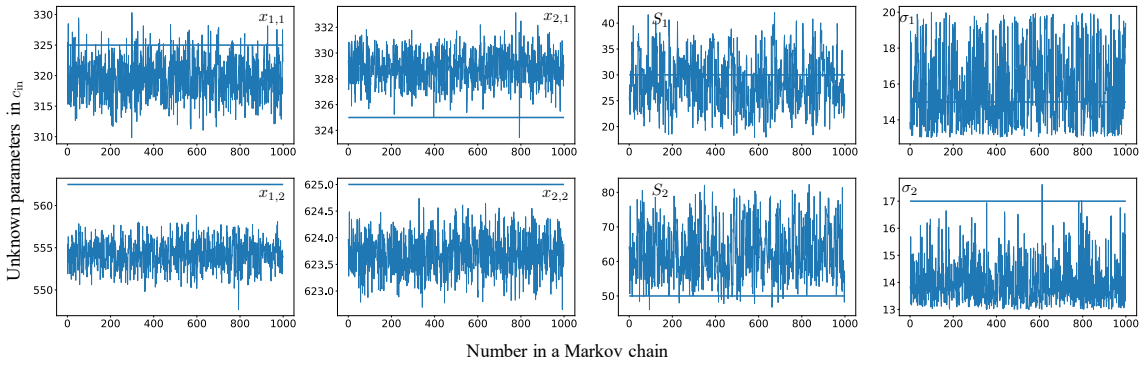

Figure 3.17: HMC-NUTS chains of the parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$ obtained by sampling from the CNN surrogate (3.10). Each Markov chain represents a parameter value plotted as function of the number of iterations (links in the chain). The horizontal lines are the true values of each parameter.

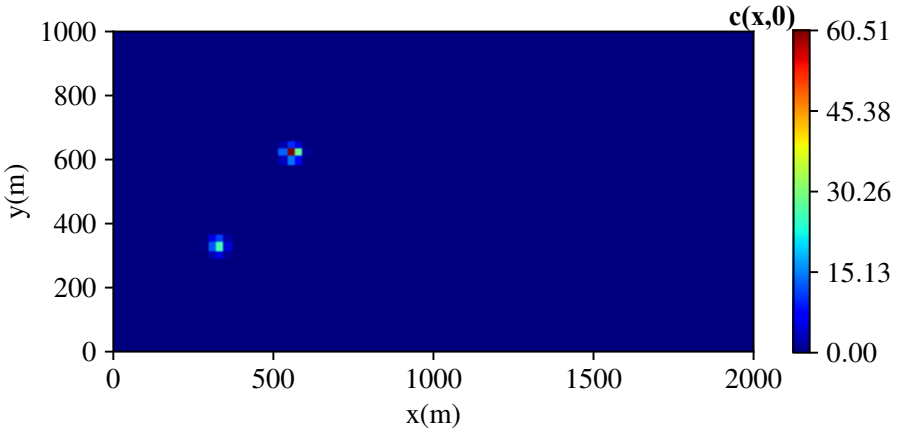

Figure 3.18: The initial concentration c_{in}, constructed with the two Gaussian plumes computed with the HMC chain(second half) obtained by sampling from the CNN surrogate model. The mean values of the parameters are shown in Table 3.6.

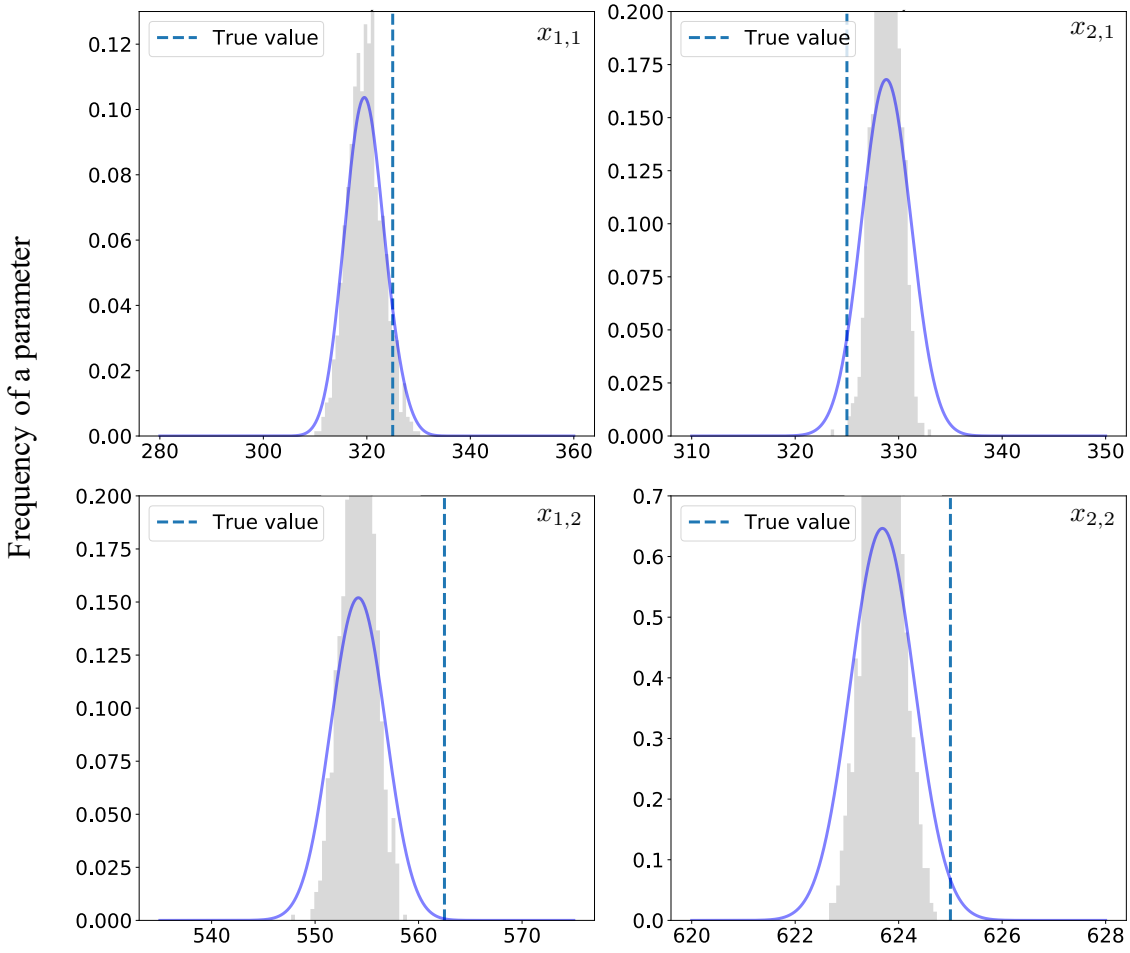

Figure 3.19: Probability density functions (solid lines) and histograms (gray bars) of the centers of mass of the two commingling spills, $(x_{1,1}, x_{2,1})$ and $(x_{1,2}, x_{2,2})$, computed with HMC-NUTS drawing samples from the CNN surrogate. Vertical dashed lines represent the true locations.

chains of the parameters $\mathbf{m}$, i.e., the parameter values plotted as function of the number of samples N (excluding the first 1000 samples used in the burn-in stage). The prediction error of the CNN surrogate has similar effect on the shape of the trace plots to Figure 3.13, visibly better mixed too. The PDFs shown in Figure 3.19 with only 2000 samples are already very smooth and concentrated at the peak location compared to DRAM-CNN results in Figure 3.16. The chain statistics are summarized in Table 3.6, the estimated mean is slightly farther from the reference value compared to the DRAM-CNN results shown in Table 3.5, but the standard deviation is an order of magnitude smaller. The initial concentration map constructed with the two Gaussian plumes with the parameter in this MCMC chain is shown in Figure 3.18. The autocorrelation τ for all estimated parameters are much smaller; the Geweke convergence diagnostic p are all very close to 1. Visualization of the autocorrelation for the three samplers are shown in Figure 3.20, Figure 3.21, and Figure 3.22. The autocorrelation of these three chains show clearly that the HMC-CNN sampler with only 2000 samples reaches convergence, saving an order of magnitude samples. These statistics all prove that the convergence of the HMC-CNN sampler is much faster than that of DRAM sampling with either PDE based forward model and CNN surrogate forward model. On the other hand, we would like to indicate that the samples of $x_{1,2}, x_{2,2}$ did not cover the reference value, as shown in Figure 3.19, the discrepancy between the reference value and the sample mean is up to 10 m for $x_{1,2}$ in a field that is 2000×1000 m^2, which can be a practical issue in field applications.

Parameter	True value	Mean	Std	τ	p
$x_{1,1}$	325	319.66525	3.3039	1.09175	0.997543
$x_{2,1}$	325	328.80032	1.2776	1.24368	0.99958
$x_{1,2}$	562.5	554.15515	1.6658	1.26398	0.99993
$x_{2,2}$	625	623.68756	0.3582	1.19211	0.99994
S_1	30	28.5775	4.8943	3.45711	0.99916
σ_1	15	15.8092	2.0209	1.88012	0.98740
S_2	50	62.712563	7.6878	2.42575	0.98592
σ_2	17	13.965155	0.8016	1.16434	0.99879
M_1	20.4244	22.1330	-	-	-
M_2	43.5802	44.0599	-	-	-

Table 3.6: HMC chain statistics—mean, standard deviation, integrated autocorrelation time τ, and Geweke convergence diagnostic p—of the parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$ obtained by sampling from the CNN surrogate. Also shown is the total contaminant mass of the two co-mingling plumes, M_1 and M_2.

3.4.4 Computational Efficiency of MCMC with CNN Surrogate

Our CNN-based DRAM is about 20 times faster than DRAM with the PDE-based transport model (Table 3.7). This computational speed-up can be attributed to either the algorithmic improvement or the different hardware architecture or both. That is because while the off-the-shelf PDE-based software utilizes central processing units (CPU), NN training takes place on graphics processing

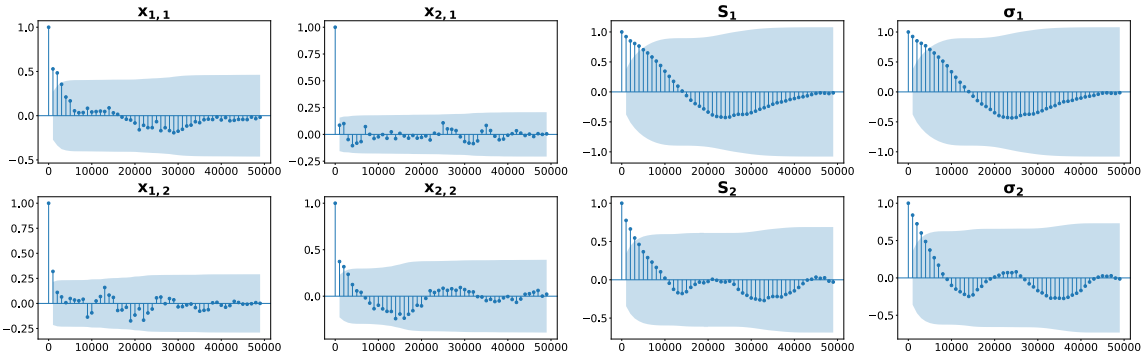

Figure 3.20: Auto-correlation plots of the of the parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$, computed with DRAM-MCMC drawing samples from the PDE-based transport model. Half of the chain with 50000 samples are used for these plots.

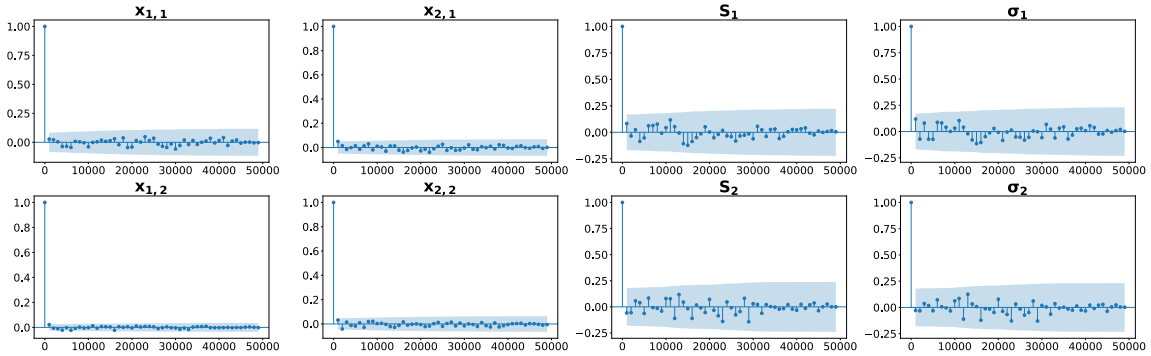

Figure 3.21: Auto-correlation plots of the of the parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$, computed with DRAM-MCMC drawing samples from the CNN surrogate. Half of the chain with 50000 samples are used for these plots.

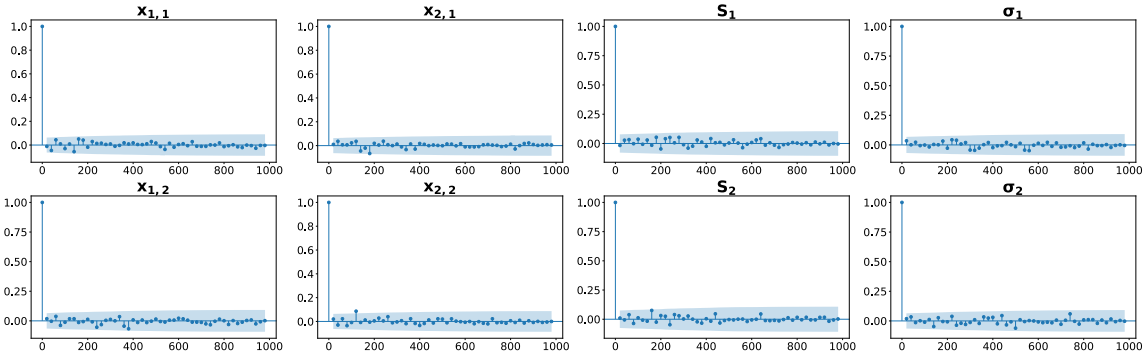

Figure 3.22: Auto-correlation plots of the of the parameters $\mathbf{m}$ characterizing the initial plume configuration $c_{\text{in}}(\mathbf{x})$, computed with HMC-NUTS drawing samples from the CNN surrogate. Half of the chain with 1000 samples are used for these plots.

units (GPUs), e.g., within the `GoogleColab` environment used in our study, without much effort on the user's part. To disentangle these sources of computational efficiency, we also run the CNN-based DRAM on the same CPU architecture used for the PDE-based DRAM. Table 3.7 demonstrates that the CNN-based MCM ran on CPU is about twice faster than the PDE sampler. This indicates that the computational speed-up of the CNN-based sampler is in large part due to the use of GPUs for CNN-related computations. One could rewrite PDE-based transport models to run on GPUs, but it is not practical. At the same time, no modifications or special expertise are needed to run the `Pytorch` implementation [101] of neural networks on GPUs. The CNN-based HMC though, takes much longer time to generate each sample in the HMC chain, notably slower than both PDE-based DRAM sampler and CNN-DRAM sampler. However, the convergence of the chain is much faster given that the chain is well-mixed with only 1000 burn in samples.

	T_{run}	T_{train}	T_{ave}
PDE	101849.0	0	1.01849
CNN on GPU	1101.7	4007.4	0.05109
CNN on CPU	37450.0	4007.4	0.41457
HMC-CNN on GPU	6824.0	4007.4	5.41550

Table 3.7: Total run time (in seconds) of the MCMC samplers, T_{run}, based on the PDE-based transport model and its CNN surrogate. The PDE sampler uses CPUs; the CNN sampler is trained and simulated on GPUs provided by `GoogleColab`; for the sake of comparison, also reported is the run time of the CNN sampler on the CPU architecture used for the PDE-based sampler. In the first three cases, DRAM consists of $N_{\text{sam}} = 10^5$ samples. The average run-time per sample, T_{ave}, is defined as $T_{\text{ave}} = (T_{\text{run}} + T_{\text{train}})/N_{\text{sum}}$, where T_{train} is the CNN training time. HMC-CNN on GPU reports the time required for 2000 samples, the CNN surrogate used in this scenario is the same as the previous two case, and is run on GPU.

3.5 Conclusions and Discussion

We proposed an MCMC approach that uses DRAM sampling and draws samples from a CNN surrogate of a PDE-based model. We also tested the CNN surrogate model with an HMC sampler. The approach was used to reconstruct contaminant release history from sparse and noisy measurements of solute concentration. In our numerical experiments, water flow and solute transport take place in a heterogeneous two-dimensional aquifer; the goal is to identify the spatial extent and total mass of two commingling plumes at the moment of their release into the aquifer. Our analysis leads to the following major conclusions.

1. The CNN-based DRAM is able to identify the locations of contaminant release, as quantified by the centers of mass of commingling spills forming the initial contaminant plume.

2. Although somewhat less accurate, the estimates of the spread and strength of these spills is adequate for field applications with DRAM samplers. Their integral characteristics, the total mass of each spill, are correctly identified.

3. The estimation errors cannot be eliminated with more computations. Instead, they reflect both the ill-posedness of the problem of source identification and the relative dearth of information provided by sparse concentration data.

4. Replacement of a PDE-based transport model with its CNN-based surrogate increases uncertainty in, i.e., widens the confidence intervals of, the source identification.

5. The CNN-based DRAM is about 20 times faster than MCMC with the high-fidelity transport model. This computational speed-up is in large part due to the use of GPUs for CNN-related computations, while the PDE solver utilizes CPUs.

6. The CNN-based HMC sampler is notably slower in terms of obtaining each sample due to the Hamiltonian dynamics evolution. The convergence of the chain is much faster than both PDE-based and CNN-based DRAM sampler. However, although the samples of the release locations are close to the reference value, they did not cover the reference value, which can be a practical issue in field application.

While we relied on a CNN to construct a surrogate of the PDE-based model of solute transport, other flavors of NNs could have be used for this purpose. We are not aware of published comparisons of alternative NNs in the context of image-to-image prediction, which is required by our DRAM method. In the somewhat related context of spectrum sensing [136], the comparison of a fully connected neural network (FNN), a recurrent neural network (RNN), and a CNN revealed the FNN to have small utility for ordered and correlated samples like images; the CNN and RNN to exhibit a comparable performance in terms of accuracy, and the RNN to be more efficient in terms of memory requirements.

In general, the direct comparison of the performance of a FNN and a CNN on the same task is not helpful and can be misleading because of the freedom of the architecture of each network and the presence of multiple tuning parameters in both. However, the results reported in Section 3.3.2 suggest that a FNN would contain significantly more parameters given the size of the input and output images. This applies even to a relatively shallow FNN. Some studies in image classification, e.g., [18], claim that, relative to FNNs, CNNs require more training data to achieve convergence and avoid overfitting. Even if this conclusion generalizes to our application it is of little practical significance, because we found the combined cost of the training-data generation and NN training to be significantly lower than the cost of MCMC sampling.

Properly trained autoregressive models and RNNs can be a strong competitor to CNNs, because they perform like a fixed time-step predictor and, consequently, *might* generalize better. RNNs are likely to be more expensive because of higher prediction frequency, but require less memory for each prediction. Our implementation of CNNs utilized a parallel GPU architecture to carry out convolutional operations. However, since GPUs have become more affordable, this drawback can be ignored.

Our computational examples deal with an instantaneous contaminant release. Because a CNN has been shown to provide an accurate surrogate of the PDE-based transport model with temporally distributed sources [94] and because MCMC is known to accurately reconstruct prolonged contaminant release history [7], our CNN-based MCMC is expected to provide comparable computational gains when used to identify prolonged contaminant releases.

Chapter 4

Joint Conductivity and Source Identification

In this chapter, we perform inverse modeling to construct the conductivity field and the contaminant release history with noisy solute concentration and hydraulic head data. We utilize CAAE parameterization to obtain a reduced-order representation of the hydraulic conductivity field, and the data assimilation method ESMDA with CAAE and a DenseED surrogate to obtain an approximation of the posterior distribution of the model parameters.

4.1 Introduction

Design of regulatory and remedial actions for contaminated soils and aquifers rely on reconstruction of the contaminant release history. Given subsurface heterogeneity, this task is inseparable from the need to identify hydraulic and transport properties of the subsurface environment. Both tasks have to contend with sparse and noisy measurements collected many years or decades after the contamination event took place. Prior to recent breakthroughs in computer architecture and algorithmic development, this joint inversion of hydraulic and water-quality data for real-world problems was so demanding computationally as to defy a solution unless dramatic (and often unrealistic) simplifications of the problem were made. For example, past efforts to reconstruct a contaminant release history found it necessary to assume solute migration to be one-or two-dimensional and subsurface properties, such as hydraulic conductivity $K(\mathbf{x})$, to be known with certainty [4, 117, 137, among many others]. Yet, aquifers are seldom, if ever, homogeneous, with $K(\mathbf{x})$ often varying by orders of magnitude within the same aquifer and exhibiting highly non-Gaussian, multimodal behavior [123, 129, 135]. Likewise, while the assumption of two-dimensional groundwater flow is often valid, accounting for the three-dimensional nature of contaminant migration is essential to prediction

accuracy.

Our effort in joint inversion of hydraulic conductivity and contaminant release history from error-prone measurements of hydraulic head and solute concentration revolves around two challenges. The first is to describe the unknown non-Gaussian heterogeneous conductivity field with an adequate prior distribution. The second is to estimate a large number of unknown parameters in the inverse problem.

To tackle the first challenge, a parameterization of the high-dimensional conductivity field with a low-dimensional latent variable is commonly used [81, 142]. Parameterizations based on the principle component analysis (PCA) [114, 125] perform well for Gaussian random fields, but require ad-hoc modifications for non-Gaussian fields [82]. DNN-based parameterizations eliminate the need for the Gaussianity assumption [16, 82]. Two popular methods of this class are generative adversarial network (GAN) [40] and variational autoencoders [61]. Both produce a DNN that learns a two-way mapping between the training conductivity fields and the low-dimensional latent variable. Random realizations from the latent variable distribution can be decoded to a conductivity field that is statistically similar to those drawn from the training data set. In addition to the large reduction in the number of the unknown parameters, such parameterizations make it feasible to tackle the latent variable distribution, which is typically a standard normal by construction. This simplicity, in turn, facilitates the solution of the inverse problem with ensemble methods discussed below.

The second challenge, high dimensionality of the parameter space, manifests itself in significant computational burden of an inversion procedure. Parameter estimation, which lies at the heart of an inverse problem, is achieved by matching the noisy measurements with the prediction of a flow and solute transport model. Strategies for solving typically ill-posed inverse problems fall into two main categories, deterministic and probabilistic. Deterministic methods, such as least square regression [128] and hybrid optimization with a genetic algorithm [5, 77], seek a "best" estimate of the unknown parameters, without quantifying the uncertainty inherent in this type of problems. Probabilistic methods, such as Markov chain Monte Carlo or MCMC [34] and data assimilation via Kalman filters [31, 32, 133, 134] and their variants [30, 140], overcome this shortcoming of their deterministic counterparts. Yet, the high cost of necessary repeated forward solves undermines their utility for large, complex inverse problems, unless dedicated high-performance computing facilities are available for the task.

Two complementary strategies can be deployed to alleviate this cost. The first aims to reduce the number of forward simulations needed for an inversion algorithm to converge. The second seeks to reduce the computational cost of each forward solve. Efforts in the former direction include the design of efficient MCMC variants, such as delayed rejection adaptive Metropolis (DRAM) sampling [43, 44] which slightly outperforms a random walk Metropolis-Hastings MCMC in terms of efficiency [138, 143, 132]. Gradient-based MCMC methods, such as hybrid Monte Carlo (HMC) sampling [7], converges faster than these and other MCMC variants. However, computation of the

gradient of a Hamiltonian dynamical system is prohibitive for high-dimensional transport problems. Learning on statistical manifolds provides another possible solution [14, 13]. Ensemble-based inversion methods are generally faster since they allow nearly perfect parallelization, because of the independence of samples in the ensemble. Variants of Kalman filters, such as iterative Ensemble Kalman filter (IEnKF), have been used for estimation of three-dimensional heterogeneous permeability fields [17]. A relatively new variant of Kalman filter, ensemble smoother with multiple data assimilation (ESMDA) [30], has gained popularity in subsurface flow history matching [122, 59, 54].

In terms of inversion complexity, we subdivide recent groundwater-related studies into three categories: estimation of hydraulic conductivity from measurements of hydraulic head and, optionally, of solute concentration [95, 56]; estimation of contaminant release history from concentration measurements, for known flow and transport parameters [143, 138]; and estimation of both contaminant release history and hydraulic conductivity from hydraulic head and solute concentration data in two- [94, 134, 58] and three-dimensional [57] aquifers. We briefly discuss the latter category to highlight the novelty of our approach.

A low-dimensional representation of the random log-normal conductivity obtained via the Karhunen-Loève expansion (KLE) in [94] loses its attractiveness if the subsurface environment is highly heterogeneous, exhibiting short correlation lengths and multimodal statistics; additionally, this study relies on a linear transport model. The deep learning-based strategies of ensemble inversion were adopted in [134, 58] to estimate both a non-Gaussian conductivity field and the source of contamination, yet their reported accuracy is relatively low. In the adjacent field of petroleum engineering, CNN post-processing of PCA (CNN-PCA) parameterization and ESMDA were used to estimate both a channelized permeability and the oil/water rate [122]. However, this application deals with an observable quantity (the oil/water production rate), while ours has to contend with an unobservable one (the location and strength of a contaminant release).

To address the shortcomings of the joint inversion strategies mentioned above, we use a convolutional adversarial autoencoders (CAAE) to parameterize a non-Gaussian conductivity field [95], train a surrogate dense encoder-decoder DNN to replace the PDE-based model of subsurface flow and transport, and apply the ESMDA inversion framework to identify the spatiotemporally extended source of contamination and the latent variables representing the conductivity field. We posit that combination of these three components, which yields the method we refer to as CAAE–DenseED–ESMDA, provides a fast and robust inversion solution.

In Section 4.2 we formulate the problem of joint reconstruction of hydraulic conductivity field and contaminant release history from sparse and noisy measurements of hydraulic head and solute concentration. Our inversion strategy, combining CAAE parameterization of the conductivity field (Section 4.3.2), a convolutional DNN surrogate of the flow and transport model (Section 4.3.3), and the ESMDA inversion method (Section 4.3.1), is described in Section 4.3. Results of our numerical experiments are reported in Section 4.4; they demonstrate that our method is about 8 times faster

than CAAE-ESMDA with the PDE-based flow and transport model. Main conclusions drawn from this study are summarized in Section 4.5.

4.2 Problem Formulation

The problem formulation consists of the description of a reactive transport model (Section 4.2.1) and the specification of a data model (Section 4.2.2).

4.2.1 Contaminant Transport Model

We consider transport of a reactive solute in a three-dimensional steady-state groundwater flow field. The latter is described by:

$$\nabla \cdot (K \nabla h) = 0, \qquad \mathbf{x} = (x_1, x_2, x_3)^\top \in \Omega \subset \mathbb{R}^3, \tag{4.1}$$

where $K(\mathbf{x})$ is the hydraulic conductivity of the aquifer Ω, and $h(\mathbf{x})$ is the hydraulic head. This PDE is subject to appropriate boundary conditions on the simulation domain boundary $\partial\Omega$. After the flow equation is solved, the average pore velocity $\mathbf{u}(\mathbf{x}) = (u_1, u_2, u_3)^\top$ is computed from the Darcy law,

$$\mathbf{u} = -\frac{K}{\theta} \nabla h, \tag{4.2}$$

where $\theta(\mathbf{x})$ is the aquifer's porosity.

Starting at some unknown time t_0, a contaminant with volumetric concentration c_s enters the aquifer through either point-wise or spatially distributed sources $\Omega_\mathrm{s} \subset \Omega$. The contaminant is released for unknown duration T with unknown intensity $q_\mathrm{s}(\mathbf{x}, t)$ (volumetric flow rate per unit source volume), such that $q_\mathrm{s}(\mathbf{x}, t) \neq 0$ for $t_0 \leq t \leq t_0 + T$. The contaminant is advected by the flow, while undergoing hydrodynamic dispersion and sorption to the solid matrix with rate R_n. Without loss of generality, the spatiotemporal evolution of the contaminant's volumetric concentration $c(\mathbf{x}, t)$ is described by an advection-dispersion-reaction equation:

$$\frac{\partial \theta c}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c) - \nabla \cdot (\theta \mathbf{u} c) - R_n(c) + q_\mathrm{s} c_\mathrm{s}, \qquad \mathbf{x} \in \Omega, \quad t > t_0, \tag{4.3}$$

The dispersion coefficient $\mathbf{D}$, a semipositive second-rank tensor. If the coordinate system is aligned with the mean flow direction, such that $\mathbf{u} = (u \equiv |\mathbf{u}|, 0, 0)^\top$, then the components of this tensor are:

$$D_{11} = \theta D_\mathrm{m} + \alpha_L u, \quad D_{22} = \theta D_\mathrm{m} + \alpha_T u, \quad D_{33} = \theta D_\mathrm{m} + \alpha_C u, \quad D_{ij} = \theta D_\mathrm{m} \ \text{ for } \ i \neq j, \tag{4.4}$$

where D_m is the coefficient of molecular diffusion for the contaminant in free water; α_L is the longitudinal dispersivity; and α_T and α_C are transverse dispersivities in the x_2 and x_3 directions,

respectively. The chemical reactions considered represent sorption of the dissolved contaminant onto the solid surface of the porous media. Thus, the reaction terms $R_n(c)$ has the form:

$$R_n(c) = -\rho_b \frac{\partial \tilde{c}}{\partial t}. \tag{4.5}$$

We assume the system to be in local chemical equilibrium, i.e., sorption to be much faster than advection and dispersion, We also assume that sorption does not affect the porosity θ, which remains constant throughout the simulations. With these assumptions, (4.3) reduces to:

$$R\theta \frac{\partial c}{\partial t} = \nabla \cdot (\theta \mathbf{D} \nabla c) - \nabla \cdot (\theta \mathbf{u} c) + q_s c_s \tag{4.6}$$

wherein $R(c)$ is the dimensionless retardation factor defined as:

$$R = 1 + \frac{\rho}{\theta} \frac{\partial \tilde{c}}{\partial c}. \tag{4.7}$$

A sorption isotherm defines the relationship between the sorbed concentration, $\tilde{c}$, and the dissolved concentration, c. Among the popular isotherms—linear, Langmuir, and Freundlich—we adopt the latter, for the sake of concreteness. According to the Freundlich isotherm,

$$\tilde{c} = K_f c^a, \tag{4.8}$$

where K_f is the Freundlich constant, $(L^3 M^{-1})^a$; and a is the Freundlich exponent. The units of all the transport quantities are summarized in Table 4.1.

Table 4.1: *Quantities in the transport model* (4.6) *and their units.*

Term	Physical quantity	Units
c	dissolved concentration	ML^{-3}
θ	porosity of the subsurface medium	-
x_i	the distance along the respective Cartesian coordinate axis	L
D_{ij}	hydrodynamic dispersion coefficient tensor	$\mathrm{L}^2\mathrm{T}^{-1}$
u_i	pore water velocity	LT^{-1}
q_s	volumetric flow rate per volume, sources (+) and sinks (−)	T^{-1}
c_s	concentration of source or sink flux	ML^{-3}
R_n	chemical reaction term	$\mathrm{ML}^{-3}\mathrm{T}^{-1}$
ρ_b	bulk density of the medium	ML^{-3}
$\tilde{c}$	concentration sorbed	ML^{-3}
K_f	Freundlich constant	$(\mathrm{L}^3\mathrm{M}^{-1})^a$
a	Freundlich exponent	-

4.2.2 Parameters of Interests

Our goal is to identify the conductivity field $K(\mathbf{x})$ and the contaminant source $q_s c_s(\mathbf{x}, t)$, given the flow and transport models, (4.1)–(4.8), and measurements of contaminant concentration and hydraulic head. Other parameters in the transport model, such as porosity, reaction term coefficients, etc., are assumed to be known. Identification of the source term $q_{\rm s}c_{\rm s}$ is tantamount to finding the location(s), $\mathbf{S}_{\rm l}$, and strength, $\mathbf{S}_{\rm s} \in \mathbb{R}^{N_{\rm re}}$, of the contaminant source; with the elements $S_{{\rm s}j}$ ($j = 1, \ldots, N_{\rm re}$) of the vector $\mathbf{S}_{\rm s}$ denoting the release strength at jth time interval.

Measurements of hydraulic head, $\bar{h}_m = \bar{h}(\mathbf{x}_m)$, and solute concentration, $\bar{c}_{mi} = \bar{c}(\mathbf{x}_m, t_i)$, are collected at locations $\{\mathbf{x}_m\}_{m=1}^M$ at times $\{t_i\}_{i=1}^I$. In lieu of field observations, we generate these data by corrupting the solution of (4.1)–(4.8) obtained for the reference parameter values by random measurement errors ϵ_{mi}^c and ϵ_m^h, such that:

$$\bar{c}_{m,i} = c(\mathbf{x}_m, t_i) + \epsilon_{mi}^c, \qquad \bar{h}_m = h(\mathbf{x}_m) + \epsilon_m^h; \qquad m = 1, \ldots, M, \quad i = 1, \ldots, I, \tag{4.9}$$

where $c(\mathbf{x}_m, t_i)$ and $h(\mathbf{x}_m)$ are the model predictions. The zero-mean Gaussian random variables ϵ_{mi}^c have covariance $\mathbb{E}[\epsilon_{mi}^c \epsilon_{nj}^c] = \delta_{ij} R_{mn}^c$, where $\mathbb{E}[\cdot]$ denotes the ensemble mean; δ_{ij} is the Kronecker delta function; and R_{mn}^c with $m, n \in [1, M]$ are components of the $M \times M$ spatial covariance matrix $\mathbf{R}^c$ of measurements errors. To be specific, we set $\mathbf{R}^c = \sigma_c \mathbf{I}$, where σ_c is the standard deviation of the measurement errors, and $\mathbf{I}$ is the $(M \times M)$ identity matrix. The hydraulic head measurement errors ϵ_m^h are zero-mean Gaussian random variables with covariance $\mathbb{E}[\epsilon_m^h \epsilon_n^h] = R_{mn}^h$ with $m, n \in [1, M]$. We set $\mathbf{R}^h = \sigma_h \mathbf{I}$, where σ_h is the standard deviation of the measurement errors.

The error model in (4.9) assumes the flow and transport models (4.1)–(4.8) to be exact and the measurements errors to be unbiased and uncorrelated in time but not in space. The groundwater flow equation is solved with MODFLOW [47], and the solute transport equation with with MT3DMS [141, 8]. We use `Flopy` [6], a `Python` implementation of these two packages.

4.3 Methodology

Below we describe the three elements of our inversion framework: ensemble smoother with multiple data assimilation (ESMDA), convolutional adversarial autoencoders (CAAE) parameterization of the conductivity field, and a Dense encoder-decoder (DenseED) neural network surrogate of the forward model.

4.3.1 Ensemble Smoother with Multiple Data Assimilation (ESMDA)

Upon a spatiotemporal discretization, the uncertain (random) input parameters in (4.1)–(4.4) are rearranged into a vector $\mathbf{m}$ of length N_m; these inputs include the discretized source term ($\mathbf{S}_{\rm l}$, $\mathbf{S}_{\rm s}$) and hydraulic conductivity $K(\mathbf{x})$. Similarly, we arrange the random measurements $\bar{c}_{m,i}$ and $\bar{h}_m$ into

a vector $\mathbf{d}$ of length $N_d = M(I+1)$, and the random measurement noise ϵ^c_{mi} and ϵ^h_m into a vector $\boldsymbol{\varepsilon}$ of the same length. Then, the error model (4.9) takes the vector form,

$$\mathbf{d} = \mathbf{g}(\mathbf{m}) + \boldsymbol{\varepsilon}, \tag{4.10}$$

where $\mathbf{g}(\cdot)$ is the vector, of length N_d, of the correspondingly arranged stochastic model predictions $c(\mathbf{x}_m, t_i)$ and $h(\mathbf{x}_m)$ predicated on the model inputs $\mathbf{m}$. Let $f_{\mathbf{m}}$ denote a prior probability density function (PDF) of the inputs $\mathbf{m}$, which encapsulates the knowledge about the aquifer's properties and contaminant source before any measurements are assimilated. Our goal is to improve this prior by assimilating the measurements $\mathbf{d}$, i.e., to compute the posterior PDF of the model parameters, $f_{\mathbf{m}|\mathbf{d}}$. As we have done in the previous chapters, this task is accomplished via the Bayes rule,

$$f_{\mathbf{m}|\mathbf{d}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}}) = \frac{f_{\mathbf{m}}(\tilde{\mathbf{m}}) f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}})}{f_{\mathbf{d}}(\tilde{\mathbf{d}})}, \qquad f_{\mathbf{d}}(\tilde{\mathbf{d}}) = \int f_{\mathbf{m}}(\tilde{\mathbf{m}}) f_{\mathbf{d}|\mathbf{m}}(\tilde{\mathbf{m}};\tilde{\mathbf{d}}) \mathrm{d}\tilde{\mathbf{m}}, \tag{4.11}$$

where $f_{\mathbf{d}|\mathbf{m}}$ is the likelihood function, i.e., the joint PDF of the concentration and hydraulic head measurements conditioned on the corresponding model predictions; and $f_{\mathbf{d}}$, is the "evidence" that serves as a normalizing constant so that $f_{\mathbf{m}|\mathbf{d}}(\mathbf{m};\cdot)$ integrates to 1.

To compute (4.11), we use ESMDA [30], which is an ensemble updating method similar to ensemble smoother (ES) [124] or ensemble Kalman filter (EnKF) [31, 32]. To place ESMDA in the proper perspective, we briefly describe ES. The method is initiated by drawing N_{e} samples $\mathbf{M}^f = \{\mathbf{m}^f_1, \ldots, \mathbf{m}^f_{N_{\mathrm{e}}}]$ from the prior PDF $f_{\mathbf{m}}$. These data are then updated as:

$$\mathbf{m}^a_j = \mathbf{m}^f_j + \mathbf{C}^f_{\mathbf{MD}}(\mathbf{C}^f_{\mathbf{DD}} + \mathbf{C}_{\mathbf{D}})^{-1}[\mathbf{d}_{uc,j} - g(\mathbf{m}^f_j)], \qquad j = 1, \ldots, N_{\mathrm{e}}, \tag{4.12}$$

forming $\mathbf{M}^a = [\mathbf{m}^a_1, \ldots, \mathbf{m}^a_{N_{\mathrm{e}}}]$, the updated ensemble conditioned on the measurements $\mathbf{d}$. Here, $\mathbf{C}_{\mathbf{D}} \in \mathbb{R}^{N_d \times N_d}$ is the covariance matrix of the measurement errors ε; $\mathbf{d}_{uc,j} \sim \mathcal{N}(\mathbf{d}, \mathbf{C}_{\mathbf{D}})$ are the perturbed measurements with the measurement error covariance $\mathbf{C}_{\mathbf{D}}$; $\mathbf{C}^f_{\mathbf{DD}} \in \mathbb{R}^{N_d \times N_d}$ is the auto-covariance matrix of the model predictions $\mathbf{D}^f = \mathbf{D}^f = [g(\mathbf{m}^f_1), \ldots, g(\mathbf{m}^f_{N_e})]$; and $\mathbf{C}^f_{\mathbf{MD}} \in \mathbb{R}^{N_m \times N_d}$ is the cross-covariance matrix between $\mathbf{M}^f$ and $\mathbf{D}^f$. During the update, all the data $\mathbf{d}$ are used once, simultaneously. This global update may cause an unacceptably large mismatch between the model response and the measurements, which precipitated the development of an iterative ES with smaller-scale updates.

While ES performs a single large Gauss-Newton correction to the ensemble $\mathbf{M}^f$, ESMDA makes a smaller correction during each update and deploys the inflated covariance matrix $\mathbf{C}_{\mathbf{D}}$ to damp the changes in the ensemble at early iterations [35, 131]. (In the linear Gaussian case, ESMDA and ES yield identical results.) We use the following algorithm to implement ESMDA.

- Set the number of data assimilation iterations, N_a, and the corresponding inflation coefficients α_i $i = 1, \ldots, N_a$. Requiring $\sum_{i=1}^{N_a} \alpha_i = 1$ guarantees the consistency with ES in the linear

Gaussian case. Generate the initial ensemble $\mathbf{m}_j^1$ $(j = 1, \ldots, N_\mathrm{e})$ from the prior PDF $f_\mathbf{m}$.

- Repeat the following steps for $i = 1, \ldots, N_a$:
  1. Run the forward simulation for each member $\mathbf{m}_j^i$ $(j = 1, \ldots, N_\mathrm{e})$, from the parameter ensemble $\mathbf{M}^f$, to obtain the corresponding model predictions (and, in the synthetic case, observations) $g(\mathbf{m}_j^i)$.
  2. Perturb the measurements with inflated measurement noise: $\mathbf{d}_{uc,j}^i \sim \mathcal{N}(\mathbf{d}, \alpha_i \mathbf{C_D})$.
  3. Compute the cross covariance matrix $\mathbf{C}_\mathbf{MD}^i$ and the auto-covariance matrix of the predicted data $\mathbf{C}_\mathbf{DD}^i$.
  4. Update the ensemble as in (4.12), but with $\mathbf{C_D}$ replaced by $\alpha_i \mathbf{C_D}$:

$$\mathbf{m}_j^{i+1} = \mathbf{m}_j^i + \mathbf{C}_\mathbf{MD}^i (\mathbf{C}_\mathbf{DD}^i + \alpha_i \mathbf{C_D})^{-1} [\mathbf{d}_{uc,j}^i - g(\mathbf{m}_j^i)], \qquad j = 1, \ldots, N_\mathrm{e}. \tag{4.13}$$

The inverse, $\mathbf{C}_i^{-1}$, of the matrix $\mathbf{C}_i = \mathbf{C}_\mathbf{DD}^i + \alpha_i \mathbf{C_D}$ is approximated by its pseudo-inverse with truncated singular value decomposition (TSVD).

4.3.2 CAAE Parameterization of Conductivity Field

Let the matrix $\mathbf{k} \in \mathbb{R}^{W \times H \times D}$ denote the log-conductivity field $\ln K(\mathbf{x})$ defined on a three-dimensional numerical grid, which consists of W, H and D elements in the three spatial directions. We use CAAE to parameterize the high-dimensional $\mathbf{k}$ with a low-dimensional latent variable $\mathbf{z}$. CAAE consists of two components, a GAN and an AE.

GAN [40] is a DNN strategy for generating data from complex distributions without having to actually acquire the full PDF. This strategy comprises two networks: a generator $\mathcal{G}(\cdot)$ that generates samples similar to $\mathbf{k}$; and a discriminator $\mathcal{D}(\cdot)$ that is trained to distinguish between the generated samples and the real data samples. By "playing an adversarial game", the discriminator $\mathcal{D}(\cdot)$ improves its ability to catch flaws in the generated samples, and the generator $\mathcal{G}(\cdot)$ improves its capacity to generate realistic samples that try to trick the discriminator.

AE learns a low-dimensional representation $\mathbf{z}$ of the data $\mathbf{k}$, and then generates a reconstruction $\hat{\mathbf{k}}$ from $\mathbf{z}$ that closely matches the original data $\mathbf{k}$. The encoded latent variable $\mathbf{z}$ is constructed to follow a PDF $f_\mathbf{z}(\tilde{\mathbf{z}})$ that is easy to sample from, e.g., a standard normal PDF $\mathcal{N}(\mathbf{0}, \mathbf{I})$. A variational autoencoders (VAE) [61] forces the empirical PDF of $\mathbf{z}$ computed from the samples of $\mathbf{k}$, $f_{\mathbf{z}|\mathbf{k}}(\tilde{\mathbf{z}})$, to be close to the target PDF $f_\mathbf{z}(\tilde{\mathbf{z}})$ by adding the Kullback-Leibler divergence $\mathrm{KL}[f_{\mathbf{z}|\mathbf{k}} \| f_\mathbf{z}]$ between the empirical and target PDFs to the total loss function:

$$\mathcal{L}_\mathrm{VAE} = \mathcal{L}_\mathrm{rec}(\mathbf{k}, \hat{\mathbf{k}}) + \mathrm{KL}\big[f_{\mathbf{z}|\mathbf{k}} \| f_\mathbf{z}\big], \tag{4.14}$$

where $\mathcal{L}_\mathrm{rec}(\mathbf{k}, \hat{\mathbf{k}})$ is the discrepancy between the data $\mathbf{k}$ and their reconstruction $\hat{\mathbf{k}}$, choices of this discrepancy function include L_1 or L_2 norm. We use the former to define the average, the average

reconstruction error $\mathcal{L}_{\text{rec}}$ over N training samples,

$$\mathcal{L}_{\text{rec}} = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{k}_i - \hat{\mathbf{k}}_i||_1, \tag{4.15}$$

CAAE differs from VAE in the way it minimizes the discrepancy between the empirical PDF $f_{\mathbf{z}|\mathbf{k}}(\tilde{\mathbf{z}})$ and the target PDF $f_{\mathbf{z}}(\tilde{\mathbf{z}})$ of the latent random variable $\mathbf{z}$. Instead of minimizing the KL divergence $\text{KL}[f_{\mathbf{z}|\mathbf{k}} \| f_{\mathbf{z}}]$, adversarial autoencoders (AAE) employs an adversarial training procedure to minimize this discrepancy. The training of the encoder $\mathcal{G}(\cdot)$, decoder, and the discriminator $\mathcal{D}(\cdot)$ is divided into the reconstruction phase and the regularization phase [90]. Parameters in the encoder and decoder are updated by minimizing the loss function:

$$\mathcal{L}_{\text{ED}} = \mathcal{L}_{\text{rec}} + w\mathcal{L}_{\mathcal{G}}. \tag{4.16}$$

We use $\mathcal{L}_{\mathcal{G}}$ to quantify the decoder's ability to trick the discriminator,

$$\mathcal{L}_{\mathcal{G}} = -\frac{1}{N}\sum_{i=1}^{N} \ln\big\{\mathcal{D}[\mathcal{G}(\mathbf{k}_i)]\big\}. \tag{4.17}$$

The weight factor w in (4.16) is used to assign relative importance to these two sources of error. In the simulations reported below, we set $w = 0.01$.

After the encoder and decoder are updated in the first training phase, the discriminator $\mathcal{D}(\cdot)$ is trained in the second phase to minimize the loss function:

$$\mathcal{L}_{\mathcal{D}} = -\frac{1}{N}\sum_{i=1}^{N} \Big\{ \ln\big[\mathcal{D}(\mathbf{z}_i)\big] + \ln\big[1 - \mathcal{D}[\mathcal{G}(\mathbf{x}_i)\big] \Big\}. \tag{4.18}$$

By iterating between these two training phases, one obtains the mappings from $\mathbf{k}$ to $\mathbf{z}$ and from $\mathbf{z}$ to $\hat{\mathbf{k}}$, and the decoder reaches its goal of constructing realizations $\hat{\mathbf{k}}$ similar to the data $\mathbf{k}$.

4.3.3 DenseED Neural Networks as Forward Model Surrogates

ESMDA inversion requires a large number of forward solves of the PDE-based model (4.1)–(4.8) for multiple realizations of the parameters $\mathbf{m}$. To alleviate the cost of each forward run, we replace the PDE-based model with its CNN surrogate.

Several approaches to constructing an input-output surrogate are collated in Table 4.2. We choose an autoregressive image-to-image (i-to-i) regression model, rather than its image-to-sparse-observation counterpart, because of its superior generalizability [143]. We choose an autoregressive i-to-i model over a one-to-many i-to-i model based on computer-memory considerations: for three-dimensional problems with I time steps, memory allocated for input and output can be prohibitively

large; also, the autoregressive scheme reduces the number of DNN parameters needed for the regression task. The source location ($\mathbf{S}_{\mathrm{l},t}$) and strength ($\mathbf{S}_{\mathrm{s},t}$) for the release period $[t, t+\Delta t]$ are assembled into a three-dimensional matrix $\mathbf{S}(\mathbf{x}, t) \in \mathbb{R}^{D\times H\times W}$.

Model	Input	Output	Modeling frequency
PDE model	$\mathbf{m}$	$c(\mathbf{x}, t_i)$, $h(\mathbf{x})$	1
Image-to-image	$\mathbf{m}$	$c(\mathbf{x}, t_i)$, $h(\mathbf{x})$	1
Image-to-sensors	$\mathbf{m}$	$c(\mathbf{x}_m, t_i)$, $h(\mathbf{x}_m)$	1
Autoregressive i-to-i	$c(\mathbf{x}, t), \ln K(\mathbf{x}), S(\mathbf{x}, t)$	$c(\mathbf{x}, t+\Delta t)$, $h(\mathbf{x})$	I

Table 4.2: Alternative input-output frameworks for construction of a surrogate model. The data are collected at M locations $\mathbf{x}_m$ $(m = 1, \cdots, M)$ at I times t_i $(i = 1, \cdots, I)$. The source location ($\mathbf{S}_{\mathrm{l},t}$) and strength ($\mathbf{S}_{\mathrm{s},t}$) for the release period $[t, t+\Delta t]$ are assembled into a three-dimensional matrix $\mathbf{S}(\mathbf{x}, t)$.

An autoregressive surrogate $\mathbf{NN}_{\mathrm{auto}}$ replaces the PDF-based model:

$$\mathbf{g} : \mathbf{m} \xrightarrow{\text{PDEs}} \{c(x_m, t_i), h(x_m)\}_{m,i=1}^{M,I} \tag{4.19}$$

with a CNN that sequentially (I times) predicts the system state at the next time step,

$$\mathbf{NN}_{\mathrm{auto}} : c(\mathbf{x}, t_i), K(\mathbf{x}), S(\mathbf{x}, t_i) \xrightarrow{\text{CNN}} \{c(\mathbf{x}, t_{i+1}), h(\mathbf{x})\}, \qquad i = 0, \ldots, I-1. \tag{4.20}$$

If the three-dimensional simulation domain is discretized with a $D \times H \times W$ grid, then the autoregressive CNN surrogate $\mathbf{NN}_{\mathrm{auto}}$ performs the following input-to-output mapping:

$$\mathbf{NN}_{\mathrm{auto}} : \mathbb{R}^{n_x\times D\times H\times W} \to \mathbb{R}^{n_y\times D\times H\times W}, \tag{4.21}$$

where $n_x = 3$, denotes the three channels representing the concentration $c(\mathbf{x}, t_i)$ and source terms $S(\mathbf{x}, t_i)$ at time t_i, and the log-conductivity $\ln K(\mathbf{x})$; and $n_y = 2$ designates the two output channels representing the concentration $c(\mathbf{x}, t_{i+1})$ at time t_{i+1} the hydraulic head $h(\mathbf{x})$. A representative input-to-output example is shown in Figure 4.1.

We use a three-dimensional DenseED architecture to solve the image-to-image regression task with a coarsen-refine process, with the convolutional operations. The encoder extracts the high-level coarse features of the input maps, while the decoder subsequently refines the coarse features to the full maps [94, Fig. 2]. We use the L_1-norm loss function, the L_2-norm weight regularization, and stochastic gradient descent [15] in the CNN training process. A detailed description of this surrogate model and its training procedure can be found in [94]. We have slightly extended their procedure by adding the measurement locations to the loss function. This allows us to penalize the prediction error at these specific locations.

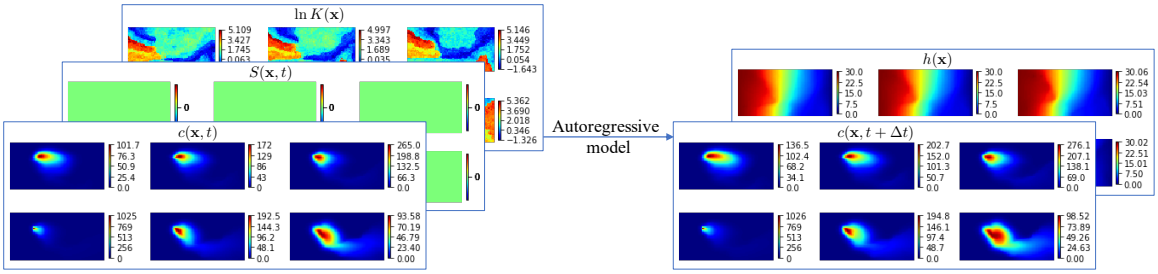

Figure 4.1: Autoregressive surrogate $\mathbf{NN}_{\text{auto}}$ of the PDE-based flow and transport model (4.1)–(4.8). Each image depicts the flattened three-dimensional field, from the first layer on the top to the bottom layer. The three input channels (left) correspond to $c(\mathbf{x}, t)$, $S(\mathbf{x}, t)$, and $\ln K(\mathbf{x})$. The two output channels (right) correspond to $c(\mathbf{x}, t + \Delta t)$ and $h(\mathbf{x})$.

4.3.4 CAAE–DenseED–ESMDA Inversion Framework

We combine the CAAE parameterization of the conductivity field with the DenseED CNN surrogate of the forward model to obtain fast and accurate predictions of concentration $c(\mathbf{x}, t)$ and $h(\mathbf{x})$ for a given set of inputs. Then, we utilize ESMDA to identify the unknown parameters, including the conductivity field and the source terms $(\mathbf{S}_{\text{l}}, \mathbf{S}_{\text{s}})$. The CAAE parameterization enables one to estimate the discretized log-conductivity field $\mathbf{k}$ through the latent variable $\mathbf{z}$. Our CAAE–DenseED–ESMDA inversion framework is implemented in the following algorithm.

1. Train a CAAE; obtain the decoder $\mathcal{D}$ that maps the low-dimensional latent variable $\mathbf{z}$ back onto the log-conductivity field $\mathbf{k}$.

2. Train an autoregressive DenseED CNN $\mathbf{NN}_{\text{auto}}$ to predict $c(\mathbf{x}, t)$ and $h(\mathbf{x})$ for a given conductivity field and contaminant release history.

3. Generate the initial input ensemble $\mathbf{M}^f$ of size N_{e}, whose elements $\mathbf{m}_j^1$ $(j = 1, \ldots, N_{\text{e}})$ are defined as $\mathbf{m}_j^1 = (\mathbf{z}_j^1, \mathbf{S}_{\mathbf{l}j}^1, \mathbf{S}_{\mathbf{s}j}^1)^\top$. Here, $\mathbf{z}_j^1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{\mathbf{z}})$ is the latent variable for the log conductivity field; and $\mathbf{S}_{\mathbf{l}j}^1$ and $\mathbf{S}_{\mathbf{s}j}^1$ denote respectively the source location and strength in all release periods, drawn from an appropriate prior distribution.

4. Perform the ESMDA inversion with N_a data assimilation iterations and the inflation coefficients α_i $(i = 1, \ldots, N_a)$. For $i = 1, \ldots, N_a$,

   (a) Obtain the log-conductivity realizations $\mathbf{k}_j^i = \mathcal{D}(\mathbf{z}_j^i)$ with $j = 1, \ldots, N_{\text{e}}$;

   (b) Form the release configuration $\{\mathbf{S}_{\text{l},j}^i, \mathbf{S}_{\text{s},j}^i\}$ into the input matrix $\mathbf{S}_j^i$, and predict $c(\mathbf{x}, t)$ and $h(\mathbf{x})$ at the measurement times and locations, $\mathbf{NN}_{\text{auto}}(\mathbf{m}_j^i)$ for all j;

   (c) Update the ensemble $\mathbf{m}_j^i$ via ESMDA with α_i to obtain $\mathbf{m}_j^{i+1}$.

5. The end result, $\mathbf{m}_j^{N_a+1}$, serves as the final ensemble from which PDFs of the log conductivity field and the contaminant release parameters are estimated.

4.4 Numerical experiments

4.4.1 Experimental Setup

A confined heterogeneous aquifer is described as a rectangular cuboid Ω of size 2500 m $\times$ 1250 m $\times$ 300 m; it is discretized with a mesh consisting of $81 \times 41 \times 6$ cells. Groundwater flow is driven by constant heads $h_{\mathrm{L}} = 30$ m and $h_{\mathrm{R}} = 0$ m imposed along the left ($x_1 = 0$) and right ($x_1 = 2500$ m) facets of the cuboid, respectively; the remaining boundaries are impermeable to flow. Hydraulic conductivity of this aquifer, $K(\mathbf{x})$, is unknown (except when generating the ground truth); equiprobable realizations of $Y(\mathbf{x}) = \ln K(\mathbf{x})$ are generated by extracting $81 \times 41 \times 6$ patches from the 150 px $\times$ 180 px $\times$ 105 px training image [91] in Figure 4.2, available at `https://github.com/GAIA-UNIL/trainingimages`. One such cropped log-conductivity field $Y(\mathbf{x})$ and the corresponding hydraulic head $h(\mathbf{x})$, obtained as a solution of the groundwater flow equation (4.1), are shown in Figure 4.3. These fields serve as ground truth.

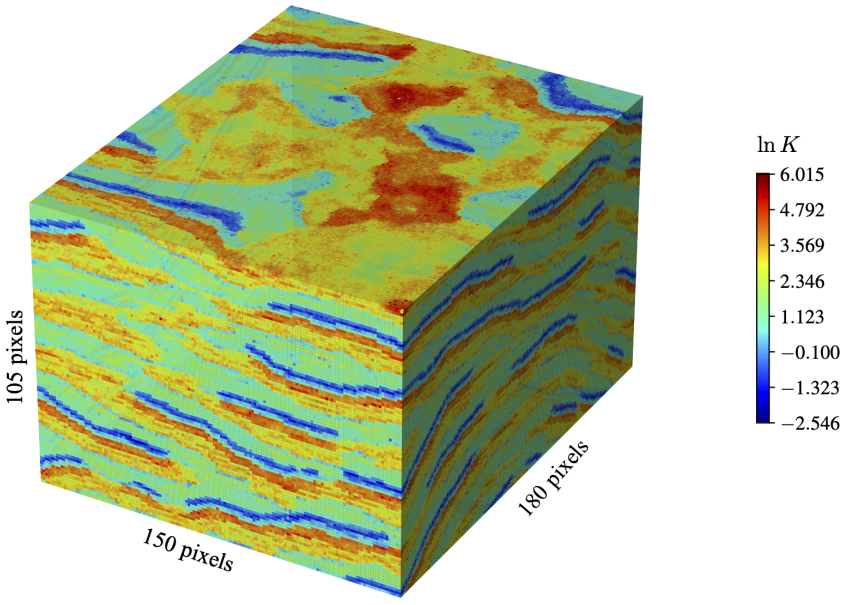

Figure 4.2: Training image, consisting of $150 \times 180 \times 105$ pixels. Equiprobable realizations of log-conductivity $Y(\mathbf{x}) = \ln K(\mathbf{x})$ are generated by randomly selecting patches of size $81 \times 41 \times 6$ pixels. Conductivity K is in m/d.

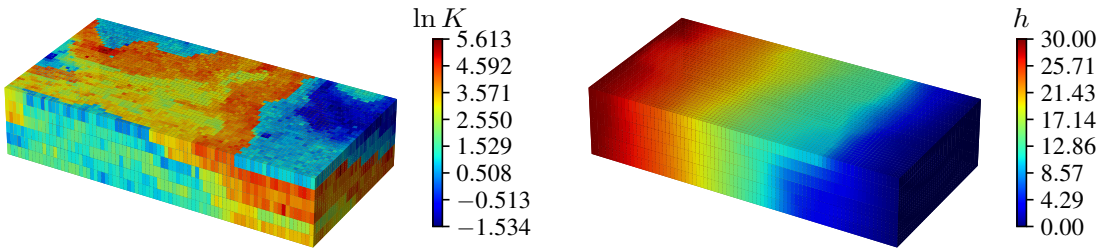

Figure 4.3: Log-conductivity $Y(\mathbf{x})$ (left) and the corresponding hydraulic head $h(\mathbf{x})$ (right), which serve as ground truth and to generate measurements of h at observation wells. Conductivity K is in m/d and head h in m.

Porosity θ and bulk density ρ of the soil; dispersivities α_L, α_T and α_C; and the parameters K_f and a of the Freundlich isotherm are constant. Values of these transport parameters, which are representative of a sandy alluvial aquifer in Southern California [80], are presented in Table 4.3. The contaminant enters the aquifer via a point source, whose depth is known (the fourth layer from the top of the domain) but the location in the horizontal plane (S_{l}^x and S_{l}^y) is uncertain. The contaminant release is known to occur during a 20-year period, but its strength is uncertain. Following the standard practice in groundwater modeling, we divide this time interval into $N_{\mathrm{re}} = 5$ sub-intervals ("stress periods" in the MODFLOW/MT3D language) during each of which the release strength (S_{s}) is constant. In this configuration, the unknown contaminant release history is represented by the vector $\mathbf{S} = (\mathbf{S}_{\mathrm{l}}, \mathbf{S}_{\mathrm{s}})$, where $\mathbf{S}_{\mathrm{l}} = (S_{\mathrm{l}}^x, S_{\mathrm{l}}^y)^\top$ and $\mathbf{S}_{\mathrm{s}} = (S_{\mathrm{s},1}, S_{\mathrm{s},2}, S_{\mathrm{s},3}, S_{\mathrm{s},4}, S_{\mathrm{s},5})^\top$. The values of $\mathbf{S}$ used to generate the ground-truth concentrations are reported in Table 4.4. Combined with the discretized version $\mathbf{k}$ of the uncertain log-conductivity field $Y(\mathbf{x})$, this yields 19933 unknowns to be determined from the measurements of solute concentration $c(\mathbf{x}, t)$ and hydraulic head $h(\mathbf{x})$. Expert knowledge about possible location and strength of the contaminant release is encapsulated in the uniform ("uninformative") prior distributions for $\mathbf{S}_{\mathrm{l}}$ and $\mathbf{S}_{\mathrm{s}}$), which are shown in Table 4.4.

These measurements are collected at observations wells, whose completion allows one to collect water samples either in each of the model's six vertical layers or only in one layer (the second, the forth, or the sixth). We consider two alternative networks of observation wells, whose locations are depicted in Figure 4.4. During the simulated time horizon of 40 years, the contaminant concentration is sampled at $I = 10$ time intervals of four years each, and the hydraulic head is measured once since the flow is at steady-state. The data at all space-time locations are generated by adding 2% measurement error, $\epsilon \sim \mathcal{N}\{\mathbf{0}, (0.02\mathbf{g}(\mathbf{m}))^2\}$, to the solution $\mathbf{g}(\mathbf{m})$ of the flow and transport

Property	Value	Units
ϕ	0.3	–
K_f	0.1	$(\mathrm{m}^3/\mathrm{g})^a$
a	0.9	–
ρ	1587	kg/m^3
α_L	35	m
α_T/α_L	0.3	–
α_C/α_L	0.3	–
D_m	10^{-9}	m^2/d

Table 4.3: Values of the transport parameters for a dissolved contaminant migrating in a generic sandy alluvial aquifer in Southern California [80].

	S_l^x	S_l^y	$S_{\mathrm{s},1}$	$S_{\mathrm{s},2}$	$S_{\mathrm{s},3}$	$S_{\mathrm{s},4}$	$S_{\mathrm{s},5}$
Truth	291	625	224	174	869	201	741
Prior	$[125, 625]$	$[125, 1125]$	$[50, 1000]$	$[50, 1000]$	$[50, 1000]$	$[50, 1000]$	$[50, 1000]$

Table 4.4: Parameters $\mathbf{S}_\mathrm{l} = (S_\mathrm{l}^x, S_\mathrm{l}^y)^\top$ and $\mathbf{S}_\mathrm{s} = (S_{\mathrm{s},1}, S_{\mathrm{s},2}, S_{\mathrm{s},3}, S_{\mathrm{s},4}, S_{\mathrm{s},5})^\top$ used to represent, respectively, the location and strength of a contaminant release. Reported as "Truth" are their (unknown) reference values used to generate ground truth and concentration measurements, and "Prior" the intervals on which their uniform priors, $\mathcal{U}[\cdot, \cdot]$, are defined. The values of $\mathbf{S}_\mathrm{l}$ are in m, and of $\mathbf{S}_\mathrm{s}$ in g/m^3.

model (4.1)–(4.8) with the input parameter values identified as "ground truth" above.

4.4.2 CAAE Training for Conductivity Parameterization

We train a CAAE DNN to parameterize the discretized log conductivity field $\mathbf{k} \in \mathbb{R}^{81\times41\times6}$. The end goal is an encoder $\mathcal{G}(\mathbf{k})$ that maps an input field $\mathbf{k}$ onto a low dimensional latent variable $\mathbf{z} \in \mathbb{R}^{2\times2\times11\times21}$ with standard-Gaussian prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and a decoder $\mathrm{De}(\mathbf{z})$ that reconstructs $\mathbf{k}$ from this latent variable. The training is done on 23000 realizations of $\mathbf{k}$, obtained as randomly selected $(81 \times 41 \times 6)$ patches from the large training image in Figure 4.2. Additional 4000 images cropped from this image serve as the testing set. The architecture of the CAAE DNN is similar to that in [95], except that the latent variable $\mathbf{z}$ has $2 \cdot 2 \cdot 11 \cdot 21 = 924$ elements. With 50-epochs training and the learning rate of $2 \cdot 10^{-4}$, the Adam optimizer is used to obtain the DNN parameters and, thus, build $\mathcal{G}(\mathbf{k})$ and $\mathrm{De}(\mathbf{z})$.

A representative realization of $\ln K(\mathbf{x}) \longmapsto \mathbf{k}$ from the test set and its reconstruction via decoder, $\hat{\mathbf{k}} = \mathrm{De}(\mathbf{z})$, are shown in Figure 4.5. After the training is complete, the mean absolute error $\|\mathbf{k}-\hat{\mathbf{k}}\|_1$, averaged over all the elements of the numerical mesh and over the 4000 members of the testing data set, is 0.228464. The reconstructed log-conductivity field $\hat{\mathbf{k}}$ captures the main structural features of its original counterpart $\mathbf{k}$. Some loss of information is unavoidable in reduced-order modeling but, overall, the performance of this autoencoders is adequate to achieve accurate inverse modeling results, as we shown in section 4.4.4 below.

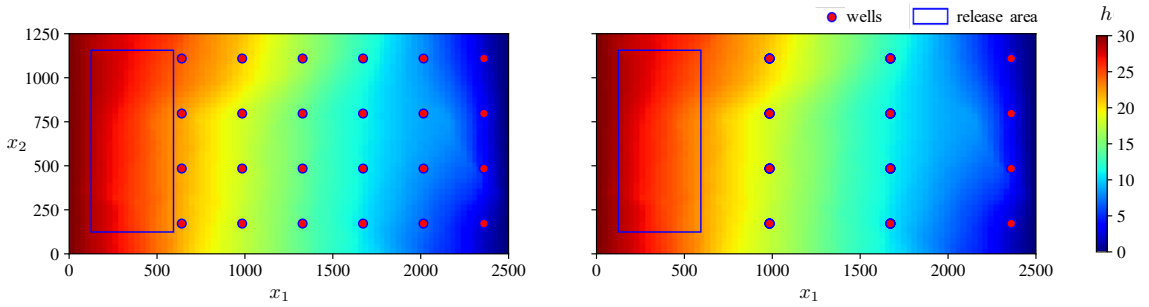

Figure 4.4: Two alternative networks of observational wells (red dots) in which measurements of hydraulic head h and solute concentration c are collected. The well locations are superposed on the ground-truth distribution of hydraulic head in the fourth layer of the MODFLOW model. The blue box represents a region of possible contaminant release from a point source that is known to be located in the fourth model-layer, x_1, x_2 in m.

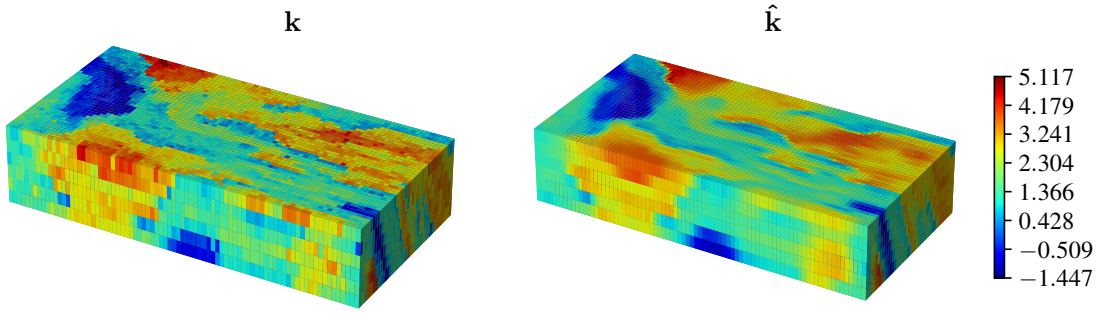

Figure 4.5: A representative realization of $\ln K(\mathbf{x}) \longmapsto \mathbf{k}$ from the test set (left) and its reconstruction (right) via the CAAE encoder, $\mathbf{z} = \mathcal{G}(\mathbf{k})$, and decoder, $\hat{\mathbf{k}} = \mathrm{De}(\mathbf{z})$.

4.4.3 DenseED Surrogate Model

As mentioned in section 4.3, although only model predictions at the well locations are necessary for the inversion, a DNN that predicts $c(\mathbf{x}, t_i)$ and $h(\mathbf{x})$ at all points $\mathbf{x}$ of the simulation domain has better generalization properties. We train our CNN on $N = 800$ Monte Carlo realizations of the PDE-based model (4.1)–(4.8) with corresponding realizations of the input parameters $\mathbf{m}$ (the discretized log-conductivity $\mathbf{k}$ and contaminant release history $\mathbf{S}$). Another set of $N_{\text{test}} = 150$ realizations are retained for testing. These 950 realizations form 950×10 autoregressive input-output pairs. The CNN contains three dense blocks with $N_l = 3$, 6, and 3 internal layers, has the growth rate of $R_g = 48$ and $N_{\text{in}} = 48$ initial features; it was trained for 200 epochs. We use the L_1-norm loss function and the L_2-norm weight regularization, apply stochastic gradient descent [15] in the parameter estimation process, and add 5 times the L_1-norm loss at the well locations to the total loss to penalize the prediction error at the observation wells. The CNN's output is the hydraulic head $h(\mathbf{x})$ and the solute concentration $c(\mathbf{x}, t_i)$ at the next time step t_i.

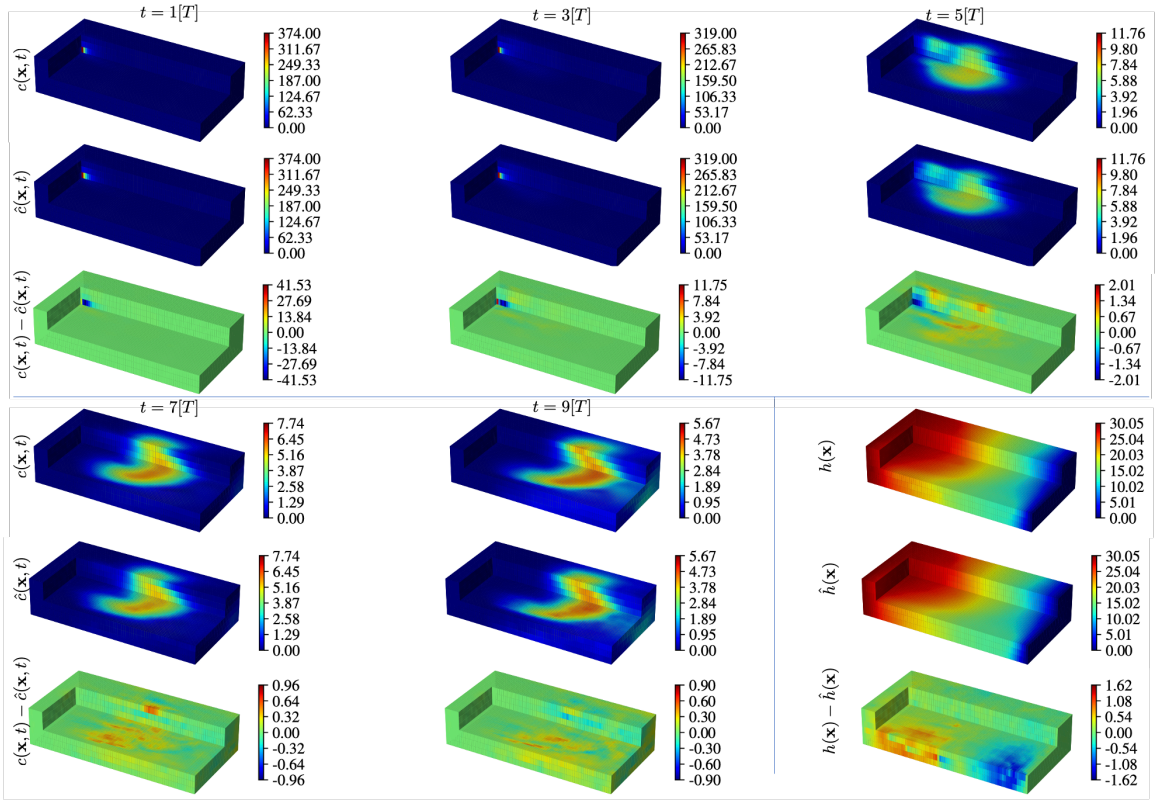

Figure 4.6: Predictions of the solute concentration obtained with the PDE-based model, $c(\mathbf{x}, t)$, and its DenseED CNN surrogate, $\hat{c}(\mathbf{x}, t)$, times $t =$ ... y, $t =$ ... y, $t =$ ... y, $t =$ ... y, and $t =$ ... y. Also shown are the corresponding predictions of the hydraulic head, $h(\mathbf{x})$ and $\hat{h}(\mathbf{x})$; and the difference between these two types $c(\mathbf{x}, t) - \hat{c}(\mathbf{x}, t), h(\mathbf{x}) - \hat{h}(\mathbf{x})$ of prediction.

Figure 4.6 exhibits temporal snapshots of the solute concentrations alternatively predicted with the PDE-based model, $c(\mathbf{x}, t_i)$, and the CNN surrogate, $\hat{c}(\mathbf{x}, t_i)$, for a given realization of the log-conductivity field and the contaminant release configuration (both drawn from the test set). Also presented are the hydraulic head maps predicted by the autoregressive model, $\hat{h}(\mathbf{x})$, and the PDF-based model, $h(\mathbf{x})$. The accuracy of our CNN surrogate is quantified by the total root mean square error, $(\|c(\mathbf{x}, t) - \hat{c}(\mathbf{x}, t)\|_2 + \|h(\mathbf{x}) - \hat{h}(\mathbf{x})\|_2)/2$. It falls to 0.853 at the end of the training process. It is worthwhile emphasizing here that the $N_{\text{NMC}} = 800$ Monte Carlo realizations used to train the CNN surrogate are but a small fraction of the forward runs required by ESMDA inversion framework. One could achieve more accurate predictions for three-dimensional problems by either deploying a more complex DNN architecture [127, 95] or using much larger N_{NMC} or both. However, similar to the CAAE training, we focus on the development of efficient methodologies for three-dimensional inverse modeling that accommodate the trade-off between the accuracy and computational feasibility.

4.4.4 ESMDA Inversion

We demonstrate the use of the CAAE parameterization and the DenseED CNN surrogate of the PDE-based forward model to accelerate the ESMDA inversion. The combination of these three techniques constitutes our CAAE-DenseED-ESMDA framework to approximate the joint posterior PDF of the uncertain model parameters $\mathbf{m}$ consistent with both model predictions and field observations. In the simulations reported below, we select $N_a = 10$ inflation factors in (4.13) and set their values to $\alpha_i = 10$ for $i = 1, \ldots, N_a$, and perform ESMDA with 10 iterations. To ascertain the impact of the the DenseED CNN surrogate on the inversion accuracy, we also run CAAE-ESMDA with the PDE-based forward model implemented in MODFLOW and MT3DMS.

Scenario	Inversion framework	Number of wells	Well completion
1	CAAE–ESMDA	24	all 6 layers
2	CAAE–DenseED–ESMDA	24	all 6 layers
3	CAAE–DenseED–ESMDA	9	all 6 layers
4	CAAE–DenseED–ESMDA	24	layers 2, 4, and 6
5	CAAE–DenseED–ESMDA	9	layers 2, 4, and 6

Table 4.5: Five scenarios of the inverse modeling. The well locations in the dense and sparse observational networks are shown in Figure 4.4. These wells are completed in either all six layers of the model or in three layers only.

In total, five alternative implementations of our inversion algorithm are considered. Summarized in Table 4.5, these scenarios cover the reliance on either the PDE- or CNN-based forward model, and on the data provided by either the dense or sparse network of observational wells in Figure 4.4. The dense network in Scenarios 1 and 2 consists of 24 wells that are completed in all 6 layers of the model, yielding $24 \cdot 6 = 144$ measurements of the solute concentration and hydraulic head at each observation time. The dense network in Scenario 4 refers to the same wells but completed in layers

2, 4, and 6 only, yielding $24 \cdot 3 = 72$ measurements at each observation time. The sparse networks in Scenarios 3 and 5 comprise 12 observation wells that are completed either in all layers or in three layers, resulting in either $12 \cdot 6 = 72$ or $12 \cdot 3 = 36$ measurements at each observation time.

In all five scenarios, the ensemble size for ESMDA is set to $N_\mathrm{e} = 960$. Figure 4.7 shows this number to sufficient for the algorithm's convergence. It exhibits the averaged ensemble observation error

$$\mathcal{E}_\mathrm{obs} = \left\| \left(\frac{1}{N_\mathrm{e}} \sum_{j=1}^{N_\mathrm{e}} \mathbf{g}(\mathbf{m}_j) \right) - \mathbf{d} \right\|_2$$

that is plotted as function of the number of iteration of the ESMDA algorithm. Here, $\mathbf{g}(\mathbf{m}_j)$ denotes the forward model prediction, for the input parameters $\mathbf{m}_j$, at the same space-time locations as the measurements $\mathbf{d}$.

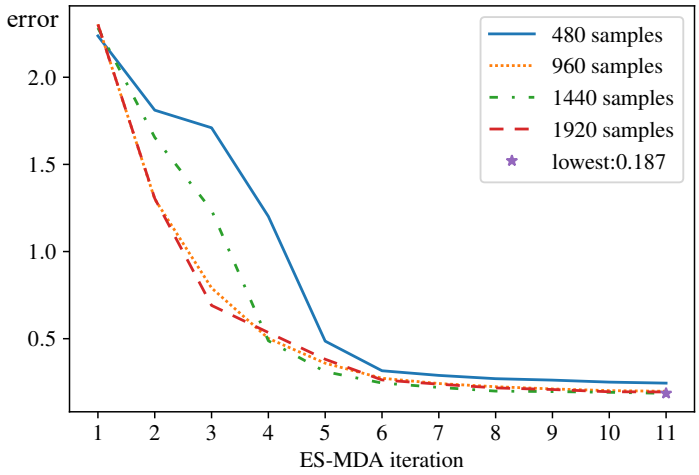

Figure 4.7: Averaged ensemble observation error, $\mathcal{E}_\mathrm{obs}$, in Scenario 2, plotted as function of the number of iteration of the ESMDA algorithm for several ensemble sizes, N_e. As a result, $N_\mathrm{e} = 960$ is chosen to be the ensemble size for all five scenarios in Table 4.5.

Dense Observation Network (Scenarios 1 and 2)

Figure 4.8 exhibits posterior statistics (mean $\langle Y \rangle$ and standard deviation σ_Y) of the log-conductivity $Y(\mathbf{x})$, obtained after the assimilation of all 144 measurements via either CAEE-ESMDA (Scenario 1) or CAEE-DenseED-ESMDA (Scenario 2). In both scenarios, the posterior ensemble mean $\langle Y \rangle$, reconstructed from the latent variable $\mathbf{z}$, correctly identifies the low-conductivity region in the right top region of the three-dimensional domain and the high-conductivity regions elsewhere. As expected, the mean log-conductivity fields, $\langle Y \rangle$, are smoother than the reference field Y (Figure 4.3), but

the realizations drawn from the posterior Gaussian PDF $\mathcal{N}(\langle Y \rangle, \sigma_Y)$ exhibit more realistic features (right column in Figure 4.8). Regardless of the forward model used (the only difference in Scenarios 1 and 2), our data assimilation framework yields consistent predictions of σ_Y (middle column in Figure 4.8). It is small throughout most of the domain, indicating the reduced uncertainty in the estimation of hydraulic conductivity $K(\mathbf{x})$ due to assimilation of the concentration and head measurements. The maximum values of σ_Y and, hence, the largest predictive uncertainty in the $K(\mathbf{x})$ estimation, are along the interface between the high- and low-conductivity regions. This finding suggests that the model predictions of hydraulic head and solute concentration are least sensitive to the changes in hydraulic conductivity in that domain; it reaffirms the conclusion of the sensitivity analysis of the relative importance of uncertainties in the spatial arrangement of hydrofacies and their hydraulic conductivities [130].

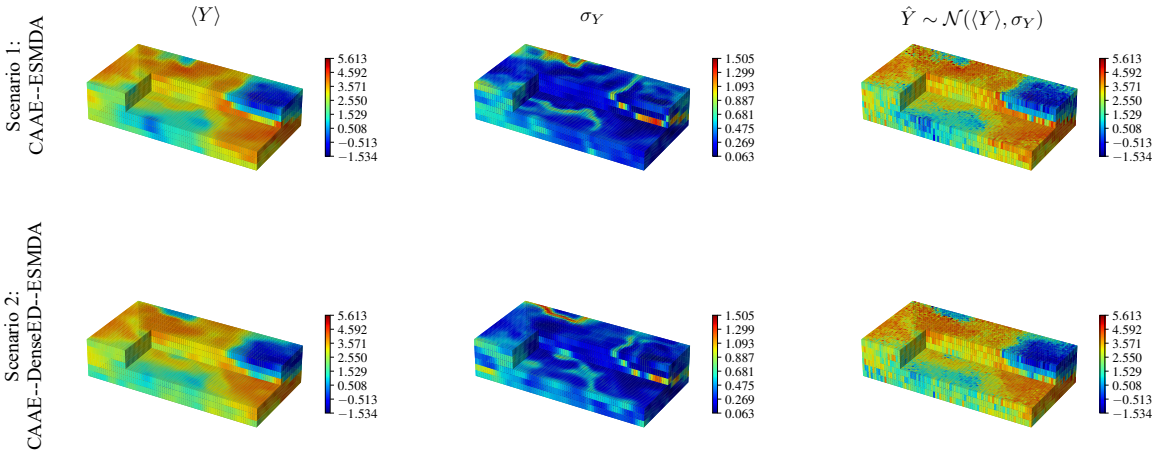

Figure 4.8: Posterior mean ($\langle Y \rangle$, left column) and standard deviation (σ_Y, middle column) of the log-conductivity field $Y(\mathbf{x})$ obtained upon assimilation of concentration and head measurements from the dense observation network. These statistics are obtained via our inversion algorithm CAAE-ESMDA that relies on either the PDE-based forward model (Scenario 1, top row) or its DenseED CNN surrogate (Scenario 2, bottom row). Also shown are representative realizations of $Y(\mathbf{x})$ drawn from the resulting posterior PDF $\mathcal{N}(\langle Y \rangle, \sigma_Y)$ (right column).

The same inversion experiments yield estimates of the contaminant release history $\mathbf{S}$, which are shown in Figures 4.9 and 4.10 for Scenarios 1 and 2, respectively. Regardless of the forward model used, our inversion algorithm accurately estimates the release strength during stress periods 1, 2, and 4 ($S_{\mathrm{s},1}$, $S_{\mathrm{s},2}$, and $S_{\mathrm{s},4}$); the estimates are close to their reference values and have tight 95% confidence intervals. At the same time, the estimates of the source strength during stress periods 3 and 5 ($S_{\mathrm{s},3}$ and $S_{\mathrm{s},5}$) fail to converge to their reference values and exhibit large error bars. The two assimilation strategies yield nearly identical estimates of the contaminant release location, $\mathbf{S}_{\mathrm{l}} = (S_{\mathrm{l}}^x, S_{\mathrm{l}}^y)^\top$; the estimates of both quantities have tight confidence intervals, but the reference value of S_{l}^x lies slightly outside the confidence interval of its estimator.

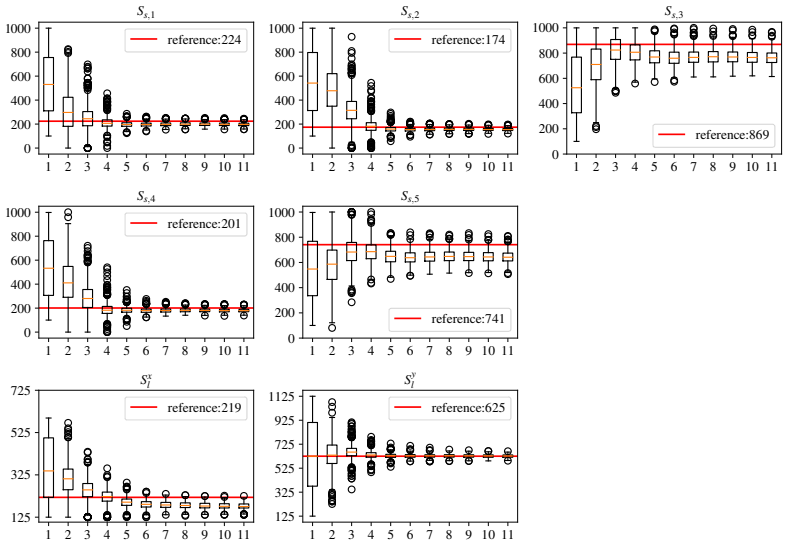

Figure 4.9: Boxplots of the ensembles for the contaminant release terms, $\mathbf{S} = (\mathbf{S}_\mathrm{l}, \mathbf{S}_\mathrm{s})$ with $\mathbf{S}_\mathrm{l} = (S_\mathrm{l}^x, S_\mathrm{l}^y)^\top$ and $\mathbf{S}_\mathrm{s} = (S_{\mathrm{s},1}, \ldots, S_{\mathrm{s},5})^\top$, and their confidence intervals, obtained via the CAAE-ESMDA inversion with the PDE-based forward model (Scenario 1). These quantities are plotted as function of the ESMDA iterations and contrasted with their reference values (horizontal lines). The source location $\mathbf{S}_\mathrm{l}$ is in m; and the contaminant release strength in each of the five stress periods, $\mathbf{S}_\mathrm{s}$, is in $\mathrm{g/m^3}$.

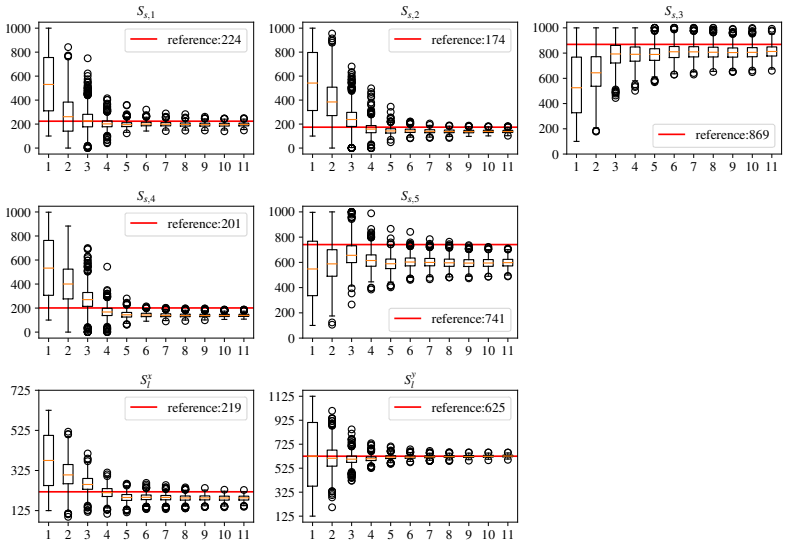

Figure 4.10: Boxplots of the ensembles for the contaminant release terms, $\mathbf{S} = (\mathbf{S}_\mathrm{l}, \mathbf{S}_\mathrm{s})$ with $\mathbf{S}_\mathrm{l} = (S_\mathrm{l}^x, S_\mathrm{l}^y)^\top$ and $\mathbf{S}_\mathrm{s} = (S_{\mathrm{s},1}, \ldots, S_{\mathrm{s},5})^\top$, and their confidence intervals, obtained via the CAAE-ESMDA inversion with the DenseED CNN surrogate (Scenario 2). These quantities are plotted as function of the ESMDA iterations and contrasted with their reference values (horizontal lines). The source location $\mathbf{S}_\mathrm{l}$ is in m; and the contaminant release strength in each of the five stress periods, $\mathbf{S}_\mathrm{s}$, is in $\mathrm{g/m^3}$.

Sparse Observation Network (Scenarios 3–5)

Figures 4.11–4.13 present the contaminant release history $\mathbf{S}$ estimated by our CAAE-DenseED-ESMDA inversion algorithm for the reduced amounts of concentration and head data in Scenarios 3–5, respectively. The algorithm's performance in Scenario 2 and 4, which differ only in the number of measurements along each well of the dense observation network, is very similar. Also, the inversion of data from Scenario 4 (more wells with lower vertical sampling density) is more accurate than its counterpart from Scenario 3 (fewer wells with higher vertical sampling density). These findings are reassuring, since very few real-world wells are screened in each layer of a numerical flow and transport model. As expected, the source identification on the data provided by the dense observation network (Scenario 2) is superior to that on the data from the sparse network (Scenarios 3 and 5). Even in the most data-poor Scenario 5, our inversion algorithm is able to capture the location of the contaminant release, albeit with a wide confidence interval.

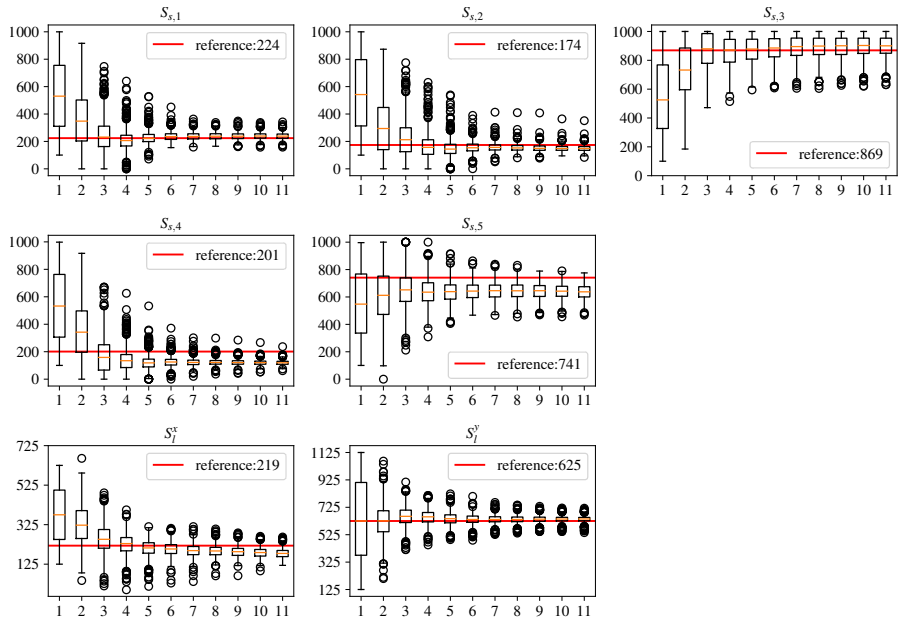

Figure 4.11: Boxplots of the ensembles for the contaminant release terms, $\mathbf{S} = (\mathbf{S}_\mathrm{l}, \mathbf{S}_\mathrm{s})$ with $\mathbf{S}_\mathrm{l} = (S_\mathrm{l}^x, S_\mathrm{l}^y)^\top$ and $\mathbf{S}_\mathrm{s} = (S_{\mathrm{s},1}, \ldots, S_{\mathrm{s},5})^\top$, and their confidence intervals, obtained via the CAAE-ESMDA inversion with the DenseED CNN surrogate (Scenario 3). These quantities are plotted as function of the ESMDA iterations and contrasted with their reference values (horizontal lines). The source location $\mathbf{S}_\mathrm{l}$ is in m; and the contaminant release strength in each of the five stress periods, $\mathbf{S}_\mathrm{s}$, is in $\mathrm{g/m^3}$.

Figure 4.14 further illuminates this inter-scenarios comparison by exhibiting the 95% confidence intervals (error bars) for the inversion of hydraulic head and solute concentration data for these five designs of the observation campaign. Having most data and relying on a more accurate (PDE-based)

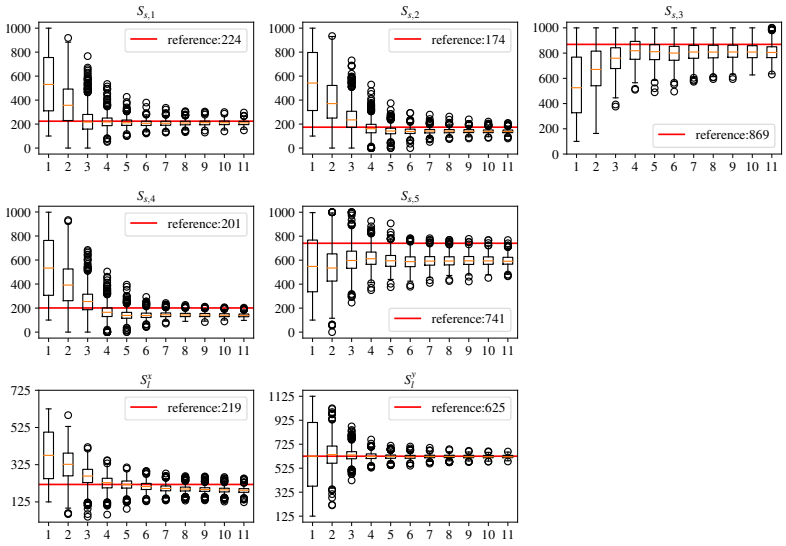

Figure 4.12: Boxplots of the ensembles for the contaminant release terms, $\mathbf{S} = (\mathbf{S}_\mathrm{l}, \mathbf{S}_\mathrm{s})$ with $\mathbf{S}_\mathrm{l} = (S_\mathrm{l}^x, S_\mathrm{l}^y)^\top$ and $\mathbf{S}_\mathrm{s} = (S_{\mathrm{s},1}, \ldots, S_{\mathrm{s},5})^\top$, and their confidence intervals, obtained via the CAAE-ESMDA inversion with the DenseED CNN surrogate (Scenario 4). These quantities are plotted as function of the ESMDA iterations and contrasted with their reference values (horizontal lines). The source location $\mathbf{S}_\mathrm{l}$ is in m; and the contaminant release strength in each of the five stress periods, $\mathbf{S}_\mathrm{s}$, is in $\mathrm{g/m^3}$.

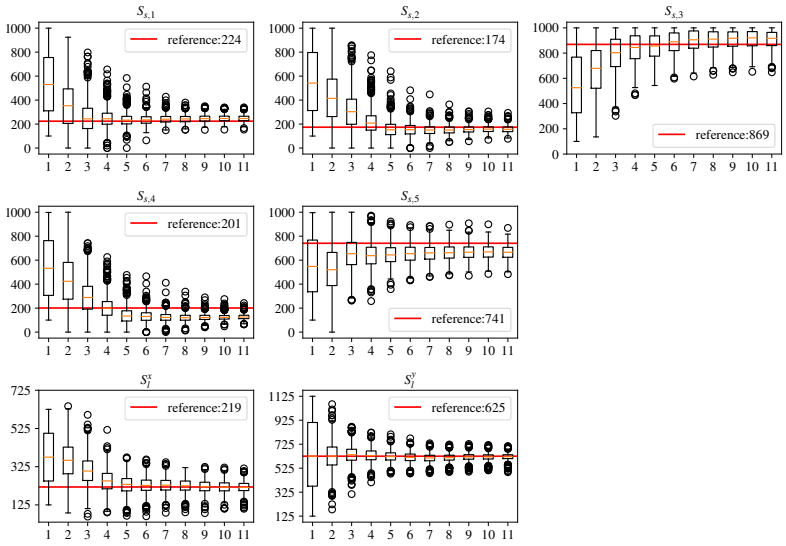

Figure 4.13: Boxplots of the ensembles for the contaminant release terms, $\mathbf{S} = (\mathbf{S}_\mathrm{l}, \mathbf{S}_\mathrm{s})$ with $\mathbf{S}_\mathrm{l} = (S_\mathrm{l}^x, S_\mathrm{l}^y)^\top$ and $\mathbf{S}_\mathrm{s} = (S_{\mathrm{s},1}, \ldots, S_{\mathrm{s},5})^\top$, and their confidence intervals, obtained via the CAAE-ESMDA inversion with the DenseED CNN surrogate (Scenario 5). These quantities are plotted as function of the ESMDA iterations and contrasted with their reference values (horizontal lines). The source location $\mathbf{S}_\mathrm{l}$ is in m; and the contaminant release strength in each of the five stress periods, $\mathbf{S}_\mathrm{s}$, is in $\mathrm{g/m^3}$.

forward model, the inversion in Scenario 1 achieves the best parameter estimation results (except for $S_{s,3}$). Scenarios 1, 2 and 4 yield comparable estimates of the contaminant release location. The estimate of S_l^x is less accurate than that of S_l^y, indicating it is easier to estimate the source location in the plane perpendicular to the flow direction.

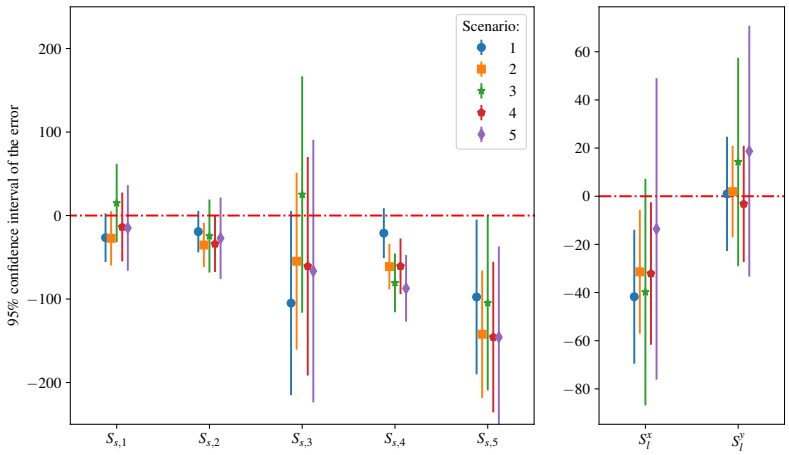

Figure 4.14: Error bars for estimation of the contaminant release history $\mathbf{S} = (\mathbf{S}_\mathrm{l}, \mathbf{S}_\mathrm{s})$, i.e., its location $\mathbf{S}_\mathrm{l} = (S_\mathrm{l}^x, S_\mathrm{l}^y)^\top$ and strength in each stress period, $\mathbf{S}_\mathrm{s} = (S_{\mathrm{s},1}, \ldots, S_{\mathrm{s},5})^\top$, for the data-availability scenarios in Table 4.5.

Figures 4.15–4.17 show reconstructions of the log-conductivity field from the hydraulic head and solute concentration data in Scenarios 3–5, respectively. While quantitative differences in the reconstruction of the posterior mean log-conductivity $\langle Y(\mathbf{x}) \rangle$ are hard to discern visually, the posterior standard deviation σ_Y varies between scenarios. This measure of predictive uncertainty is appreciably smaller in Scenario 4, which has most data, than in Scenarios 3 and 5, which have less. This again demonstrates an expected result: data availability is the key to successful inversion. The manifestation of this truism are the box-plots in Figures 4.11–4.13 and the error bars in Figure 4.14. It also validates the self-consistency of our parameter estimation algorithm.

Hydraulic Conductivity Estimation from Hydraulic Head Data

In addition to the five scenarios from Table 4.5, we consider the task of estimating the hydraulic conductivity of a heterogeneous aquifer from hydraulic head measurements. This task is important in its own right, as a classical hydraulic inversion problem. It also serves to ascertain the importance of tracer data (even of uncertain quality) for this task. Figure 4.18 exhibits the posterior mean and standard deviation of the log conductivity $Y(\mathbf{x})$ obtained by assimilating the hydraulic head data from the dense network of observational wells (Scenario 2). In the absence of solute concentration

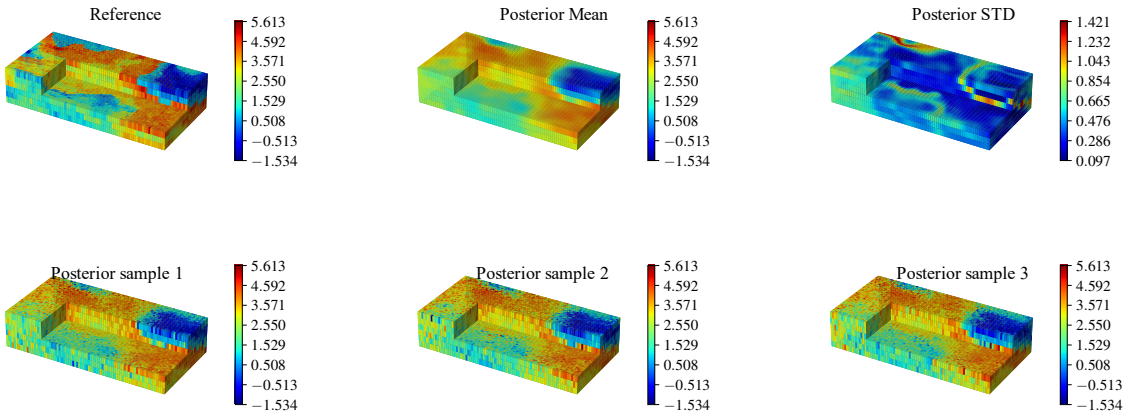

Figure 4.15: Estimates of the posterior mean ($\langle Y \rangle$, middle column) and standard deviation (σ_Y, right column) of log-conductivity $Y(\mathbf{x})$ obtained via CAAE-DenseED-ESMDA algorithm from the hydraulic head and solute concentration measurements in Scenario 3. Also shown is the reference log-conductivity field (left column) and representative realizations drawn from the posterior PDF $\mathcal{N}(\langle Y \rangle, \sigma_Y)$ (bottom row).

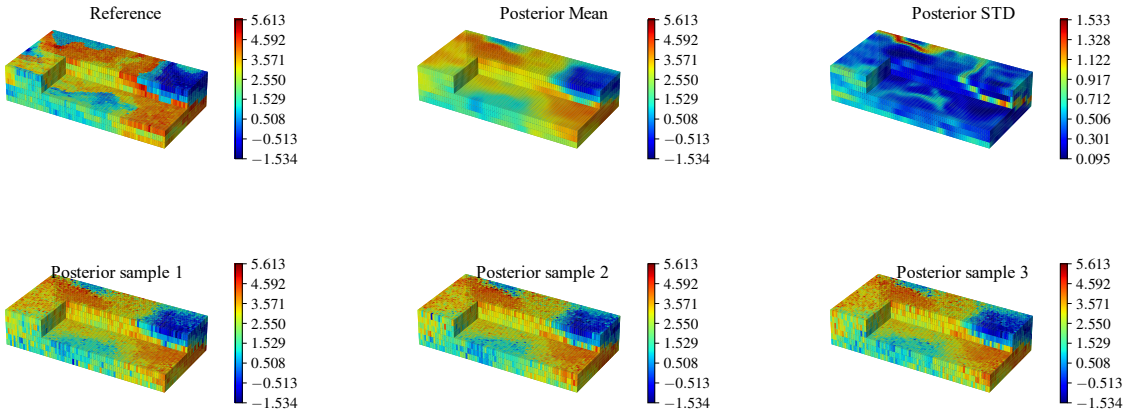

Figure 4.16: Estimates of the posterior mean ($\langle Y \rangle$, middle column) and standard deviation (σ_Y, right column) of log-conductivity $Y(\mathbf{x})$ obtained via CAAE-DenseED-ESMDA algorithm from the hydraulic head and solute concentration measurements in Scenario 4. Also shown is the reference log-conductivity field (left column) and representative realizations drawn from the posterior PDF $\mathcal{N}(\langle Y \rangle, \sigma_Y)$ (bottom row).

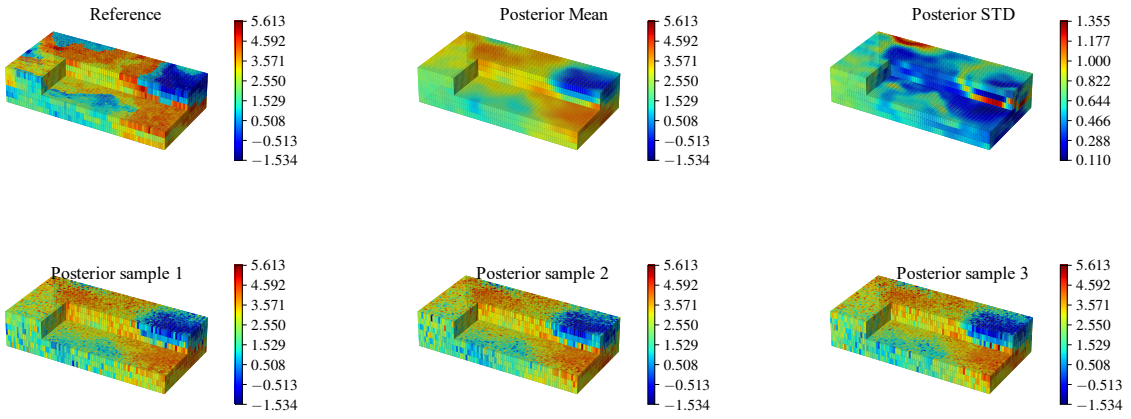

Figure 4.17: Estimates of the posterior mean ($\langle Y \rangle$, middle column) and standard deviation (σ_Y, right column) of log-conductivity $Y(\mathbf{x})$ obtained via CAAE-DenseED-ESMDA algorithm from the hydraulic head and solute concentration measurements in Scenario 5. Also shown is the reference log-conductivity field (left column) and representative realizations drawn from the posterior PDF $\mathcal{N}(\langle Y \rangle, \sigma_Y)$ (bottom row).

measurements, the posterior mean estimated via CAAE-DenseED-ESMDA algorithm misses some features on the left of the computational domain, which are captured in Scenarios 1 and 2. The posterior standard deviation $\sigma_Y(\mathbf{x})$ is much higher in a large part of the domain, relative to Scenarios 1 and 2 where the concentration data are used. Realizations of the log-conductivity field $Y(\mathbf{x})$ drawn from the resulting posterior PDF $\mathcal{N}(\langle Y \rangle, \sigma_Y)$ show higher variability and less continuity in the regions which $Y(\mathbf{x})$ is supposed to be smoother. Thus, not surprisingly, the addition of information in the form of concentration measurements reduces the estimation uncertainty, as quantified by $\sigma_Y(\mathbf{x})$.

4.4.5 Comparison of Computational Costs

The computational costs of CAAE-ESMDA with the PDE-based forward model and its counterpart with the DenseED CNN surrogate are shown in Table 4.6. CAAE-ESMDA with the PDE-based model ran on CPU, while the DenseED CNN surrogate was trained and simulated on GPUs provided by `GoogleColab`. In both cases, ESMDA consists of $N_\mathrm{e} = 960$ samples in each ensemble and 10 iterations are performed, resulting in $N_\mathrm{sum} = N_\mathrm{e} \times (10 + 1) = 10560$ forward model runs. Overall, CAAE-DenseED-ESMDA is two orders of magnitude faster than CAAE-ESMDA with the PDE-based forward model.

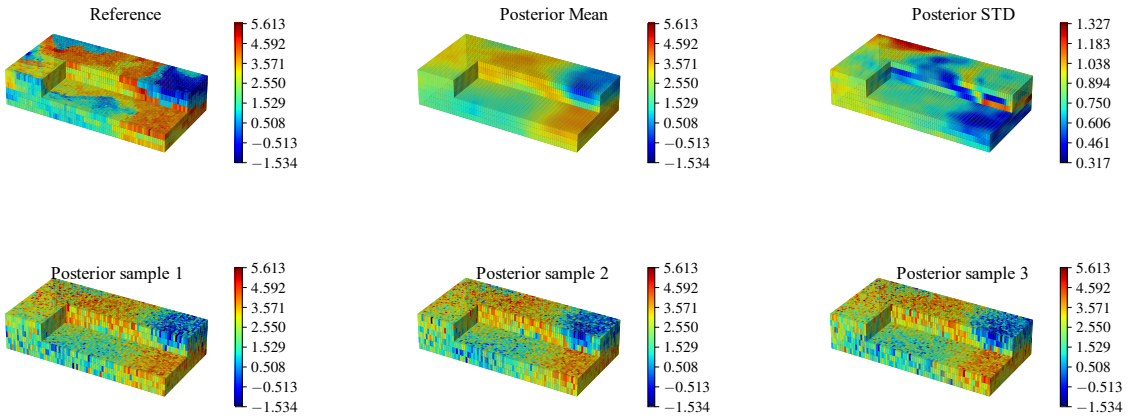

Figure 4.18: Estimates of the posterior mean ($\langle Y \rangle$, middle column) and standard deviation (σ_Y, right column) of log-conductivity $Y(\mathbf{x})$ obtained via CAAE-DenseED-ESMDA algorithm from the hydraulic head measurements only. Also shown is the reference log-conductivity field (left column) and representative realizations drawn from the posterior PDF $\mathcal{N}(\langle Y \rangle, \sigma_Y)$ (bottom row).

	T_{run}	T_{dataset}	T_{train}	T_{ave}
CAAE-ESMDA	388200.0	0.0	0.0	36.8
CAAE-DenseED-ESMDA	1893.9	34922.0	9439.2	4.4

Table 4.6: Total run time of the CAAE-ESMDA, T_{run}, includes the costs of the PDE-based forward model and its CNN surrogate. The average run-time per sample, T_{ave}, is defined as $T_{\text{ave}} = (T_{\text{run}} + T_{\text{dataset}} + T_{\text{train}})/N_{\text{sum}}$, where T_{dataset} is the time for obtaining the training and testing data sets, and T_{train} is the CNN training time. CAAE parameterization is used in both cases, the training time is 18678.23, the running time of CAAE is negligible in both data assimilation strategies. All times are in seconds.

4.5 Conclusions and Discussion

We proposed an CAAE-DenseED-ESMDA algorithm to infer the statistics of both aquifer properties (e.g., hydraulic conductivity) and contaminant release history from sparse and noisy observations of hydraulic head and solute concentration. The algorithm relies on CAEE to obtain a low-dimensional representation of the high-dimensional discretized conductivity field (and, if necessary, other spatially distributed input parameters); deploys a DenseED CNN surrogate of the PDE-based transport model to accelerate the forward runs; and adopts ESMDA to solve the inverse problem. The algorithm's computational efficiency is such that it enables one to handle three-dimensional problems.

To demonstrate the salient features of our inversion methodology, we conduct a series of numerical experiments. They deal with flow and transport in a three-dimensional heterogeneous aquifer with uncertain hydraulic conductivity field; our goal is to estimate the latter, and the contaminant release history, from the measurements of hydraulic head and contaminant concentration collected in a few observation wells. These numerical experiments lead to the following conclusions.

1. The CAAE-DenseED-ESMDA inversion framework is capable of both identifying the contaminant release source and reconstructing a three-dimensional hydraulic conductivity field from sparse (in space and time) and noisy measurements of solute concentration and hydraulic head.

2. The CAAE-ESMDA inversion, with or without the DenseED CNN surrogate of the PDE-based forward model, yields estimates of the contaminant release strength that differ from the reference values by up to 30.42%. That can be attributed to the imperfect reconstruction of hydraulic conductivity field or relative insensitivity of the observed solute concentrations to the contaminant release strengths in each stress period (the inverse problem's ill-posedness).

3. Deployment of the DenseED CNN surrogate within our CAAE-ESMDA inversion framework provides an order of magnitude speed up, while giving identical estimates of the hydraulic conductivity field; it also increases the predictive uncertainty (posterior standard deviation) relative to that obtained via the CAAE-ESMDA inversion with the PDE-based model.

4. The computational efficiency of CAAE-DenseED-ESMDA, relative to that of CAAE–ESMDA with the high-fidelity PDE model, is mostly due to the use of GPUs for CNN-related computations, while the PDE solver for the flow and transport model (e.g., MODLFLOW and MT3DMS) utilizes CPUs.

5. For the same number of observation wells, placing them in the direction of the regional hydraulic head gradient (as determined by hydraulic heads along the boundary of the simulation domain) results in a more accurate reconstruction of the contaminant release history than placing them in the transverse direction.

6. Deployment of CAAE-DenseED-ESMDA allows one to investigate questions, such as measuring the data assimilation accuracy versus the ensemble size or designing a network of observation wells, that cannot be answered with CAAE-ESMDA with the PDE-based forward model, whose computational cost might be prohibitive.

Although the flow and transport simulators, MODFLOW and MT3DMS, can be parallelized to run on multiple CPU cores, that is a much more arduous task than carrying out NN-related computations on GPUs available in Google-Colab or other cloud computing environments. The latter takes very little implementation effort and can be done on a personal computer. The advantage of our method largely depends on the feasibility of accessing GPU computing resources versus deploying multicores parallelization with the physics-based forward model.

Chapter 5

Overall Conclusions and Future Work

5.1 Conclusions and Discussion

This dissertation demonstrated versatile combinations of inversion frameworks and neural networks surrogate models on the application of subsurface flow and transport inverse problems. The two most popular inversion methods: MCMC and ensemble-based methods are applied to reconstruct contaminant release history and hydraulic conductivity from sparse noisy measurements of solute concentration and hydraulic head. Respectively, DRAM MCMC sampling and HMC sampling are applied on a medium dimensional inverse problems with eight unknown parameters in a two-dimensional source identification problem in Chapter 3, indicating that the usage of a neural network surrogate model can achieve comparable accuracy when used to reconstruct the contaminant release history, the Auto-Differentiation also enables gradient-based HMC sampling, which is often prohibitive if using a PDF solver to simulate the flow and transport model; ensemble based inversion framework, ESMDA, is used in an application to reconstruct a three-dimensional hydraulic conductivity field, and a dynamic contaminant release history in Chapter 4. The CAAE–DenseED–ESMDA inversion framework was able to identify the contaminant release location, and reconstruct a three-dimensional conductivity field with noisy solute concentration and hydraulic head measurements. Besides the two advanced inverse modeling methods, neural network surrogate models are used through all three applications to enhance the computational efficiency of the inversion framework while keeping comparable accuracy to the methods with physics-based forward models. In the application in Chapter 2, a brute-force inversion framework is made possible with the usage of a neural network surrogate. These applications and experiments indicate that the integration of neural network surrogate model with appropriate inversion method can provide an alternative in inverse modeling. We hope that

these studies will be helpful in the future for more complicated and realistic inverse problems.

5.2 Recommendations for Future Work

- A series of studies characterizing the fractional connected area (FCA) and other discrete fracture network (DFN) statistical characteristics in [87, 88, 86] indicated the effectiveness of using thermal breakthrough curves, electric potential, electrical resistivity in the inverse modeling. We leave it a future study to incorporate FCA into the input of our NN surrogate model, and inversion framework as well, potentially with more measurements in addition to the thermal breakthrough curves from cross-borehole thermal experiments (CBTEs).

- To continue the study of a conductivity field reconstruction and contaminant release history identification, applying a Bayesian approach on the forward surrogate modeling would be a direction to be explored: an ensemble of three-dimensional dense convolutional encoder-decoder (DenseED) networks are trained to perform as the forward surrogate for the flow and transport processes. The full workflow will be a combination of the CAAE and this Bayesian DenseED forward surrogate models, and an inversion framework such as ESMDA is used at the end, forming a CAAE-BayesianDenseED-ESMDA inversion framework. The Bayesian approach on the neural networks training enables a 'fully Bayesian' inversion, previously prohibitive due to the prediction error caused by a deterministic forward surrogate model.

- Inverse problems on contaminant release history or conductivity reconstruction always rely on an assumption that the forward model is a perfect model, whereas a realistic transport phenomena in the field can be very different. Recent studies on surrogate modeling gradually expanded from two-dimensional to three-dimensional, yet this might cause the gap between synthetic data inverse modeling and real data inverse modeling even larger. Besides effort to model the physical processes with more accurate high-fidelity simulators, we hope that surrogate modeling can someday be trained on real dataset and used in a more realistic scenario.

- Contaminant release history identification with less prior knowledge about the release location and time is of our interests for future study. In our and many other research identification cited in Chapter 3 and Chapter 4, the contaminant release region is restricted in a small area relative to the whole domain. A practical issue with relaxing this constraint in our study is related to the effort on neural network surrogate model training. Since neural network surrogates do not easily generalize to predict the concentration transport or the fluid flow beyond the time of training data, the training dataset usually has to cover the prior intervals of these unknown parameters. A larger release region or more versatile release time setting would cause the requirement of the training dataset to be increased. For this concern, we would like to explore neural network architectures or training techniques that saves training data, or converges faster

in the training process. The fewer physics-based forward model runs needed, the more obvious the advantage of training and using a surrogate model is.

- Dense nonaqueous phase liquid (DNAPL) is a type of contaminant source that exhibits very different transport properties than soluble contaminant in groundwater, and in fact [20], found in many aquifers. Contaminant source identification of DNAPL with neural network surrogate forward model remains a future study interest for us.

- Principal component geostatistical approach (PCGA) [64], was applied in a study of bathymetry imaging [76] and outperformed ensemble based method in terms of accuracy and computational efficiency. We are interested in a comparison of CAAE-PCGA inversion strategy on a contaminant source identification problem to our CAAE-ESMDA method shown in Chapter 4, where CAAE is again used to parameterize the non-Gaussian conductivity field as a low dimensional latent variable $\mathbf{z}$ which is constructed to be Gaussian.

- Another possible future work is about a study on thermal-hydrologic-chemical physical processes involved in heat conduction and energy transfer due to fluid flow and chemical transport in a site-scale reservoir [97]. The governing equations here are restricted to be for fluid flow, chemical transport, and heat transfer processes. Generally, the detailed permeability field remains unknown even with some core measurements, hence some inverse analysis is needed. The goal of this study would be to identify the permeability, initial temperature profile, and the heat flux on the bottom of a reservoir, given some measurements of the temperature and tracer concentration taken at later times. The subsurface simulator PFLOTRAN [46] can solve the system of nonlinear partial differential equations of a multiphase, multicomponent, and multiscale reactive flow and transport in porous materials. However, the physics involved in this thermal-hydrologic-chemical process is simulated by solving coupled PDEs, hence the computational cost with PFLOTRAN is again prohibitive for inverse modeling which requires a large number of forward runs. The combination of MCMC or ESMDA and a neural network surrogate model can again be used to solve this inverse problem.

Bibliography

[1] A. F. Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] R. A. Akoachere II and G. Van Tonder. The trigger-tube: A new apparatus and method for mixing solutes for injection tests in boreholes. *Water SA*, 37(2), 2011.

[3] M. Amirabdollahian and B. Datta. Identification of contaminant source characteristics and monitoring network design in groundwater aquifers: An overview. *Journal of Environmental Protection*, 4:26–41, 2013.

[4] M. M. Aral, J. Guan, and M. L. Maslia. Identification of contaminant source location and release history in aquifers. *Journal of hydrologic engineering*, 6(3):225–234, 2001.

[5] M. T. Ayvaz. A hybrid simulation–optimization approach for solving the areal groundwater pollution source identification problems. *Journal of Hydrology*, 538:161–176, 2016.

[6] M. Bakker, V. Post, C. D. Langevin, J. D. Hughes, J. White, J. Starn, and M. N. Fienen. Scripting modflow model development using python and flopy. *Groundwater*, 54(5):733–739, 2016.

[7] D. A. Barajas-Solano, F. J. Alexander, M. Anghel, and D. M. Tartakovsky. Efficient gHMC reconstruction of contaminant release history. *Frontiers in Environmental Science*, 7:149, 2019. doi: 10.3389/fenvs.2019.00149.

[8] V. Bedekar, E. D. Morway, C. D. Langevin, and M. J. Tonkin. MT3D-USGS version 1: A US Geological Survey release of MT3DMS updated with new and expanded transport capabilities for use with MODFLOW. Technical report, US Geological Survey, Reston, VA, 2016.

[9] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

[10] E. Bonnet, O. Bour, N. E. Odling, P. Davy, I. Main, P. Cowie, and B. Berkowitz. Scaling of fracture systems in geological media. *Reviews of Geophysics*, 39(3):347–383, 2001. ISSN 1944-9208. doi: 10.1029/1999RG000074. URL `http://dx.doi.org/10.1029/1999RG000074`.

[11] F. Boso and D. M. Tartakovsky. Data-informed method of distributions for hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, 42(1):A559–A583, 2020. doi: 10.1137/19M1260773.

[12] F. Boso and D. M. Tartakovsky. Learning on dynamic statistical manifolds. *Proceedings of the Royal Society A*, 476(2239):20200213, 2020. doi: 10.1098/rspa.2020.0213.

[13] F. Boso and D. M. Tartakovsky. Data-informed method of distributions for hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, 42(1):A559–A583, 2020.

[14] F. Boso and D. M. Tartakovsky. Learning on dynamic statistical manifolds. *Proceedings of the Royal Society A*, 476(2239):20200213, 2020.

[15] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[16] S. W. Canchumuni, A. A. Emerick, and M. A. C. Pacheco. History matching geological facies models based on ensemble smoother and deep generative models. *Journal of Petroleum Science and Engineering*, 177:941–958, 2019.

[17] A. Chaudhuri, H.-J. Hendricks-Franssen, and M. Sekhar. Iterative filter based estimation of fully 3D heterogeneous fields of permeability and Mualem-van Genuchten parameters. *Advances in Water Resources*, 122:340–354, 2018.

[18] F.-C. Chen and M. R. Jahanshahi. NB-CNN: Deep learning-based crack detection using convolutional neural network and Naïve Bayes data fusion. *IEEE Transactions on Industrial Electronics*, 65(5):4392–4400, 2017.

[19] V. Ciriello, I. Lauriola, and D. M. Tartakovsky. Distribution-based global sensitivity analysis in hydrology. *Water Resources Research*, 55(11):8708–8720, 2019. doi: 10.1029/2019WR025844.

[20] N. R. Council et al. *Alternatives for managing the nation's complex contaminated groundwater sites*. National Academies Press, 2013.

[21] P. Davy, A. Sornette, and D. Sornette. Some consequences of a proposed fractal nature of continental faulting. *Nature*, 348(6296):56–58, 1990.

[22] J. de La Bernardie, O. Bour, T. Le Borgne, N. Guihéneuf, E. Chatton, T. Labasque, H. Le Lay, and M.-F. Gerard. Thermal attenuation and lag time in fractured rock: Theory and field

measurements from joint heat and solute tracer tests. *Water Resources Research*, 54(12): 10–053, 2018.

[23] S. Demirel, D. Roubinet, J. Irving, and E. Voytek. Characterizing near-surface fractured-rock aquifers: Insights provided by the numerical analysis of electrical resistivity experiments. *Water*, 10(9):1117, 2018.

[24] C. Dorn, N. Linde, T. Le Borgne, O. Bour, and M. Klepikova. Inferring transport characteristics in a fractured rock aquifer by combining single-hole ground-penetrating radar reflection monitoring and tracer test data. *Water Resources Research*, 48(11), 2012.

[25] C. Dorn, N. Linde, T. Le Borgne, O. Bour, and J.-R. de Dreuzy. Conditioning of stochastic 3-d fracture networks to hydrological and geophysical data. *Advances in Water Resources*, 62, Part A(0):79–89, 2013. ISSN 0309-1708. doi: 10.1016/j.advwatres.2013.10.005. URL `http://www.sciencedirect.com/science/article/pii/S0309170813001863`.

[26] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987.

[27] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

[28] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[29] A. H. Elsheikh, I. Hoteit, and M. F. Wheeler. Efficient Bayesian inference of subsurface flow models using nested sampling and sparse polynomial chaos surrogates. *Computer Methods in Applied Mechanics and Engineering.*, 269:515–537, 2014.

[30] A. A. Emerick and A. C. Reynolds. Ensemble smoother with multiple data assimilation. *Computers & Geosciences*, 55:3–15, 2013.

[31] G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5): 10143–10162, 1994.

[32] G. Evensen. The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

[33] P. Fischer, A. Jardani, and N. Lecoq. Hydraulic tomography of discrete networks of conduits and fractures in a karstic aquifer by using a deterministic inversion algorithm. *Advances in Water Resources*, 112:83–94, 2018.

[34] D. Gamerman and H. F. Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. Chapman and Hall/CRC, 2006.

[35] G. Gao, A. C. Reynolds, et al. An improved implementation of the lbfgs algorithm for automatic history matching. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2004.

[36] J. F. Geweke. *Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments*, volume 196. Federal Reserve Bank of Minneapolis, Research Department, Minneapolis, MN, 1991.

[37] H. Ghorbanidehno, A. Kokkinaki, J. Lee, and E. Darve. Recent developments in fast and scalable inverse modeling and data assimilation methods in hydrology. *Journal of Hydrology*, page 125266, 2020.

[38] V. R. Gisladottir, D. Roubinet, and D. M. Tartakovsky. Particle methods for heat transfer in fractured media. *Transport in Porous Media*, 115(2):311–326, 2016.

[39] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Boston, MA, 2016. `http://www.deeplearningbook.org`.

[40] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[41] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[42] P. J. Green and A. Mira. Delayed rejection in reversible jump Metropolis–Hastings. *Biometrika*, 88(4):1035–1053, 2001.

[43] H. Haario, E. Saksman, and J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7 (2):223–242, 2001.

[44] H. Haario, M. Laine, A. Mira, and E. Saksman. DRAM: efficient adaptive MCMC. *Statistics and Computing volume*, 16(4):339–354, 2006.

[45] A. S. Haddad, H. Hassanzadeh, J. Abedi, and Z. Chen. Application of tracer injection tests to characterize rock matrix block size distribution and dispersivity in fractured aquifers. *Journal of Hydrology*, 510:504–512, 2014.

[46] G. E. Hammond, P. C. Lichtner, and R. Mills. Evaluating the performance of parallel subsurface simulators: An illustrative example with pflotran. *Water Resources Research*, 50(1): 208–228, 2014.

[47] A. W. Harbaugh. *MODFLOW-2005, the US Geological Survey modular ground-water model: the ground-water flow process*. US Department of the Interior, US Geological Survey, Reston, VA, 2005.

[48] M. Herlihy, N. Shavit, V. Luchangco, and M. Spear. *The art of multiprocessor programming*. Newnes, 2020.

[49] G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012.

[50] T. Hirata. Fractal dimension of fault systems in japan: fractal structure in rock fracture geometry at various scales. In *Fractals in geophysics*, pages 157–170. Springer, 1989.

[51] 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(1):1593–1623, 2014.

[52] I. Jang, J. Kang, and C. Park. Inverse fracture model integrating fracture statistics and well-testing data. *Energy Sources Part A Recovery Utilization and Environmental Effects*, 30(18): 1677–1688, 2008.

[53] Y. H. Jang, T. H. Lee, J. H. Jung, S. I. Kwon, and W. M. Sung. The oil production performance analysis using discrete fracture network model with simulated annealing inverse method. *Geosciences Journal*, 17(4):489–496, 2013.

[54] S. Jiang and L. J. Durlofsky. Data-space inversion using a recurrent autoencoder for time-series parameterization. *Computational Geosciences*, 25(1):411–432, 2021.

[55] J. Joseph and J. H. Guillaume. Using a parallelized mcmc algorithm in r to identify appropriate likelihood functions for swat. *Environmental Modelling & Software*, 46:292–298, 2013.

[56] L. Ju, J. Zhang, L. Meng, L. Wu, and L. Zeng. An adaptive gaussian process-based iterative ensemble smoother for data assimilation. *Advances in Water Resources*, 115:125–135, 2018.

[57] X. Kang, A. Kokkinaki, P. K. Kitanidis, X. Shi, A. Revil, J. Lee, A. Soueid Ahmed, and J. Wu. Improved characterization of dnapl source zones via sequential hydrogeophysical inversion of hydraulic-head, self-potential and partitioning tracer data. *Water Resources Research*, 56(8): e2020WR027627, 2020.

[58] X. Kang, A. Kokkinaki, P. K. Kitanidis, X. Shi, J. Lee, S. Mo, and J. Wu. Hydrogeophysical characterization of nonstationary DNAPL source zones by integrating a convolutional variational autoencoder and ensemble smoother. *Water Resources Research*, 57(2):e2020WR028538, 2021.

[59] S. Kim, B. Min, S. Kwon, and M.-g. Chu. History matching of a channelized reservoir using a serial denoising autoencoder integrated with es-mda. *Geofluids*, 2019, 2019.

[60] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[61] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[62] P. K. Kitanidis. Quasi-linear geostatistical theory for inversing. *Water Resources Research*, 31(10):2411–2419, 1995.

[63] P. K. Kitanidis. On the geostatistical approach to the inverse problem. *Advances in Water Resources*, 19(6):333–342, 1996.

[64] P. K. Kitanidis and J. Lee. Principal component geostatistical approach for large-dimensional inverse problems. *Water Resources Research*, 50(7):5428–5443, 2014.

[65] M. V. Klepikova, T. Le Borgne, O. Bour, K. Gallagher, R. Hochreutener, and N. Lavenant. Passive temperature tomography experiments to characterize transmissivity and connectivity of preferential flow paths in fractured media. *Journal of Hydrology*, 512:549–562, 2014.

[66] M. V. Klepikova, T. Le Borgne, O. Bour, M. Dentz, R. Hochreutener, and N. Lavenant. Heat as a tracer for understanding transport processes in fractured media: Theory and field assessment from multiscale thermal push-pull tracer tests. *Water Resources Research*, 52(7):5442–5457, 2016.

[67] M. V. Klepikova, T. Le Borgne, O. Bour, and J.-R. de Dreuzy. Inverse modeling of flow tomography experiments in fractured media. *Water Resources Research*, 49(11):7255–7265, NOV 2013. ISSN 0043-1397. doi: {10.1002/2013WR013722}.

[68] K. Kreutz-Delgado. The complex gradient operator and the cr-calculus. *arXiv preprint arXiv:0906.4835*, 2009.

[69] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[70] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Tran. Neural Net.*, 9(5):987–1000, 1998.

[71] A. Lang and J. Potthoff. Fast simulation of Gaussian random fields. *Monte Carlo Methods and Applications*, 17(3):195–214, 2011.

[72] L. Le Cam. *Asymptotic methods in statistical decision theory*. Springer Science & Business Media, 2012.

[73] R. Le Goc, J.-R. De Dreuzy, and P. Davy. An inverse problem methodology to identify flow channels in fractured media using synthetic steady-state head and geometrical data. *Advances in Water Resources*, 33(7):782–800, 2010.

[74] H. Lee and I. S. Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131, 1990.

[75] J. Lee, H. Yoon, P. K. Kitanidis, C. J. Werth, and A. J. Valocchi. Scalable subsurface inverse modeling of huge data sets with an application to tracer concentration breakthrough data from magnetic resonance imaging. *Water Resources Research*, 52(7):5213–5231, 2016.

[76] J. Lee, H. Ghorbanidehno, M. W. Farthing, T. J. Hesser, E. F. Darve, and P. K. Kitanidis. Riverine bathymetry imaging with indirect observations. *Water Resources Research*, 54(5): 3704–3727, 2018.

[77] S. Leichombam and R. K. Bhattacharjya. New hybrid optimization methodology to identify pollution sources considering the source locations and source flux as unknown. *Journal of Hazardous, Toxic, and Radioactive Waste*, 23(1):04018037, 2018.

[78] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[79] J. E. Liggett, A. D. Werner, B. D. Smerdon, D. Partington, and C. T. Simmons. Fully integrated modeling of surface-subsurface solute transport and the effect of dispersion in tracer hydrograph separation. *Water Resources Research*, 50(10):7750–7765, 2014.

[80] J. E. Liggett, D. Partington, S. Frei, A. D. Werner, C. T. Simmons, and J. H. Fleckenstein. An exploration of coupled surface–subsurface solute transport in a fully integrated catchment model. *Journal of Hydrology*, 529:969–979, 2015.

[81] N. Linde, P. Renard, T. Mukerji, and J. Caers. Geological realism in hydrogeological and geophysical inverse modeling: A review. *Advances in Water Resources*, 86:86–101, 2015.

[82] Y. Liu, W. Sun, and L. J. Durlofsky. A deep-learning-based geological parameterization for history matching complex models. *Mathematical Geosciences*, 51(6):725–766, 2019.

[83] G. Lods, D. Roubinet, J. M. Matter, R. Leprovost, P. Gouze, and O. D. P. S. Team. Groundwater flow characterization of an ophiolitic hard-rock aquifer from cross-borehole multi-level hydraulic experiments. *Journal of Hydrology*, 589:125152, 2020.

[84] H. Lu and D. M. Tartakovsky. Prediction accuracy of dynamic mode decomposition. *SIAM Journal on Scientific Computing*, 42(3):A1639–A1662, 2020.

[85] H. Lu and D. M. Tartakovsky. Lagrangian dynamic mode decomposition for construction of reduced-order models of advection-dominated phenomena. *Journal of Computational Physics*, 407:109229, 2020.

[86] L. Magnúsdóttir. *Fracture Characterization in Geothermal Reservoirs Using Time-lapse Electric Potential Data*. Stanford University, 2013.

[87] L. Magnusdottir and R. Horne. Fracture connectivity of fractal fracture networks estimated using electrical resistivity. In *Proceedings, 38th Workshop on Geothermal Reservoir Engineering. Stanford University, Stanford*, 2013.

[88] L. Magnusdottir and R. N. Horne. Inversion of time-lapse electric potential data to estimate fracture connectivity in geothermal reservoirs. *Mathematical Geosciences*, 47(1):85–104, 2015.

[89] I. G. Main, P. G. Meredith, P. R. Sammonds, and C. Jones. Influence of fractal flaw distributions on rock deformation in the brittle field. *Geological Society, London, Special Publications*, 54(1):81–96, 1990.

[90] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations (ICLR)*, 2016.

[91] G. Mariethoz and B. F. Kelly. Modeling complex geological structures with elementary training images and transform-invariant distances. *Water Resources Research*, 47(7), 2011.

[92] A. M. Michalak and P. K. Kitanidis. A method for enforcing parameter nonnegativity in Bayesian inverse problems with an application to contaminant source identification. *Water Resources Research*, 39(2), 2003.

[93] P. Miles. pymcmcstat: A Python package for Bayesian inference using delayed rejection adaptive Metropolis. *Journal of Open Source Software,*, 4(38):1417, 2019.

[94] S. Mo, N. Zabaras, X. Shi, and J. Wu. Deep autoregressive neural networks for high-dimensional inverse problems in groundwater contaminant source identification. *Water Resources Research*, 55(5):3856–3881, 2019.

[95] S. Mo, N. Zabaras, X. Shi, and J. Wu. Integration of adversarial autoencoders with residual dense convolutional networks for inversion of solute transport in non-gaussian conductivity fields. *Water Resources Research*, 2019.

[96] S. Mo, Y. Zhu, N. Zabaras, X. Shi, and J. Wu. Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. *Water Resources Research*, 55(1):703–728, 2019.

[97] M. K. Mudunuru, B. Ahmmed, S. Karra, V. V. Vesselinov, D. R. Livingston, and R. S. Middleton. Site-scale and regional-scale modeling for geothermal resource analysis and exploration. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2020.

[98] R. M. Neal. *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, Ontario, Canada, 1993.

[99] S. P. Neuman and D. M. Tartakovsky. Perspective on theories of non-Fickian transport in heterogeneous media. *Advances in Water Resources*, 32(5):670–680, 2009. doi: 10.1016/j.advwatres.2008.08.005.

[100] F. L. Paillet, J. H. Williams, J. Urik, J. Lukes, M. Kobr, and S. Mares. Cross-borehole flow analysis to characterize fracture connections in the melechov granite, bohemian-moravian highland, czech republic. *Hydrogeology Journal*, 20(1):143–154, 2012.

[101] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[102] P. E. Pehme, J. P. Greenhouse, and B. L. Parker. The active line source temperature logging technique and its application in fractured rock hydrogeology. *Journal of Environmental & Engineering Geophysics*, 12(4):307–322, 2007.

[103] P. E. Pehme, B. Parker, J. Cherry, J. Molson, and J. Greenhouse. Enhanced detection of hydraulically active fractures by temperature profiling in lined heated bedrock boreholes. *Journal of Hydrology*, 484:1–15, 2013.

[104] T. Ptak, M. Piepenbrink, and E. Martac. Tracer tests for the investigation of heterogeneous porous media and stochastic modelling of flow and transport—a review of some recent developments. *Journal of hydrology*, 294(1-3):122–163, 2004.

[105] M. M. Rajabi, B. Ataie-Ashtiani, and C. T. Simmons. Model-data interaction in groundwater studies: Review of methods, applications and future directions. *Journal of Hydrology*, 567: 457–477, 2018.

[106] T. Read, O. Bour, V. Bense, T. Le Borgne, P. Goderniaux, M. Klepikova, R. Hochreutener, N. Lavenant, and V. Boschero. Characterizing groundwater flow and heat transport in fractured rock using fiber-optic distributed temperature sensing. *Geophysical Research Letters*, 40 (10):2055–2059, 2013.

[107] D. Roubinet, H.-H. Liu, and J.-R. de Dreuzy. A new particle-tracking approach to simulating transport in heterogeneous fractured porous media. *Water Resources Research*, 46:W11507,

2010. ISSN 1944-7973. doi: 10.1029/2010WR009371. URL `http://dx.doi.org/10.1029/2010WR009371`.

[108] D. Roubinet, J. Irving, and F. D. Day-Lewis. Development of a new semi-analytical model for cross-borehole flow experiments in fractured media. *Advances in Water Resources*, 76:97–108, 2015.

[109] D. Roubinet, J. Irving, and P. A. Pezard. Relating topological and electrical properties of fractured porous media: Insights into the characterization of rock fracturing. *Minerals*, 8(1): 14, 2018. doi: 10.3390/min8010014.

[110] D. Roubinet, J.-R. de Dreuzy, and D. M. Tartakovsky. Particle-tracking simulations of anomalous transport in hierarchically fractured rocks. *Computers & Geosciences*, 50(SI):52–58, JAN 2013. ISSN 0098-3004. doi: 10.1016/j.cageo.2012.07.032.

[111] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[112] A. Ruiz Martinez, D. Roubinet, and D. M. Tartakovsky. Analytical models of heat conduction in fractured rocks. *Journal of Geophysical Research-Solid Earth*, 119(1):83–98, JAN 2014. ISSN 2169-9313. doi: 10.1002/2012JB010016.

[113] M. Sahimi. *Flow and transport in porous media and fractured rock: from classical methods to modern approaches*. John Wiley & Sons, 2011.

[114] P. Sarma, L. J. Durlofsky, and K. Aziz. Kernel principal component analysis for efficient, differentiable parameterization of multipoint geostatistics. *Mathematical Geosciences*, 40(1): 3–32, 2008.

[115] C. Scholz, N. Dawers, J.-Z. Yu, M. Anders, and P. Cowie. Fault growth and fault scaling laws: Preliminary results. *Journal of Geophysical Research: Solid Earth*, 98(B12):21951–21961, 1993.

[116] G. Severino, D. M. Tartakovsky, G. Srinivasan, and H. Viswanathan. Lagrangian models of reactive transport in heterogeneous porous media with uncertain properties. *Proceedings of the Royal Society A*, 468(2140):1154–1174, 2012. doi: 10.1098/rspa.2011.0375.

[117] M. F. Snodgrass and P. K. Kitanidis. A geostatistical approach to contaminant source identification. *Water Resources Research*, 33(4):537–546, 1997.

[118] A. Sokal. Monte Carlo methods in statistical mechanics: foundations and new algorithms. In *Functional integration*, pages 131–192. Springer, 1997.

[119] M. Somogyvári, M. Jalali, S. Jimenez Parras, and P. Bayer. Synthetic fracture network characterization with transdimensional inversion. *Water Resources Research*, 53(6):5104–5123, 2017.

[120] G. Srinivasan, D. M. Tartakovsky, M. Dentz, H. Viswanathan, B. Berkowitz, and B. A. Robinson. Random walk particle tracking simulations of non-Fickian transport in heterogeneous media. *J. Comput. Phys.*, 229(11):4304–4314, 2010. doi: 10.1016/j.jcp.2010.02.014.

[121] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147. PMLR, 2013.

[122] M. Tang, Y. Liu, and L. J. Durlofsky. Deep-learning-based surrogate flow modeling and geological parameterization for data assimilation in 3d subsurface flow. *Computer Methods in Applied Mechanics and Engineering*, 376:113636, 2021.

[123] D. M. Tartakovsky and C. L. Winter. Uncertain future of hydrogeology. *Journal of Hydrologic Engineering*, 13(1):37–39, 2008.

[124] P. J. Van Leeuwen and G. Evensen. Data assimilation and inverse methods in terms of a probabilistic formulation. *Monthly Weather Review*, 124(12):2898–2913, 1996.

[125] H. X. Vo and L. J. Durlofsky. A new differentiable parameterization based on principal component analysis for the low-dimensional representation of complex geological models. *Mathematical Geosciences*, 46(7):775–813, 2014.

[126] K. Watanabe and H. Takahashi. Fractal geometry characterization of geothermal reservoir fracture networks. *Journal of Geophysical Research: Solid Earth*, 100(B1):521–528, 1995.

[127] G. Wen, M. Tang, and S. M. Benson. Towards a predictor for co2 plume migration using deep neural networks. *International Journal of Greenhouse Gas Control*, 105:103223, 2021.

[128] R. E. White. Nonlinear least squares algorithm for identification of hazards. *Cogent Math.*, 2 (1):1118219, 2015.

[129] C. L. Winter, D. M. Tartakovsky, and A. Guadagnini. Moment equations for flow in highly heterogeneous porous media. *Surveys in Geophysics*, 24(1):81–106, 2003.

[130] C. L. Winter, A. Guadagnini, D. Nychka, and D. M. Tartakovsky. Multivariate sensitivity analysis of saturated flow through simulated highly heterogeneous groundwater aquifers. *J. Comput. Phys.*, 217(1):166–175, 2006.

[131] Z. Wu, A. C. Reynolds, and D. S. Oliver. Conditioning geostatistical models to two-phase production data. *SPE journal*, 4(02):142–155, 1999.

[132] Y. Xia and N. Zabaras. Bayesian multiscale deep generative model for the solution of high-dimensional inverse problems. *arXiv preprint arXiv:2102.03169*, 2021.

[133] T. Xu and J. J. Gómez-Hernández. Joint identification of contaminant source location, initial release time, and initial solute concentration in an aquifer via ensemble Kalman filtering. *Water Resources Research*, 52(8):6587–6595, 2016.

[134] T. Xu and J. J. Gómez-Hernández. Simultaneous identification of a contaminant source and hydraulic conductivity via the restart normal-score ensemble Kalman filter. *Advances in Water Resources*, 112:106–123, 2018.

[135] H.-J. Yang, F. Boso, H. A. Tchelepi, and D. M. Tartakovsky. Method of distributions for quantification of geologic uncertainty in flow simulations. *Water Resources Research*, 56(7): e2020WR027643, 2020. doi: 10.1029/2020WR027643.

[136] Z. Ye, A. Gilman, Q. Peng, K. Levick, P. Cosman, and L. Milstein. Comparison of neural network architectures for spectrum sensing. In *2019 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2019.

[137] H.-D. Yeh, T.-H. Chang, and Y.-C. Lin. Groundwater contaminant source identification by a hybrid heuristic approach. *Water Resources Research*, 43(9), 2007.

[138] J. Zhang, L. Zeng, C. Chen, D. Chen, and L. Wu. Efficient Bayesian experimental design for contaminant source identification. *Water Resources Research*, 51(1):576–598, 2015.

[139] J. Zhang, W. Li, L. Zeng, and L. Wu. An adaptive Gaussian process-based method for efficient Bayesian experimental design in groundwater contaminant source identification problems. *Water Resources Research*, 52(8):5971–5984, 2016.

[140] J. Zhang, G. Lin, W. Li, L. Wu, and L. Zeng. An iterative local updating ensemble smoother for estimation and uncertainty assessment of hydrologic model parameters with multimodal distributions. *Water Resources Research*, 54(3):1716–1733, 2018.

[141] C. Zheng, P. P. Wang, et al. Mt3dms: a modular three-dimensional multispecies transport model for simulation of advection, dispersion, and chemical reactions of contaminants in groundwater systems; documentation and user's guide. 1999.

[142] H. Zhou, J. J. Gómez-Hernández, and L. Li. Inverse methods in hydrogeology: Evolution and recent trends. *Advances in Water Resources*, 63:22–37, 2014.

[143] Z. Zhou and D. M. Tartakovsky. Markov chain monte carlo with neural network surrogates: Application to contaminant source identification. *Stochastic Environmental Research and Risk Assessment*, 35(3):639–651, 2021.

[144] Z. Zhou, D. Roubinet, and D. M. Tartakovsky. Thermal experiments for fractured rock characterization: theoretical analysis and inverse modeling. *arXiv preprint arXiv:2106.06632*, 2021.

[145] R. A. Zimmerman and D. M. Tartakovsky. Solute dispersion in bifurcating networks. *Journal of Fluid Mechanics*, 901:A24, 2020.