Learning Groundwater Contaminant Diffusion-Sorption Processes with a Finite Volume Neural Network

Timothy Praditia[1], Matthias Karlbauer[2], Sebastian Otte[2], Sergey Oladyshkin[1], Martin V. Butz[2], and Wolfgang Nowak[3]

[1]University of Stuttgart
[2]University of Tübingen
[3]Universität Stuttgart

November 24, 2022

Abstract

Improved understanding of complex hydrosystem processes is key to advance water resources research. Nevertheless, the conventional way of modeling these processes suffers from a high conceptual uncertainty, due to almost ubiquitous simplifying assumptions used in model parameterizations/closures. Machine learning (ML) models are considered as a potential alternative, but their generalization abilities remain limited. For example, they normally fail to predict across different boundary conditions. Moreover, as a black box, they do not add to our process understanding or to discover improved parameterizations/closures. To tackle this issue, we propose the hybrid modeling framework FINN (finite volume neural network). It merges existing numerical methods for partial differential equations (PDEs) with the learning abilities of artificial neural networks (ANNs). FINN is applied on discrete control volumes and learns components of the investigated system equations, such as numerical stencils, model parameters, and arbitrary closure/constitutive relations. Consequently, FINN yields highly interpretable results. To show this, we demonstrate FINN on a diffusion-sorption problem in clay. Results on numerically generated data show that FINN outperforms other ML models when tested under modified boundary conditions, and that it can successfully differentiate between the usual, known sorption isotherms. Moreover, we also equip FINN with uncertainty quantification methods to lay open the total uncertainty of scientific learning, and then apply it to a laboratory experiment. The results show that FINN performs better than calibrated PDE-based models as it is not restricted to choose among a limited set of sorption isotherms.

Learning Groundwater Contaminant Diffusion-Sorption Processes with a Finite Volume Neural Network

Timothy Praditia[1], Matthias Karlbauer[2], Sebastian Otte[2], Sergey Oladyshkin[1], Martin V. Butz[2], Wolfgang Nowak[1]

[1]Department of Stochastic Simulation and Safety Research for Hydrosystems, Institute for Modelling Hydraulic and Environmental Systems, University of Stuttgart, Stuttgart, Germany
[2]Neuro-Cognitive Modeling Group, Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, Tübingen, Germany

Key Points:

- We propose a hybrid modeling framework combining uncertainty-quantifying artificial neural networks with physical domain knowledge.
- Our method learns diffusion processes and discovers unknown sorption isotherms, while incorporating different boundary conditions.
- Excellent generalization is retained even when trained with sparse and noisy real-world data from a laboratory experiment.

Corresponding author: Timothy Praditia, timothy.praditia@iws.uni-stuttgart.de

Abstract

Improved understanding of complex hydrosystem processes is key to advance water resources research. Nevertheless, the conventional way of modeling these processes suffers from a high conceptual uncertainty, due to almost ubiquitous simplifying assumptions used in model parameterizations/closures. Machine learning (ML) models are considered as a potential alternative, but their generalization abilities remain limited. For example, they normally fail to predict across different boundary conditions. Moreover, as a black box, they do not add to our process understanding or to discover improved parameterizations/closures. To tackle this issue, we propose the hybrid modeling framework FINN (finite volume neural network). It merges existing numerical methods for partial differential equations (PDEs) with the learning abilities of artificial neural networks (ANNs). FINN is applied on discrete control volumes and learns components of the investigated system equations, such as numerical stencils, model parameters, and arbitrary closure/constitutive relations. Consequently, FINN yields highly interpretable results. To show this, we demonstrate FINN on a diffusion-sorption problem in clay. Results on numerically generated data show that FINN outperforms other ML models when tested under modified boundary conditions, and that it can successfully differentiate between the usual, known sorption isotherms. Moreover, we also equip FINN with uncertainty quantification methods to lay open the total uncertainty of scientific learning, and then apply it to a laboratory experiment. The results show that FINN performs better than calibrated PDE-based models as it is not restricted to choose among a limited set of sorption isotherms.

1 Introduction

Scientists and engineers have been trying to model physical phenomena occurring in nature for centuries. Among such phenomena, one of the most important yet challenging task is to calculate the transport of a quantity in space and in time through natural media. A few examples include: subsurface fluid flow modeling (e.g. Ghosh et al., 2020; T. Koch et al., 2021), climate modeling (e.g. Marchuk, 1974; IPCC, 2013), and diffusion-reaction modeling (e.g. Turing, 1952; Wei & Winter, 2017). Of course, contaminant transport and attenuation in water resources research also falls into this problem class. Problems of this type are usually described mathematically using partial differential equations (PDEs) because of their space and time dependencies.

However, despite promising development in computing power and availability of data, the behavior of several of these physical systems is still poorly understood (Winsberg, 2003). As such, simplifying assumptions are required to model parts of the processes that are either still unknown, too complicated, or act on scales much smaller than the one of interest. Concrete examples are the choice of sorption isotherms in diffusion-sorption problems (Limousin et al., 2007; Al-Ghouti & Da'ana, 2020), the relative permeability and saturation relationship in multiphase flow in porous media (K. Li & Horne, 2006; Moghadasi et al., 2015), and the reaction formulations in diffusion-reaction system (Klaasen & Troy, 1984; Allen & Cahn, 1979).

To address this issue in this work, we propose the finite volume neural network (FINN), which is a novel physics-aware ML modeling framework. As its name suggests, FINN combines the well-established numerical discretization strategy of the finite volume method (FVM) and the flexibility and learning ability of artificial neural networks (ANNs). Most importantly, this combination allows our method to explicitly and accurately learn parts of the unknown or poorly-understood processes mentioned above, while maintaining numerical stability, producing highly accurate predictions, and providing scientifically interpretable functions of interest.

There is a wide range of problems to which FINN can be applied. Here, we focus on one of them for the sake of demonstration, namely, diffusion and sorption of ground-

water contaminant in fully saturated clay. This process is relevant, e.g., in clay liners of landfills (Timms et al., 2018; Hendry et al., 2003) or in long-term tailing of groundwater pollution (Huang & Goltz, 2015; Johnson et al., 2003).

One particularly harmful contaminant is trichloroethylene (TCE), which is categorized as a carcinogenic (National Toxicology Program, US Department of Health and Human Services, 2021), yet it is still commonly used in industry (World Health Organization, Regional Office for Europe, 2000). TCE is particularly dangerous because it is a dense non-aqueous phase liquid (DNAPL) (Pankow & Cherry, 1996), meaning that it is denser than water and has a very low solubility in water. As a consequence, when TCE infiltrates into the subsurface, it migrates downwards until it reaches an impermeable barrier and forms a pool there, resulting in difficult remediation (e.g. J. Koch & Nowak, 2015; G. H. Brown et al., 2012). One of the most common impermeable barriers in the subsurface is a layer of clay. However, even though a layer of clay is impermeable, TCE can still diffuse into it and constantly contaminate the groundwater in the vicinity for a long period of time (e.g. Nowak & Guthke, 2016; Pankow & Cherry, 1996). It is accordingly necessary to build a model of such processes in order to predict the longevity of contamination, select remediation strategies and assess environmental and health risks.

Despite numerous works studying the diffusion-sorption process, there are yet many conceptual uncertainties associated in the modeling process, such as the choice among different isotherms that best describe the sorption behavior at hand (Limousin et al., 2007), the unknown parameters of those sorption isotherms (e.g. Nowak & Guthke, 2016), uncertain clay/soil parameters and effective diffusion coefficients of dissolved chemicals in water (Wilke & Chang, 1955; Hayduk & Laudie, 1974), as well as uncertain initial and boundary conditions that the model requires to be satisfied. As a consequence of these modeling uncertainties, we are faced with a model choice problem (Höge et al., 2019). Furthermore, all of the available models are inherently generated with simplifying assumptions to different extents, increasing the difficulty of the model choice problem even more. In the Bayesian world, this is known as the M-open problem (Höge et al., 2019). Thus, a more flexible way of modeling is needed, such that the unknown "true model" stands a better chance of being covered by the approach, as opposed to a discrete model selection method (i.e. choosing between different sorption isotherms). One promising solution is to implement a data-driven modeling strategy, but with smart structure and all possible hard knowledge (usually forming the core of a model) enforced by constraints, to learn these unknown relationships.

In order to approach the question of conceptual model learning, it is worth looking at the rapidly evolving field of machine learning (ML), which has revolutionized various domains including image and language processing (Krizhevsky et al., 2012; T. B. Brown et al., 2020). Recently, ML is also being applied to approximate physical processes, such as rigid body interactions, liquid propagation, or weather and sea-surface temperature prediction (Battaglia et al., 2016; De Bézenac et al., 2019; Rasp et al., 2020; Sanchez-Gonzalez et al., 2020; Espeholt et al., 2021; Lienen & Günnemann, 2022). The benefit and charm of applying ML models lies in their ability to learn an input-to-output mapping function without knowing any parts of the underlying process that describes the data (i.e. the "true model"). Furthermore, ML models also have potentials to learn more complicated functional relationships that are not addressed in the physical models due to limited computational power or lack of understanding of the modeled systems. In the following, we summarize the related works, separating them in non-physics-motivated (*pure ML*), *physics-motivated*, and *physics-aware ML* models.

An exemplary *pure ML* method for processing spatiotemporal data—as represented by the PDE in equation (1)—is the temporal convolution network (TCN) proposed by Lea et al. (2016), an ANN that performs convolution operations along both space and time dimension. TCNs are particularly efficient, since convolution operations can be implemented highly parallel on modern GPU hardware. They have also been proven suc-

cessful in various classification tasks (Kalchbrenner et al., 2016; Bai et al., 2018). On the other hand, their applicability as autoregressive models in generative forecasting tasks is limited (Almqvist, 2019; Karlbauer et al., 2020). ConvLSTM (Shi et al., 2015) is a variant of recurrent neural networks that processes temporal data points sequentially and is therefore slower than fully parallel operations. It can, however, aggregate and conserve any past information in a latent state, which implements a memory. In contrast, the temporal horizon of TCNs is exclusively limited to the receptive field, that is the number of time steps affected by the TCN filters. Moreover, autoregressive RNNs such as ConvLSTM are by training optimized towards maintaining stable predictions within a recurrent loop, which is why they are superior on related tasks (Almqvist, 2019; Karlbauer et al., 2020). However, the freedom of the pure ML models has several limitations to learn an unconstrained function that should reflect a physical process. First, they typically depend on large amounts of data in order to learn a useful mapping. Second, they can be expected to behave adequately only within the range of the data they have been trained on. And third, they can produce physically implausible predictions.

These limitations can be addressed by incorporating structural physical knowledge to formulate *physics-motivated ML* models, using the form of (relational) inductive biases (Battaglia et al., 2018). DISTANA (Karlbauer et al., 2019), for example, shares similarities with ConvLSTM, albeit with advanced and physically-motivated lateral information flow between neighboring simulated control volumes. An alternative approach is to learn (fourier) neural operators (FNO)—either in frequency or time space (Z. Li et al., 2020a, 2020b). By design, these methods are implemented to specifically learn PDEs from data, but are not guided or constrained by physical principles that would be already known.

Recently, much effort was directed to finding reasonable ways of connecting the learning abilities of ANNs not only with structural, but even with functional knowledge from physics, resulting in *physics-aware ML* methods. The physics-informed neural network (PINN), for example, explicitly learns how to solve a *given* equation, such as Burgers' or Navier-Stokes in order to accelerate simulation (Raissi et al., 2019; Jin et al., 2021). PINN has also been applied specifically to solve subsurface fluid flow problems (Tartakovsky et al., 2020) and perform data assimilation for parameter estimation, accounting for multiple physical processes (Q. He et al., 2020). Other methods do not depend on receiving the underlying equation but approximate it implicitly via the data. The weaker physics constraints are either implemented by means of convolution-like operators that only represent derivatives up to a particular degree, e.g. PDE-Net (Long et al., 2018), PhyDNet (consisting of a data-driven ConvLSTM and a physics-constrained path) from Guen and Thome (2020), or SIREN (Sitzmann et al., 2020); or by directly learning the transition function $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ (e.g. in form of a vector field) that maps the d-dimensional observation in frame t to the succeeding frame $t+1$ (Tran & Ward, 2017; De Bézenac et al., 2019). More recently, graph-based approaches are formulated by Seo et al. (2019); Salehi and Giannacopoulos (2021) to explicitly consider e.g. differences between neighboring control volumes on spatially irregularly distributed data.

Nevertheless, a common downside of all these approaches is the missing facility to include explicit physical knowledge—such as the *structure* of a particular PDE—into the learning process. In contrast, and similar to our work, Bar-Sinai et al. (2019); Kochkov et al. (2021); Zhuang et al. (2021) propose to learn selected parts of ODEs, but focus more on accelerating supersampling procedures and less on predictive, explorative, and explainability tasks. APHYNITY (Yin et al., 2020) represents an alternative approach, where traditional physical models are augmented by ML methods, effectively learning to minimize the residual between an explicitly stated physical model and the observation. A survey of methods that combine physics with ML has been proposed by Karniadakis et al. (2021) and an extensive collection is maintained by Thuerey et al. (2021).

Despite these exciting developments in the areas of physics-motivated and physics-aware ML modeling, there are still important issues that are yet to be addressed. More specifically, building PINN models requires the complete knowledge of the modeled systems, including the aforementioned closure/constitutive relationships, which are usually the main source of uncertainty in the modeling process. As a consequence, PINN can be trained based on incorrect equations, if the assumptions chosen are also erroneous. Additionally, most, if not all, of the spatiotemporal ML models adopt convolutional operations to process the spatial correlation between data points. Convolutional operations, however, can only pad constant values on the domain boundaries. Therefore, such ML models have no means of sufficient boundary condition implementation if the boundary condition is not constant. In other words, they are not able to properly incorporate boundary conditions that depend on derivatives such as the Neumann or Cauchy boundary conditions. Furthermore, the generalization ability of all the existing ML models is still questionable, especially when confronted with different initial or boundary conditions. Often, another training process has to be initialized for different initial and boundary conditions, which requires a lot of observation data that is often expensive and difficult to obtain. In short, the existing physics-motivated and physics-aware ML models are either too restricted by the physical knowledge or too lenient so that they learn relationships that do not exist (i.e. they overfit).

Another crucial drawback of most ML models is the lack of practical uncertainty quantification (UQ) applications, despite the availability of numerous theoretical foundations (Jospin et al., 2022). This is mainly caused by computational challenges of existing UQ algorithms for large ML models. When dealing with real-world applications, however, UQ is critical. For example, when performing a risk assessment about a poorly-understood system based on uncertain models trained on uncertain (noisy) and sparse data (Wöhling et al., 2015; Nowak & Guthke, 2016; Xu et al., 2020). Moreover, fundamental scientific work needs hypothesis-testable models, and this is strongly supported by models with quantifiable uncertainty. Consequently, it is important to design a model with few parameters and an interpretable structure to enable feasible implementations of available UQ algorithms.

The goal of this work, therefore, is to provide a framework that merges physics-aware ML models with well-selected structures known from numerical solution. This can facilitate scientists to produce better models that balance well between the flexibility of learning data-driven models and the existing scientific knowledge. It should be emphasized that this work is not intended to develop a faster and more efficient surrogate model in place of any existing physical model, but rather to learn unknown constituents of the PDE used to model the (not yet fully understood) physical processes. The named TCE problem is a very representative problem from a broader class where the model structure is only partially known.

In a wider sense, we are interested in environmental problems where fundamental parts of the governing equations (or principles used in their derivation) are accepted as "known truth", but where other parts are uncertain or even unknown, and often treated with assumptions, closures, or other approximations. In our TCE example, it is a sorption isotherm that is most uncertain. Other instances of the same problem class are water retention curves in the Richards equation; capillary pressure and saturation relations, relative permeability and saturation relations, or expressions for hysteresis and dynamic effects in multiphase flow in porous media; turbulence closures in rivers, pipe flow or atmospheric flow; scale-dependent expressions for dispersion in heterogeneous porous media; effective turnover rate models in chemical or microbiological reactive transport problems. When these uncertain or unknown relationships are successfully learned from data, exceptional generalization ability and highly accurate predictions will follow consequently.

For this purpose, we introduce the finite volume neural network (FINN), a hybrid method that combines numerical methods (FVM and ordinary differential equation solvers)

with physical knowledge in the form of a physics-constraint network architecture, which learns unknown constituents of an uncertain set of governing equations. FINN thus becomes able to jointly learn unknown constitutive/closure relationships, PDE terms, and parameters from data. The benefits of our hybrid model are an excellent generalization ability beyond sparse training data, a proper treatment of different boundary conditions other than a constant Dirichlet condition, and an explainable model. Furthermore, with the adoption of the FVM structure and physical constraints, FINN utilizes as much existing modeling knowledge as possible. Additionally, we formulate FINN to provide an uncertainty estimate over its learned constituents when predicting a real-world soil contamination problem, and affirm FINN's advantages over a calibrated, conventional PDE-based model.

2 Methodological Background

In this section, we derive the methodological framework of this paper on the basis of an experimental reference setup and the involved equation form which will be learned by FINN. Then, we provide some background on the FVM discretization method to solve PDEs and on neural ODE as a differentiable numerical integrator as it serves the conceptual basis for FINN. Finally, we summarize a selection of UQ methods that can be implemented in FINN.

2.1 Experimental Setup and Governing Equations

The diffusion process in general is governed by a PDE of second-order in space and first-order in time:

$$\frac{\partial c}{\partial t} = \nabla \cdot (D(c)\nabla c) + q(c), \tag{1}$$

where c is the variable of interest, namely, the contaminant concentration in this study, t is time, D is a diffusion coefficient that can be a dependent variable on c, and q is the source/sink term, for example if there is a reaction or there is an addition/extraction of the contaminant to/from the domain of interest. The diffusion through clay, however, might be hampered by the presence of organic matter inside the clay that sorbs the TCE, therefore slowing down the diffusion process (Parker et al., 2004). Therefore, the sorption process has to be taken into account in the governing PDE as well, by including an additional variable in form of the retardation factor. The retardation factor is a variable that is possibly dependent on the contaminant concentration, and it defines the degree to which the diffusion process is hindered by the sorption process. The resulting diffusion-sorption equation can be solved with various numerical discretization methods, one of the most popular being FVM due to its conservation property (Moukalled et al., 2016).

This diffusion-sorption process of TCE as contaminant in water-saturated clay was studied in a laboratory experiment (Nowak, 2000), and its setup is adopted in the numerical experiments performed in this work. A clay sample with a radius of 2.54 cm (1 inch) is placed inside a stainless steel tube of length L. On the upper end of the sample, pure-phase TCE is injected through an inlet valve. There, it forms a pool, from which it can dissolve into the clay and thus installs a constant concentration condition at the upper end of the clay cylinder. The bottom end of the sample is flushed with clean water below the clay cylinder at a constant flow rate, in order to enable measurement of the dissolved TCE concentration (as it diffused downward through the clay) at various time intervals. At the end of the experiment, the clay sample is cut into horizontal slices to allow measurement of the total TCE concentration (i.e. TCE dissolved in the water and sorbed in the clay) within the cylinder. In short, there are two main variables of interest in the experiment, namely the dissolved concentration and the total concentration of the contaminant. More details of the experiment can be found in Nowak (2000); Parker et al. (2004); Nowak and Guthke (2016).

Assuming that the clay sample is homogeneous, the governing diffusion-sorption equation can be simplified into a one-dimensional system. Mathematically, the governing PDE used to calculate the dissolved concentration could be written as (Nowak & Guthke, 2016)

$$\frac{\partial c}{\partial t} = \frac{D}{R(c)} \frac{\partial^2 c}{\partial x^2}, \tag{2}$$

where c is the dissolved TCE concentration, t is time, x is distance along the axis of the cylinder, D is the effective diffusion coefficient, and R is the retardation factor, which is a function of c. As a consequence, the diffusivity (i.e. D/R) is also dependent on c.

Because the upper end of the sample is in equilibrium with the TCE in pure-phase, a Dirichlet boundary condition is applied:

$$c|_{x=0} = c_{\text{sol}} \quad \forall t : 0 \leq t \leq T, \tag{3}$$

where c_{sol} is the solubility limit of the TCE in water, and T is the experiment time. On the bottom end of the sample, the TCE concentration is not constant, and therefore, a Cauchy condition is required to model the flow-dependent boundary condition as a result of the flushing with water:

$$c|_{x=L} = \frac{D}{Q} \frac{\partial c}{\partial x} \quad \forall t : 0 \leq t \leq T, \tag{4}$$

where Q is the water flow rate at the bottom of the clay sample. The clay sample is initially clean of any contamination, resulting in an initial condition of:

$$c|_{t=0} = 0 \quad \forall x : 0 \leq x \leq L. \tag{5}$$

To derive a possible equation for the total concentration, the general definition of retardation factor R is required. The retardation factor R is defined as the ratio of sorbed to non-sorbed material as following (e.g. Fetter, 1999; Nowak & Guthke, 2016):

$$R = \frac{1}{\phi} \frac{\partial c_t}{\partial c}, \tag{6}$$

where ϕ is the porosity of the porous medium and c_t is total contaminant concentration, i.e. the contaminant concentration dissolved in the fluid and sorbed in the solid phase. By substituting equation (6) into equation (2), the equation to calculate the total contaminant concentration c_t can be written as:

$$\frac{\partial c_t}{\partial t} = D\phi \frac{\partial^2 c}{\partial x^2}. \tag{7}$$

Furthermore, to solve equation (2), the retardation factor is typically also defined using a sorption isotherm (e.g. Limousin et al., 2007), which is a parametric model. Three of the most commonly used isotherms are linear, Freundlich, and Langmuir. These isotherms define the retardation factor differently:

$$R_l = 1 + \frac{1-\phi}{\phi} \rho_s K_d, \tag{8}$$

$$R_F = 1 + \frac{1-\phi}{\phi} \rho_s K_f n_f c^{n_f - 1}, \tag{9}$$

$$R_L = 1 + \frac{1-\phi}{\phi} \rho_s \frac{s_{\max} K}{(c+K)^2}, \tag{10}$$

where equations (8), (9), and (10) describe the retardation factor formulation based on the linear, Freundlich, and Langmuir isotherm, respectively. Here, ρ_s is the bulk density of the porous medium, K_d is the linear isotherm parameter, K_f is the Freundlich

isotherm parameter, n_f is the Freundlich exponent, $s_{\max}$ is the maximum sorption capacity of the solid phase, and K is the half-saturation value.

Traditionally, the retardation factors reported in equations (8), (9), and (10) would lead to three different discrete models, one for each sorption isotherm. However, FINN allows us to define the retardation factor as a flexible function that is learned from data to best support the approximation of the overall process without constraining the model by a possibly inaccurate assumption.

2.2 Numerical Solution

Obtaining an analytical solution is impossible for many PDEs. Consequently, it is common to resort to numerical methods to solve it. The most popular numerical methods are the finite difference method (FDM), which approximates the derivatives based on Taylor's expansion (Morton & Mayers, 1994); the finite element method (FEM), which reformulates the PDE in a weak form and interpolates the solution through a function with limited element support (Logan, 1992); and FVM, which approximates the solution using a volume integral combined with the Gauss' divergence theorem (Moukalled et al., 2016).

The FVM derivation of the PDE is based on conservation laws and is the closest to physics compared to the other discretization methods. To be more specific, the applied divergence theorem leads to the right-hand side of equation (13), which now represents the flux exchanges between any control volume and its neighboring volumes. This ensures that conservation is not violated, meaning that the flux entering a control volume should be exactly the same as the flux leaving the control volume, given that the variable of interest (i.e. the concentration or quantity c) does not change over time. On the other hand, FEM does not guarantee this conservation property. Moreover, FVM allows straightforward implementation of the boundary conditions without approximation. Due to these reasons, we choose to specifically adopt FVM in this work. Note that, when the domain is discretized using a Cartesian grid, then the FVM and FDM are identical.

Following the FVM concept, the spatial domain is discretized into a number of N control volumes (cells). For each control volume, a volume integral is applied to equation (2), resulting in

$$\int_{v_i} \frac{\partial c}{\partial t} dv = \int_{v_i} \frac{D}{R(c)} \frac{\partial^2 c}{\partial x^2} dv, \tag{11}$$

where v_i is the volume, and the subscript $i = 1, \ldots, N$ denotes a specific control volume i. Since the right-hand side has a divergence term, the Gauss' divergence theorem needs to be applied (Arfken et al., 2013), leading to a surface integral over the enclosing control volume surfaces/boundaries, and resulting in the equation

$$\int_{v_i} \frac{\partial c}{\partial t} dv = \oint_{\omega \subseteq \Omega} \left(\frac{D}{R(c)} \frac{\partial^2 u}{\partial x^2} \right) \cdot \hat{n} \, d\Gamma, \tag{12}$$

where ω is a continuous surface element and $\hat{n}$ is the unit normal vector pointing outwards of ω. Furthermore, ω is a subset of all surfaces Ω enclosing the control volume i and Γ is a continuous variable along ω. Applying the surface integral in equation (12) allows flux evaluation at each enclosing control volume surface. As a result, a spatially discrete formulation of the PDE for control volume i using a Cartesian grid leads to the following:

$$\frac{\partial c_i}{\partial t} v_i = A_{i-1} \frac{D_i}{R(c_i)} \frac{c_{i-1} - c_i}{\Delta x} - A_{i+1} \frac{D_i}{R(c_i)} \frac{c_i - c_{i+1}}{\Delta x}, \tag{13}$$

where A_{i-1} and A_{i+1} are the left and right cross-sectional surface areas of control volume i, respectively. The equation is spatially discrete, and thus uses Δx instead, which is the length of the control volume.

For each $i = 1, \ldots, N$, equations (13) form a coupled set of ordinary differential equations (ODEs). The ODEs in the system of equations (13) are coupled through their connections with their respective neighbors ($i-1$ and $i+1$). To derive equation (13) into a spatially and temporally discrete equation, a temporal discretization is required. The simplest temporal discretization scheme is the Euler method (e.g. Butcher, 2008), which is a first-order time integration method. The Euler method itself is categorized into explicit and implicit schemes. In the explicit scheme, the time derivative function dc/dt is defined with the variable c at the current time step t, whereas in the implicit scheme, it is defined with the variable c at the subsequent time step $t+1$. Applying the explicit Euler method to equation (13) leads to

$$\frac{c_i^{t+1} - c_i^t}{\Delta t} v_i = A_{i-1} \frac{D_i}{R(c_i^t)} \frac{c_{i-i}^t - c_i^t}{\Delta x} - A_{i+1} \frac{D_i}{R(c_i^t)} \frac{c_i^t - c_{i+1}^t}{\Delta x}, \tag{14}$$

and applying the implicit Euler method yields

$$\frac{c_i^{t+1} - c_i^t}{\Delta t} v_i = A_{i-1} \frac{D_i}{R(c_i^{t+1})} \frac{c_{i-i}^{t+1} - c_i^{t+1}}{\Delta x} - A_{i+1} \frac{D_i}{R(c_i^{t+1})} \frac{c_i^{t+1} - c_{i+1}^{t+1}}{\Delta x}, \tag{15}$$

where the superscript t denotes the time discretization and Δt is the corresponding time step. The same discretization strategy also applies to Equation (7). As can be inferred from both equations, the implementation of the explicit method is simpler than the implicit method, because the value of c_i^{t+1} is still unknown in time step t. Furthermore, because we intend to combine numerical methods with ANNs—which fundamentally belong to the class of explicit methods—from here on we will use the explicit scheme and, when possible, drop the superscript t for clarity.

Even though the explicit method is more convenient to implement, it suffers from numerical instability. To be more specific, the size of the time step Δt has to be chosen carefully such that it does not surpass the time at which the quantity c is moving from one control volume to the other. This requirement can be controlled, e.g., by using the Courant–Friedrichs–Lewy (CFL) condition (Courant et al., 1967; Isaacson & Keller, 1994). According to CFL, a finer spatial discretization (i.e. smaller Δx) requires a smaller temporal discretization Δt. In some cases, this condition becomes very limiting when the required Δt becomes too small, leading to substantially inefficient computation.

2.3 Neural ODE

Implementing the explicit integration method from section 2.2 with an unregularized ANN will likely lead to numerical instability, as the ANN might easily enter unstable regimes (e.g. no control of the CFL condition). To mitigate this problem, a higher-order ODE integration method, such as Runge-Kutta method (Runge, 1895; Kutta, 1901), can be used, in combination with an adaptive time stepping capability, to maintain both the numerical stability and the accuracy of the integration. Such ODE solvers, however, must be differentiable to propagate an error signal in order to adapt the parameters of an ANN that will be used to represent the ODE. For a comprehensive introduction to ANNs, we refer the readers to Goodfellow et al. (2016).

Such a *fully differentiable* and ANN-based ODE solver, called Neural ODE (NODE), has recently been proposed by Chen et al. (2018). In short, NODE assumes an ANN f_θ with parameters θ to compute the change of a state vector $\mathbf{c}^t$ over time. NODE parameterizes the change of $\mathbf{c}$ over time by treating f_θ as a time-continuous function, such that

$$\frac{\partial \mathbf{c}(t)}{\partial t} = f_\theta(\mathbf{c}(t), t). \tag{16}$$

Accordingly, $f(\cdot)$ is an ANN that learns and represents the system dynamics, i.e. the derivative of the variable of interest with respect to time, which directly corresponds to the

form of equation (13). Then, the quantity of $\mathbf{c}$ in the next time step could, in principle, be computed using an explicit Euler discretization:

$$\mathbf{c}^{t+1} = \mathbf{c}^t + f_\theta(\mathbf{c}^t). \tag{17}$$

Formulating a dynamic system with Equation (17) has been proposed in the Deep Residual Learning (ResNet) architecture (K. He et al., 2016). This approach has been shown to improve model training because it can learn particular functions, such as the identity function, better and because it minimizes the vanishing gradient problem (Hochreiter et al., 2001). In the NODE framework, however, the temporal discretization of equation (16) occurs after the forward propagation of the ANN and with a higher-order scheme, rather than learning the discretization form as in equation (17).

In NODE, to integrate from $\mathbf{c}^t$ to $\mathbf{c}^{t+1}$, f_θ is optimized end-to-end in the overall training process. This leads to better accuracy and efficiency, as well as to allow adaptive time stepping strategy for better numerical stability (Chen et al., 2018). Note that this approach differs from the Elman network, i.e. the traditional recurrent neural network (Elman, 1990). The Elman network uses the ANN as a function to predict $\mathbf{c}^{t+1}$ directly from $\mathbf{c}^t$, i.e. without the appearance of $\mathbf{c}^t$ in equation (17). NODE plays a fundamental role in our model, as detailed in section 3.

2.4 Uncertainty Quantification Methods

In this work, we perform uncertainty quantification over the model parameters and all learned constituents, such as the retardation factor function $R(c)$. One of the most straightforward uncertainty quantification methods on ANNs is the Bayes-by-backprop method (Blundell et al., 2015), which parameterizes the variational posterior using the mean μ and standard deviation σ of the model parameters θ (i.e. weights and biases). In short, for each training iteration, the model parameters are sampled based on

$$\theta = \mu + \log(1 + \exp(\sigma)) \cdot \epsilon, \tag{18}$$

where $\epsilon \sim \mathcal{N}(0, I)$. The goal of the training is then to find the values of μ and σ that minimize the Kullback-Leibler divergence (Joyce, 2011) between the variational posterior $q(\theta)$ and the true posterior $\pi(\theta)$, reformulated as

$$\mathcal{L} = KL[q(\theta)||p(\theta)] - \mathbb{E}_{q(\theta)}[\log p(D|\theta)], \tag{19}$$

where $p(\theta)$ is the prior knowledge on the model parameters, and $p(D|\theta)$ is the probability of observing the data D given the model parameters θ.

The Bayes-by-backprop approach, however, assumes independent Gaussian distributions to define the model parameters, which is an oversimplification of the actual joint posterior distribution (Blei et al., 2017). In contrast, Markov chain Monte Carlo (MCMC, Bardenet et al. (2017)) provides a sampling of model parameters from the exact posterior distribution (Jospin et al., 2022). The general MCMC algorithm is summarized in Algorithm 1.

The proposal of drawing θ_t and the transition/proposal distribution Q are defined respectively for the random walk Metropolis-Hastings (MH, Chib and Greenberg (1995)), Metropolis-adjusted Langevin algorithm (MALA, Dwivedi et al. (2019)), and Barker proposal (Barker, Livingstone and Zanella (2019)), as:

$$\theta_t = \theta^{(i)} + h \cdot \epsilon, \qquad Q(\theta_t|\theta^{(i)}) = \exp(-||\theta_t - \theta^{(i)}||_2^2/h^2), \tag{20}$$

$$\theta_t = \theta^{(i)} + h\nabla\pi(\theta^{(i)}) + \sqrt{2h}\epsilon, \quad Q(\theta_t|\theta^{(i)}) = \exp(-||\theta_t - \theta^{(i)} - h\nabla\pi(\theta^{(i)})||_2^2/4h), \tag{21}$$

$$\theta_t = \theta^{(i)} + b \cdot \epsilon, \qquad Q(\theta_t|\theta^{(i)}) = \frac{1}{1 + \exp(-(\theta_t - \theta^{(i)})^T \nabla \log \pi(\theta^{(i)}))}. \tag{22}$$

Algorithm 1 General MCMC algorithm

Require: Initial parameter values $\theta^{(0)}$
  Set $i = 0$
  while $i < N$ **do**
    Draw θ_t given $\theta^{(i)}$
    Calculate acceptance probability $\alpha(\theta_t|\theta^{(i)}) = \min\left(1, \frac{\pi(\theta_t)Q(\theta^{(i)}|\theta_t)}{\pi(\theta^{(i)})Q(\theta_t|\theta^{(i)})}\right)$
    Draw a random number $u \sim \mathcal{U}[0, 1]$
    if $\alpha(\theta_t|\theta^{(i)}) > u$ **then**
      $\theta^{(i+1)} \leftarrow \theta_t$
    else
      $\theta^{(i+1)} \leftarrow \theta^{(i)}$
    end if
    $i \leftarrow i + 1$
  end while

Here, θ_t is the proposed sample, $\theta^{(i)}$ is the sample from iteration i, h is the step size, π is the posterior of the model parameters, and $\epsilon \sim \mathcal{N}(0, I)$. For Barker specifically, $b = 1$ with the probability $p = 1/(1 + \exp(-\epsilon\nabla\pi(\theta^{(i)})))$, and $b = -1$ otherwise. It is also important to note that both MALA and Barker utilize the gradient information provided by the automatic differentiation tools available in various ML libraries, including PyTorch (Paszke et al., 2019) that is used in this work.

3 Finite Volume Neural Network

In this section, we introduce the finite volume neural network (FINN)[1] framework by providing derivations and explanations on how FINN relates to the concepts from the previous section. FINN is designed to explicitly learn the individual components of the PDE using dedicated modules—defined as nonlinear ANN layers—in a compositional manner (Lake et al., 2017; Battaglia et al., 2018; Lake, 2019; Karlbauer et al., 2022). These modules are connected to effectively represent the PDE of interest (see Figure 1). Although the model has a general from, it can be set up and interconnected individually to form a specific architecture that is explicitly motivated and inspired by the physical equation that would typically be assumed to govern the modeled system at hand, i.e. the diffusion-sorption equation (2) in this work. More specifically, FINN combines the knowledge-based structure of the PDE core elements with the FVM discretization as described in section 2.2 to obtain a set of ODEs. Then, it adopts the Neural ODE method as described in section 2.3 for the time integration and learning of the PDE constituents.

As outlined in section 2.2, a PDE describes the change of a quantity at a local position under influence of its direct neighbors. Accordingly, we propose to model the adjacent flux exchange—see equation (13)—by so called flux kernels $\mathcal{F}_i$. They learn to represent the quantity entering and leaving control volume i from left and right (in the one-dimensional case). These flux kernels approximate the surface integral for each control volume i as written in equation (12), and therefore are mathematically written as

$$\mathcal{F}_i = \sum_{j=1}^{M_i} f_j \approx \oint_{\omega \subseteq \Omega} \left(D(c) \frac{\partial^2 c}{\partial x^2} \right) \cdot \hat{n} \, d\gamma, \tag{23}$$

[1] FINN is implemented in `Python 3.8.5`, using `PyTorch 1.11.0`. The code is available online for download at `https://github.com/CognitiveModeling/finn`.

where M_i is the number of discrete surface elements of control volume i and f_j is the subkernel calculated at each surface element. For one-dimensional cases, $M_i = 2$ and therefore, each flux kernel is supported by 2 subkernels $\mathcal{F}_i = \{f_{i-}, f_{i+}\}$.

The flux kernels $\mathcal{F}_i$ are provided with c_i and c_{i-1} or c_{i+1} as the inputs for f_{i-} or f_{i+}, respectively. Each subkernel consists of two modules that are formulated as neural network layers. On the one hand, $\varphi_\mathcal{N}$ is a linear layer to approximate the FVM stencil to represent the contribution of each neighboring control volume and the direction of the flux exchange. Hence, the output of $\varphi_\mathcal{N}$ is conceptually supposed to become $\partial^2 c/\partial x^2$, that is

$$\varphi_{\mathcal{N}_{i-}}(c_i, c_{i-1}) + \varphi_{\mathcal{N}_{i+}}(c_i, c_{i+1}) \approx \frac{\partial^2 c_i}{\partial x^2}. \tag{24}$$

Note that $\varphi_{\mathcal{N}_{i-}}$ and $\varphi_{\mathcal{N}_{i+}}$ share weights and thus are represented by one and the same network. Ideally, in a system with Fickian diffusion and mass conservation fulfilled, the parameters of $\varphi_\mathcal{N}$ should be $[-1, 1]$ with respect to $[c_i, c_{i-1}]$ and $[c_i, c_{i+1}]$. When f_{i-} and f_{i+} are combined, i.e. in equation (24), the coefficients become $[1, -2, 1]$ with respect to $[c_{i-1}, c_i, c_{i+1}]$. This follows from the central discretization scheme (Fornberg, 1988) of the second-order spatial derivative as

$$\frac{\partial^2 c}{\partial x^2} \approx \frac{(\partial c/\partial x)|_{i-} - (\partial c/\partial x)|_{i+}}{\Delta x}, \tag{25}$$

with $(\partial c/\partial x)|_{i-} \approx (c_{i-1} - c_i)/\Delta x$ and $(\partial c/\partial x)|_{i+} \approx (c_i - c_{i+1})/\Delta x$. As a result,

$$\frac{\partial^2 c}{\partial x^2} \approx \frac{(c_{i-1} - c_i) - (c_i - c_{i+1})}{\Delta x^2} = \frac{c_{i-1} - 2c_i + c_{i+1}}{\Delta x^2}. \tag{26}$$

The module $\varphi_\mathcal{D}$, on the other hand, is responsible to account for the (variable-dependent, possibly nonlinear) diffusion coefficient, thus

$$\varphi_\mathcal{D}(c_i) \approx D(c_i), \tag{27}$$

if the diffusion coefficient D depends on c. Otherwise, $\varphi_\mathcal{D}$ is a scalar value $\varphi_\mathcal{D} \equiv D$, which can also be set as a learnable parameter. For our exemplary diffusion-sorption case, $\varphi_\mathcal{D} \approx D/R(c)$ for the dissolved concentration, i.e. equation (2), and $\varphi_\mathcal{D} \approx D$ for the total concentration, i.e. equation (7). Because the diffusion coefficient D can be learned from the total concentration, the retardation factor $R(c)$ can also be extracted from the learned module $\varphi_\mathcal{D}$ in the dissolved concentration calculation.

Finally, subkernels f_{i-} and f_{i+} are calculated as a combination of both modules $\varphi_\mathcal{N}$ and $\varphi_\mathcal{D}$:

$$f_{i-} = \varphi_\mathcal{D}(c_i) \cdot \varphi_{\mathcal{N}_{i-}}(c_i, c_{i-1}), \tag{28}$$

$$f_{i+} = \varphi_\mathcal{D}(c_i) \cdot \varphi_{\mathcal{N}_{i+}}(c_i, c_{i+1}). \tag{29}$$

Performing calculations of flux kernels at the surface element additionally provides advantages for boundary condition treatment. In addition to Dirichlet, boundary condition types that are flux dependent, such as Neumann or Cauchy conditions (Cheng & Cheng, 2005), can easily be adopted. For Dirichlet boundary conditions, a constant value $c = c_b$ is set as the input c_{i-1} (for the flux kernel f_{i-}) or c_{i+1} (for f_{i+}) at the corresponding boundaries. For a Neumann boundary condition ν, the output of the flux kernel f_{i-} or f_{i+} at the corresponding boundaries can be set to be equal to ν. For Cauchy boundary conditions, the solution-dependent derivative is calculated and set as c_{i-1} or c_{i+1} at the corresponding boundary.

Furthermore, we also introduce the state kernel $\mathcal{S}_i$ to model $\partial c/\partial t$. The state kernel $\mathcal{S}_i$ receives c_i of the associated control volume i, along with the output of f_{i-} and f_{i+} (the fluxes to/from each neighboring cell) as inputs:

$$\mathcal{S}_i(c_i, f_{i-}, f_{i+}) = \big(\Phi(c_i) + f_{i-} + f_{i+}\big) \approx \frac{\partial c_i}{\partial t}. \tag{30}$$

Optionally, the state kernel includes a neural network module $\Phi(\cdot)$, a function that is trained to model reaction terms related to the quantity c_i in control volume i. In the processes considered in this work, however, the only source of change comes from neighboring volumes. Particularly, there are no external effects that locally modify the quantity of interest (such as e.g. sun radiation would increase temperature locally without sensible or latent heat entering from adjacent volumes). Therefore, in this work, $\Phi(\cdot)$ is the identity function and, hence, $\mathcal{S}_i$ is only responsible for integrating information coming from neighboring cells.

Taking the bigger picture into account, all control volumes share the same kernels, and thus FINN is parsimonious, exploiting translation equivariance of physical laws. Flux kernels specifically are designed similarly to message passing neural networks (Gilmer et al., 2017; Brandstetter et al., 2022) to exploit PDE-type structural knowledge upon discretization. The main difference lies in the fact that all flux kernels are uniquely labeled with a physical meaning through derivation from FVM and NODE. Additionally, boundary conditions can be switched explicitly.

During training, FINN only receives the initial condition $c(x, t = 0)$ as input. This input is processed first by the flux kernels to calculate the flux exchanges between neighboring control volumes, and then by the state kernels to be integrated through all surface elements. The output of the state kernels are then fed into a differentiable ODE solver (within a Neural ODE framework) to be integrated in time to obtain the solution c at the subsequent time step. This solution is fed back into FINN, and the same operations are recurrently applied until the final simulation time $t = T$ is reached. This way, FINN uses a closed-loop setting to propagate the dynamics forward, leading to a more stable prediction during testing (Praditia et al., 2020). This workflow is visualized in Figure 1, where the black arrows depict the direction of the input processing, and the red arrows depict the direction of the error backpropagation during training. The dashed arrows depict the closed-loop feedback between subsequent time steps, as well as coupling between neighboring control volumes.

For this particular case, FINN is trained by minimizing the loss function, defined as

$$\mathcal{L} = \frac{1}{N_e} \sum_{i}^{N_e} (c_i - \hat{c}_i)^2 + \frac{1}{N_p} \sum_{j}^{N_p} E_{p,j}, \tag{31}$$

where N_e is the number of data points of $c(x, t)$, E_p is an additional physical constraint, and N_p is the number of data points used to calculate the physical constraint. The physical constraint E_p in our current example enforces that the retardation factor is a monotonous function of c, i.e. $R(c_i) \geq R(c_j)$ for all $c_i < c_j$ and positiveness of the diffusion coefficient, i.e. $D > 0$. To enforce these conditions, the ReLU operator is used. Other thinkable constraints could ab initio dictate mass conservation, energy balances or thermodynamic principles. Likewise, soft information (like staying close to an often-successful but not fully accurate law) could also be added. This loss function is also used as the negative log posterior $-\pi(\theta)$ in the uncertainty quantification, which will be discussed later.

4 Learning Experiments

To demonstrate FINN's capability, both synthetic and real-world datasets of the TCE diffusion-sorption process are used. Recall that the hypothesized advantages of FINN, by construction and due to the physics constraints, are little data requirements, robust generalization beyond the training data distribution (i.e. applicability to different boundary conditions), and high interpretability, allowing it to learn when known models fail.

First, we generate synthetic data based on the setup described in section 2.1 in order to perform an elaborated analysis in a controlled setting. By doing so, we will also

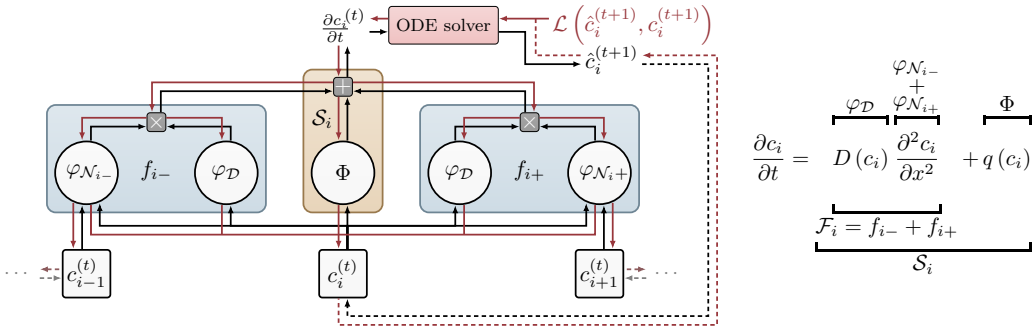

Figure 1: Schematic illustration of a flux kernel for one finite volume in FINN (top) and alignment of network modules with their corresponding parts in a PDE of interest (bottom). Black lines indicate forward information flow whereas red lines indicate gradient flow during backpropagation through time. Dashed lines indicate closed-loop feedback between subsequent time steps.

assess the performance of FINN against already existing pure deep learning models such as TCN and ConvLSTM, physics-motivated deep learning models, such as DISTANA and FNO, as well as physics-aware architectures, i.e. PINN and PhyDNet. Second, we also show that FINN is not only suitable for synthetic data, but also for real-world application by modeling real laboratory experimental data.

4.1 Synthetic Dataset

We simulate the experimental setup described in section 2.1 numerically and generate synthetic datasets solving the related (assumed-to-be-true for now) PDE. The numerical simulator is a simple finite difference code with explicit Euler, made available along with our code[2]. These synthetic datasets are generated using the three different sorption isotherms: linear, Freundlich, and Langmuir. The parameters for each isotherm are set so that they yield similar concentration distribution, thus we can show that our proposed method is able to distinguish various isotherms even with similar looking data. The parameter values are given in Table 1.

To compare the generalization ability of FINN to that of the aforementioned existing methods, for each data generated with different sorption isotherms, we define three different types of synthetic datasets: *train*, in-distribution test (*in-dis-test*), and out-of-distribution test (*out-dis-test*). The differences between these datasets lie in the time domain and boundary condition used. The *train* data is generated with $x = [0, 1]\,\mathrm{m}$, $t = [0, 2\,500]$ days, and $c_{\mathrm{sol}} = 1.0\,\mathrm{kg/m^3}$. The *in-dis-test* data is generated with $x = [0, 1]\,\mathrm{m}$, $t = [2\,500, 10\,000]$ days, and $c_{\mathrm{sol}} = 1.0\,\mathrm{kg/m^3}$. The *out-dis-test* data is generated with $x = [0, 1]\,\mathrm{m}$, $t = [0, 10\,000]$ days, and a modified upper boundary value of $c_{\mathrm{sol}} = 0.7\,\mathrm{kg/m^3}$. The simulation domain for all three types of data is discretized with $\Delta x = 0.04\,\mathrm{m}$ and $\Delta t = 5$ days. The *train* data, as its name suggests, is used to train the models. Both *in-dis-test* and *out-dis-test* are used to test the models. The difference is that *in-dis-test* data is generated with the same parameter as the *train* data, but extrapolated for a longer time span, whereas *out-dis-test* data is generated with a different boundary condition value, to test the models' generalization under a different situation than during training. A different *type* of boundary condition will be tested in section 4.2.

[2] `https://github.com/CognitiveModeling/finn`

4.1.1 State-of-the-Art Benchmark Models

In this work, we compare FINN's performance against other models that are capable of processing spatiotemporal data. These models are either pure ML models or models that possess a form of physical inductive bias.

Pure ML models. For the pure ML models, we choose **TCN** and **ConvLSTM**. **TCN** performs convolution operations over both the spatial and temporal domain (Lea et al., 2016), and it exploits the benefits of features such as dilated convolution to process a larger receptive field. In other words, the **TCN** structure allows for processing information contained in more distant preceding time steps. For **TCN**, the chosen structure has 2 input channels, a hidden layer with 32 channels, and 2 output channels.

ConvLSTM takes a more classical approach, which is to capitalize on the recurrent structure of the long short-term memory (LSTM) model to handle the temporal correlation of the data, and replaces the internal operations with convolutional operations to handle the spatial correlation of the data (Shi et al., 2015). For **ConvLSTM**, the chosen structure is 2 input and output channels, with a hidden layer containing 24 channels.

Physics-motivated ML models. The physics-motivated methods chosen as benchmark models in this work are namely **DISTANA**, **FNO**, and **CNN-NODE**. While **DISTANA** is similar to **ConvLSTM**, the main difference is that **DISTANA** propagates information laterally via additional latent feature maps (and not only by applying convolutions on the input). This can also be seen as an analogue to a flux exchange between neighboring control volumes—even though in **DISTANA**, the lateral latent information does not have any physical meaning. The lateral and dynamic input and output sizes of the **DISTANA** model are set to 1 and 2, respectively, while a hidden layer of size 16 is used.

CNN-NODE is a combination of a conventional convolutional network stem that is augmented by NODE and thus formulates an ablation of FINN to determine the relevance of FINN's modular network architecture. We use a three-layered, batch-normalized and tanh-activated convolution stem.

Table 1: Parameter values for synthetic data generation.

Parameter	Symbol	Unit	Value
Common parameters			
Effective diffusion coefficient	D	m^2/day	5.00×10^{-4}
Porosity	ϕ	-	0.29
Density	ρ_s	kg/m^3	2880
Linear isotherm			
Partitioning coefficient	K_d	m^3/kg	4.30×10^{-4}
Freundlich isotherm			
Freundlich's K	K_f	$(m^3/kg)^{n_f}$	3.50×10^{-4}
Freundlich exponent	n_f	-	0.87
Langmuir isotherm			
Half-concentration	K	kg/m^3	1.00
Sorption capacity	s_{max}	m^3/kg	5.90×10^{-4}

Due to a point-wise formulation (similarly to PINN), **FNO** approximates PDEs by learning a continuous mapping from space-time inputs to the desired outputs, i.e. $\mathbb{R}^{x \times t} \mapsto \mathbb{R}^d$, where x and t are space and time coordinates and d is the dimensionality of the target. However, FNO differs from PINN in two fundamental aspects: First, FNO learns the according mapping in frequency instead of in time domain by applying fast (inverse) Fourier transformations. Second, FNO learns purely from data and does not depend on explicit physical process knowledge. In our experiments, we apply the identical model architecture as suggested by Z. Li et al. (2020a).

Physics-aware ML models. In the class of physics-aware ML models, we chose **PINN** and **PhyDNet** as benchmark candidates. **PINN** is one of the pioneering physics-motivated ML models. It makes use of the capability of ANN to calculate analytical derivatives through backpropagation to approximate derivatives in the PDE (Raissi et al., 2019). For this problem, **PINN** is defined as a feedforward network with the size of [2, 20, 20, 20, 20, 20, 20, 20, 20, 2] (i.e. 2 input and output neurons, with 8 hidden layers, each containing 20 neurons),

PhyDNet consists of two main branches: one for calculating the physical component and the other for calculating the residual component of the data, assuming that the PDE does not fully describe the modeled system. The physical branch is inspired by the Kalman Filter, which is a data assimilation technique to recurrently update the model parameters based on observation (i.e. training) data. The residual branch adopts the **ConvLSTM** structure. With a specific condition of the **PhyDNet** structure, it reduces to a **PDE-Net** model (Guen & Thome, 2020). **PhyDNet** is defined with the PhyCell containing 32 input dimensions, 7 hidden dimensions, 1 hidden layer, and the ConvLSTM containing 32 input dimensions, 32 hidden dimensions, 1 hidden layer.

In the last class of physics-aware ML models, **FINN** is implemented with the use of the modules $\varphi_{\mathcal{N}}$ and $\varphi_{\mathcal{D}}$. Here, the module $\varphi_{\mathcal{N}}$ is defined as a linear layer that takes 2 inputs, namely the dissolved concentration c of two neighboring control volumes. For the dissolved concentration c, the module $\varphi_{\mathcal{D}}$ is defined as a feedforward network with the size of [1, 10, 20, 10, 1] that takes c as an input and outputs the retardation factor $R(c)$. For the total concentration c_t, $\varphi_{\mathcal{D}}$ is defined as a scalar parameter to learn the unknown diffusion coefficient D.

4.1.2 Benchmark Performance of ML models

All models are trained with the objective to minimize the deviation between the model predictions of c and c_t with the training data. They are trained until convergence using the L-BFGS optimizer (Malouf, 2002), except for PhyDNet and FNO, which are trained with the Adam optimizer (Kingma & Ba, 2015) and a learning rate of 1×10^{-3} due to stability issues when training with the L-BFGS optimizer. The L-BFGS optimizer is chosen because it is a quasi-Newton optimization algorithm, which means it uses an approximation of the second-order derivative (Hessian matrix). Second-order optimization algorithms are shown to be more effective in reaching the (local) optima (Kochenderfer & Wheeler, 2019). The pure ML models are trained on the first 400 time steps and validated on the remaining 100 time steps of the train data, applying early stopping (Goodfellow et al., 2016). Additionally, all models are trained with 10 different random initializations to learn about their consistency and to show better representation of each model's performance.

Table 2 shows the summary of all the trained models' performance. For each dataset and each model, the mean and standard deviation values of the prediction mean squared error (MSE) across the 10 different initializations are presented. A more detailed presentation of the MSE values for each random training initialization can be found in Appendix A. Note that PINN is not implemented for *out-dis-test* data. The reason behind this is that PINN learns the explicit relationship of the prediction as a function of x and

Table 2: Comparison of MSE and according standard deviation scores across ten repetitions between different deep learning (TCN and ConvLSTM), physics-motivated (DISTANA, CNN-NODE, and FNO), and physics-aware neural networks (PINN, PhyDNet, and FINN) methods on the different isotherms. Best results are reported in bold.

Iso.		Model	Dataset		
			Train	*In-dis-test*	*Out-dis-test*
Linear	Pure ML	TCN	$(2.4 \pm 3.1) \times 10^{-1}$	$(3.5 \pm 4.4) \times 10^{-1}$	$(2.9 \pm 3.1) \times 10^{-1}$
		ConvLSTM	$(3.7 \pm 3.9) \times 10^{-2}$	$(4.0 \pm 3.9) \times 10^{-2}$	$(5.3 \pm 4.9) \times 10^{-2}$
	Physics motiv.	DISTANA	$(2.8 \pm 6.5) \times 10^{-4}$	$(1.9 \pm 2.6) \times 10^{-3}$	$(3.9 \pm 2.3) \times 10^{-3}$
		CNN-NODE	$(2.1 \pm 3.2) \times 10^{-3}$	$(1.6 \pm 1.9) \times 10^{-1}$	$(1.5 \pm 1.8) \times 10^{-1}$
		FNO	$(7.6 \pm 3.3) \times 10^{-5}$	$(1.0 \pm 0.3) \times 10^{-3}$	$(1.9 \pm 0.4) \times 10^{-2}$
	Physics aware	PINN	$(6.3 \pm 11) \times 10^{-5}$	$(3.9 \pm 7.8) \times 10^{-3}$	-
		PhyDNet	$(3.3 \pm 1.5) \times 10^{-5}$	$(6.1 \pm 17) \times 10^{-3}$	$(1.6 \pm 1.0) \times 10^{-2}$
		FINN	$\mathbf{(2.1 \pm 1.5) \times 10^{-7}}$	$\mathbf{(2.7 \pm 1.9) \times 10^{-7}}$	$\mathbf{(1.8 \pm 1.3) \times 10^{-7}}$
Freundlich	Pure ML	TCN	$(1.1 \pm 3.5) \times 10^{-1}$	$(1.6 \pm 1.5) \times 10^{-1}$	$(1.3 \pm 1.3) \times 10^{-1}$
		ConvLSTM	$(1.9 \pm 2.8) \times 10^{-2}$	$(2.4 \pm 1.9) \times 10^{-2}$	$(4.3 \pm 3.8) \times 10^{-2}$
	Physics motiv.	DISTANA	$(8.1 \pm 7.0) \times 10^{-6}$	$(1.8 \pm 1.6) \times 10^{-4}$	$(1.5 \pm 1.4) \times 10^{-3}$
		CNN-NODE	$(4.3 \pm 8.9) \times 10^{-3}$	$(2.6 \pm 5.2) \times 10^{-1}$	$(2.2 \pm 4.3) \times 10^{-1}$
		FNO	$(6.7 \pm 17.6) \times 10^{-4}$	$(1.4 \pm 2.7) \times 10^{-3}$	$(1.4 \pm 0.5) \times 10^{-2}$
	Physics aware	PINN	$\mathbf{(4.3 \pm 2.4) \times 10^{-6}}$	$(9.7 \pm 16) \times 10^{-4}$	-
		PhyDNet	$(7.1 \pm 20) \times 10^{-4}$	$(2.2 \pm 3.7) \times 10^{-3}$	$(1.2 \pm 0.1) \times 10^{-2}$
		FINN	$(2.9 \pm 0.4) \times 10^{-5}$	$\mathbf{(2.7 \pm 0.4) \times 10^{-5}}$	$\mathbf{(2.3 \pm 0.3) \times 10^{-5}}$
Langmuir	Pure ML	TCN	$(1.3 \pm 0.6) \times 10^{-1}$	$(1.2 \pm 0.7) \times 10^{-1}$	$(1.5 \pm 0.5) \times 10^{-1}$
		ConvLSTM	$(3.9 \pm 3.4) \times 10^{-2}$	$(3.1 \pm 2.3) \times 10^{-2}$	$(6.2 \pm 4.4) \times 10^{-2}$
	Physics motiv.	DISTANA	$(2.3 \pm 2.6) \times 10^{-5}$	$(9.8 \pm 14) \times 10^{-4}$	$(3.3 \pm 3.5) \times 10^{-3}$
		CNN-NODE	$(1.8 \pm 3.1) \times 10^{-4}$	$(1.2 \pm 1.2) \times 10^{-1}$	$(9.7 \pm 10.7) \times 10^{-2}$
		FNO	$(3.5 \pm 7.2) \times 10^{-4}$	$(1.1 \pm 1.4) \times 10^{-3}$	$(1.7 \pm 0.7) \times 10^{-2}$
	Physics aware	PINN	$\mathbf{(3.3 \pm 8.9) \times 10^{-5}}$	$(6.4 \pm 17) \times 10^{-3}$	-
		PhyDNet	$(4.6 \pm 4.2) \times 10^{-5}$	$(1.3 \pm 1.4) \times 10^{-3}$	$(1.3 \pm 0.2) \times 10^{-2}$
		FINN	$(7.3 \pm 7.2) \times 10^{-5}$	$\mathbf{(7.9 \pm 7.8) \times 10^{-5}}$	$\mathbf{(6.1 \pm 6.1) \times 10^{-5}}$

t based on a specific initial and boundary condition implemented in the *train* data. When the boundary condition is changed to generate the *out-dis-test* data, this functional relationship no longer holds, and as such, PINN is no longer applicable. As a further comparison, we present the number of learnable parameters used by the different models in Table 3. To clarify, the benchmark does not include computational time, since the main purpose of our work is not to build a fast surrogate model, but rather to learn unknown functions in an interpretable fashion from the data and to generalize well to unseen data with different initial and boundary conditions.

As shown in Table 2, the pure ML models perform very poorly even during training, failing to capture/approximate the system's behavior. The physics-motivated and physics-aware models, on the other hand, perform comparably during training, showing adequate learning (except for CNN-NODE). Also, note that the pure ML models in general require more parameters compared to the physics-motivated and physics-aware models, as shown in Table 3. Even then, their performance is still not comparable. However, PhyDNet is an exception, because most of its parameters originate from the ConvLSTM branch, which is the data-driven part and not the physics-aware part of PhyDNet. During training, FINN achieves the lowest prediction error for the data generated with the

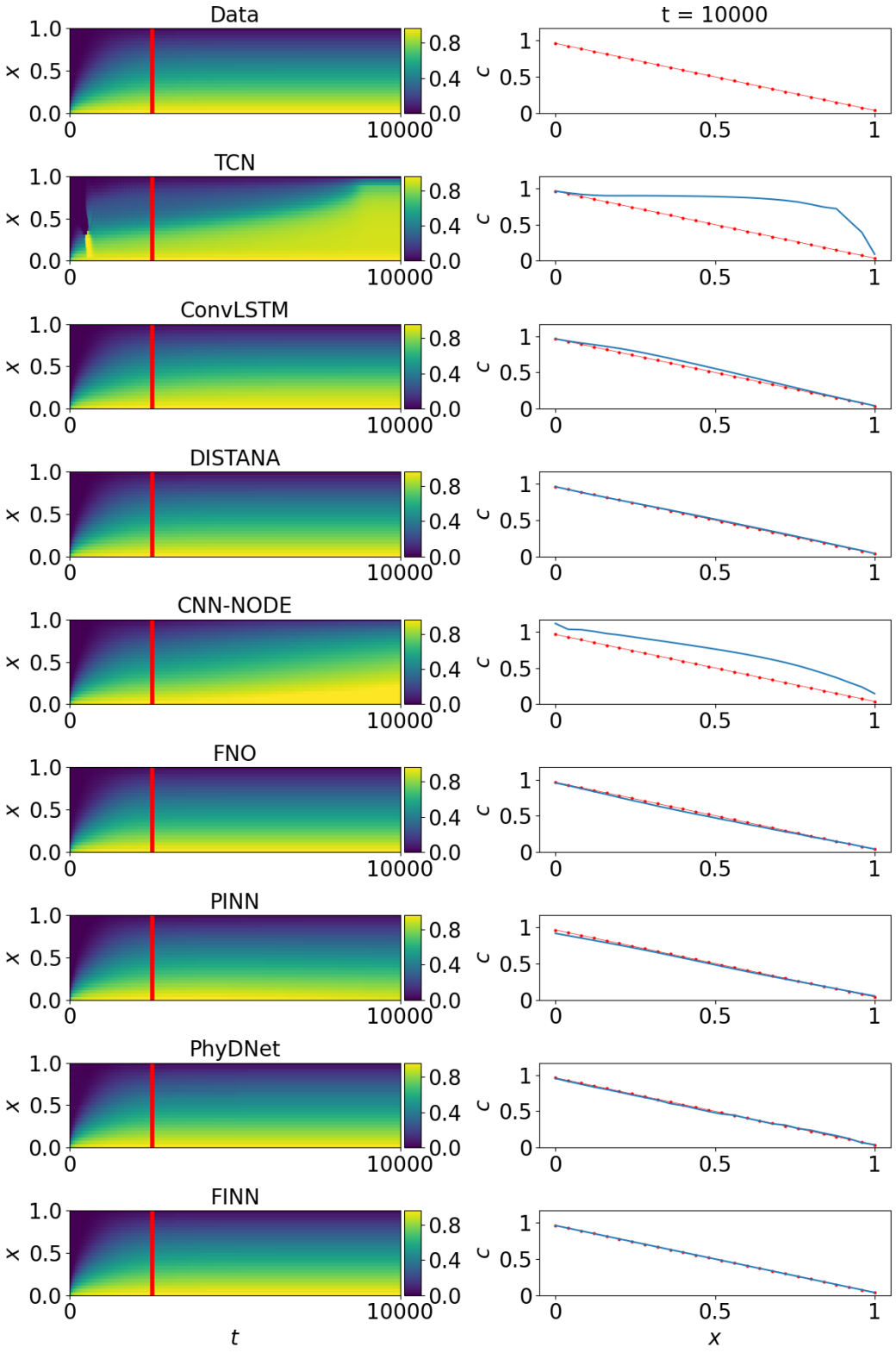

Figure 2: Plots of the dissolved concentration data generated with the Langmuir isotherm (red) and *in-dis-test* prediction (blue) using different models. The left column shows the solution over x and t (red lines mark the transition from *train* to *in-dis-test*), the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

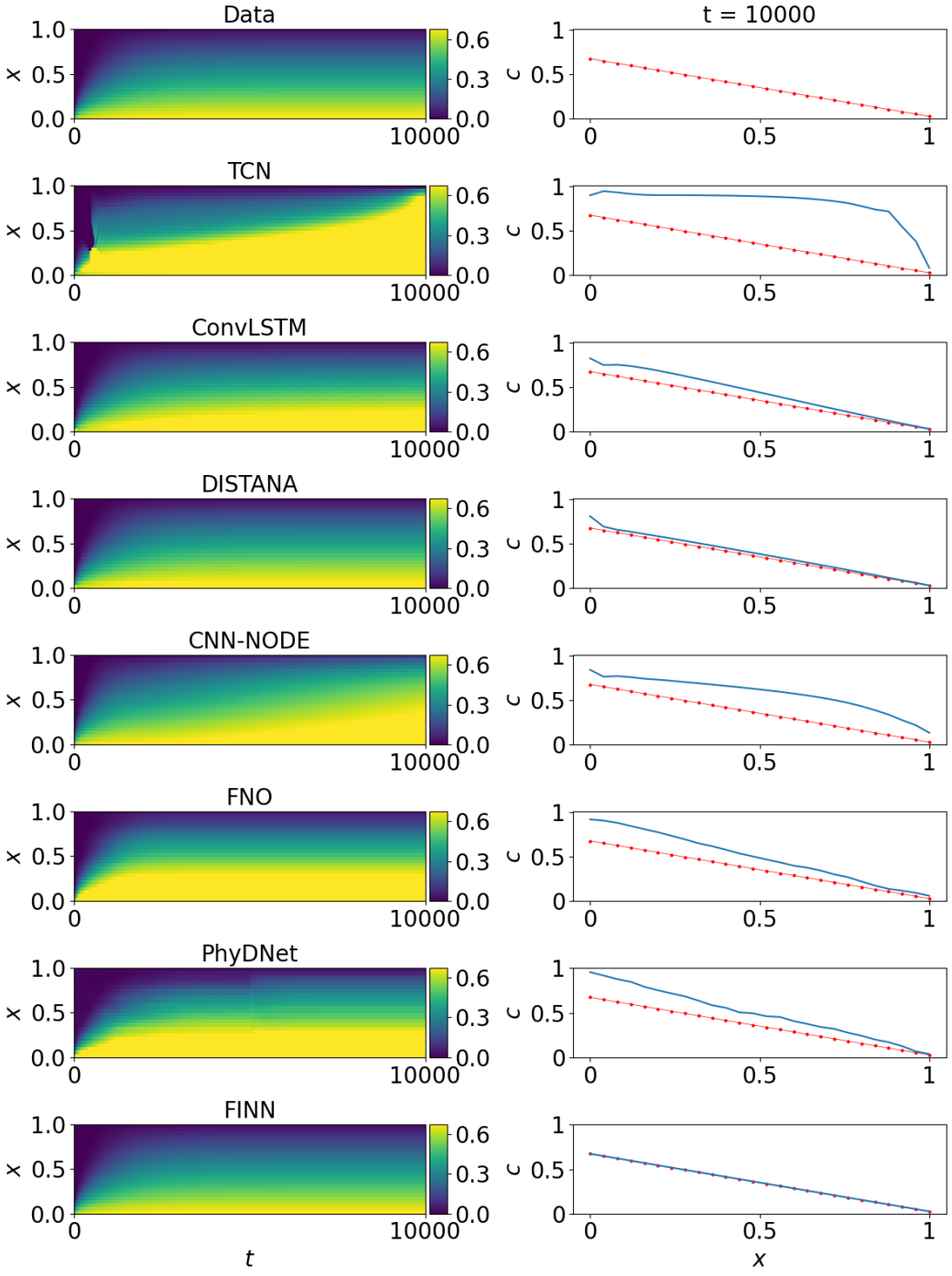

Figure 3: Plots of the dissolved concentration data generated with the Langmuir isotherm (red) and *out-dis-test* prediction (blue) using different models. The left column shows the solution over x and t, the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

Table 3: Comparison of the number of learnable parameters used by the different models. Each model uses the same number of parameters for the linear, Freundlich, and Langmuir cases.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
10 782	8 216	6 519	1 026	5 878	3 042	37 815	**464**

linear sorption isotherm, while PINN achieves the lowest prediction error for the data generated with the other sorption isotherms, namely Freundlich and Langmuir.

It is, however, more interesting to see how the models perform when confronted with unseen data, both extrapolation (*in-dis-test*) and different boundary condition (*out-dis-test*). For both test cases, all models perform significantly worse compared to the training phase. Nevertheless, FINN produces the best predictions with the lowest MSE, surpassing the other models by several orders of magnitude. More importantly, FINN is the only model with a consistently low prediction error with the same order of magnitude for all *train*, *in-dis-test*, and *out-dis-test* data. This model performance comparison is also visualized in Figure 4 for better clarity.

We also plot the predictions of the considered models for better visualization and understanding. For conciseness, we only show the plots for the dissolved concentration data generated with the Langmuir sorption isotherm due to similarities of the other plots. Figure 2 and Figure 3 show the best prediction of each model, that is the one that produces the lowest MSE among the ten randomly initialized trainings, when predicting the *in-dis-test* and the *out-dis-test* data, respectively. Figure 2 shows that most models, except for TCN and CNN-NODE, have at least one trained version that produces acceptable results even after extrapolation to a significantly longer time span ($T = 10\,000$ days) compared to the one covered during training ($T = 2\,500$ days). More interestingly and importantly, Figure 3 shows that when the boundary condition is changed to $c_{\text{sol}} = 0.7$, all models still tend to overfit to the boundary condition value used during training, i.e.

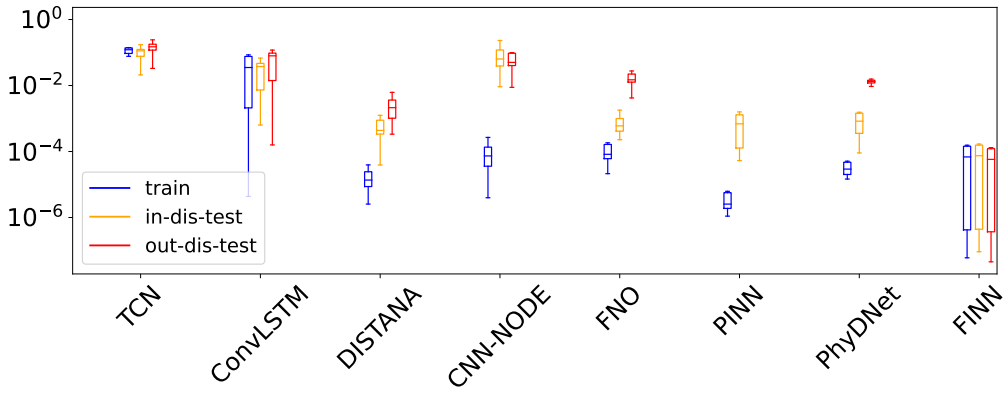

Figure 4: Average MSE comparison of different ML models with the error bars denoting the MSE standard deviation. The MSEs are calculated on the *train* (blue), *in-dis-test* (yellow), and *out-dis-test* (red) dataset generated using the Langmuir isotherm.

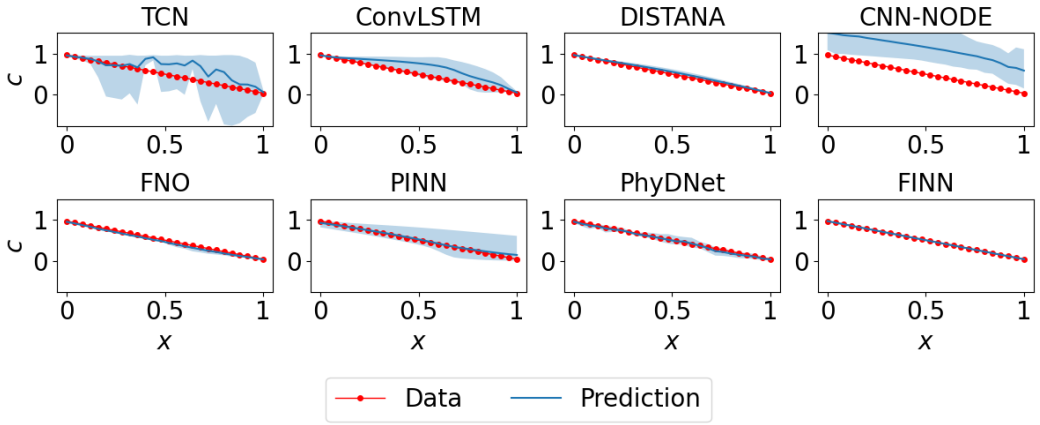

Figure 5: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration generated using the Langmuir isotherm at $t = 10\,000$ for the *in-dis-test* dataset.

$c_{\text{sol}} = 1.0$, demonstrating the tendency of all models to overestimate the dissolved concentration close to $x = 0$. One distinguishing feature of FINN is its ability to properly treat different values of numerical boundary conditions, and thus, FINN is the only model that does not suffer from the same overfitting issue of the other models, as also shown in Figure 2. Plots for data generated with other sorption isotherms and for the total concentration are presented in Appendix B.

To assess the robustness of the considered ML models, Figure 5 and Figure 6 show the prediction averaged over the ten different training initializations for the *in-dis-test* and *out-dis-test* data, respectively. Moreover, Figure 5 and Figure 6 are also equipped with 95% confidence intervals, to show the consistency of each model. These intervals are approximated with the t-distribution (Oliphant, 2006). Figure 5 shows that, even though each model has at least one good result, the other training result can still pro-

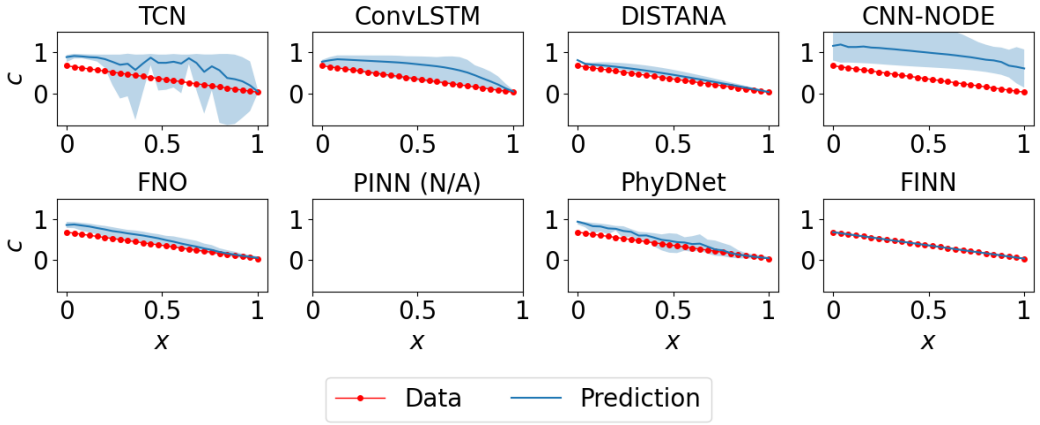

Figure 6: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration generated using the Langmuir isotherm at $t = 10\,000$ for the *out-dis-test* dataset.

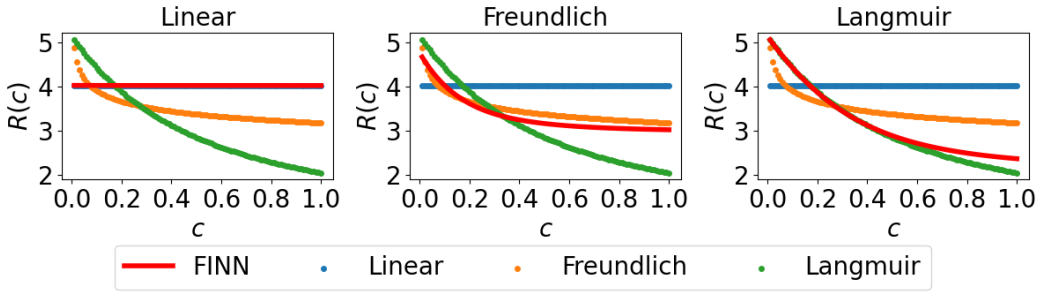

Figure 7: Learned retardation factor of FINN for the linear (left), Freundlich (middle), and Langmuir (right) sorption isotherm, compared with the retardation factor generated with the three isotherms.

duce incorrect predictions. To be more specific, the average predictions by TCN, ConvLSTM, and CNN-NODE do not fit the data. Additionally, we observe that most of the models produce highly inconsistent predictions with wide confidence intervals. Figure 6 (for *out-dis-test*) shows worse consistency of all models, evidenced by the wider confidence intervals. Additionally, all existing ML models still overfit to the boundary condition value used in the *train* data, as discussed earlier. FINN, in contrast, produces very consistent predictions, making the confidence interval hardly visible in Figure 5. Furthermore, FINN shows excellent consistency and adjustment to the new boundary condition value in Figure 6. The performance comparison between FINN and CNN-NODE emphasizes the relevance of FINN's modularized structure. Apparently, NODE alone does not guarantee accurate function approximations, which is reflected in the larger training errors of CNN-NODE compared to FINN, as well as in the test errors—consistently over all experiments. We show with this example that a structurized method to design the model using the FVM discretization as a basis is extremely beneficial.

Focusing on the physics-aware ML modeling concepts, we could state that PINN and PhyDNet lie on different extremes. PINN requires the modeler to know the complete form of the PDE to be solved. As a consequence, variables such as the diffusion coefficient or the retardation factor function also have to be known in advance to train the model. PhyDNet puts more emphasis on the data-driven part, shown by the high number of parameters in the ConvLSTM branch compared to the physics-aware branch. Therefore, PhyDNet has more freedom in learning, but can suffer from overfitting issues. This is shown by the fact that PhyDNet achieves a very low prediction error during training, but its result significantly deteriorates when predicting *in-dis-test* and *out-dis-test* data. The introduced FINN concept lies somewhere in the middle of these extremes, compromising between the freedom of learning and the rigidity of (assumed) physical knowledge. As a result, FINN outperforms the other models, especially on the *out-dis-test* data, which is considered a particularly challenging task for ML models. Finally, while FNO can—in contrast to PINN—still be applied to different initial and boundary conditions, it suffers from a noticeable performance drop when applied to the new boundary condition. This is not surprising, as the explicit function learned during training (mapping continuous space-time coordinates to c) does not hold in the *out-dis-test* scenario anymore.

4.1.3 Learning the PDE Constituents with FINN

The most important feature of FINN is its ability to interpretably learn the building blocks of the sought PDE. In our example, the numerical stencil, the diffusion co-

Table 4: Learned PDE constituents by using FINN.

Sorption isotherm	Numerical stencil	D [m^2/day]	Normalized D [m^2/day]
Linear	-1.06 ± 0.02 and 1.06 ± 0.02	$4.59 \pm 0.19 \times 10^{-4}$	$4.87 \pm 0.19 \times 10^{-4}$
Freundlich	-0.99 ± 0.04 and 0.99 ± 0.04	$4.83 \pm 0.19 \times 10^{-4}$	$4.78 \pm 0.34 \times 10^{-4}$
Langmuir	-1.08 ± 0.06 and 1.08 ± 0.06	$4.13 \pm 0.19 \times 10^{-4}$	$4.46 \pm 0.74 \times 10^{-4}$

efficient and the retardation factor function are learned during the training process. The learned numerical stencils and the learned diffusion coefficient D for the linear, Freundlich, and Langmuir sorption isotherm data are shown in the second and third column of Table 4. All the learned numerical stencils are symmetrical, meaning that the prediction is mass-conservative and clearly diffusive.

Assuming that the ideal numerical stencils should be -1 and 1, we normalize the learned diffusion coefficient (by multiplying it with the learned numerical stencils) and compare it to the real diffusion coefficient value ($D = 5.0 \times 10^{-4}\,\mathrm{m}^2/\mathrm{day}$) to evaluate the prediction error. The normalized diffusion coefficient for the linear, Freundlich, and Langmuir data are shown in the last column of Table 4. In terms of relative error to the real D value, these amount to 2.6%, 4.4%, and 10.8% error for the linear, Freundlich, and Langmuir isotherm, respectively. The prediction of the diffusion coefficient values has the highest error and variance for the Langmuir data, and the lowest error and variance for the linear data. As a consequence, FINN predicted the linear data with the highest accuracy, and Langmuir data with the lowest accuracy (see Table 2). Nevertheless, FINN's prediction is still highly consistent, as shown by the low MSE in Table 2 and the almost invisible confidence interval in Figure 5 and Figure 6.

Furthermore, a very strong advantage of FINN is its ability to successfully learn closure/constitutional relationships, which are often unknown when modeling a system. In our diffusion-sorption example, the major source of uncertainty is the retardation factor $R(c)$, which can be defined with various empirical functions. Figure 7 shows the retardation factor learned by FINN, when trained on data generated with the three aforementioned sorption isotherms. FINN is able to learn the retardation factor through its module $\varphi_{\mathcal{D}}$. The learned retardation factor also captures the linearity of the linear sorption isotherm, shown by the straight red line on the left plot in Figure 7. The retardation factors from the Freundlich and Langmuir isotherms are also captured well, even with less accuracy compared to the linear isotherm. This also contributes to the slightly higher prediction error for the Freundlich and Langmuir data compared to the linear data. Nevertheless, FINN is still able to distinguish between these different isotherms very well. Higher accuracy would need more informative data, especially at larger values of c.

4.2 Learning using an Experimental Dataset

Synthetic datasets provide good insights into FINN's performance in a controlled experiment, where the dataset is clean and abundant. However, real-world data is often only sparsely available due to costs or restrictions of equipments, the amount of time required to obtain useful data, or the difficulty in direct measurement of the system's internal states.

In the experimental setup described in section 2.1, the dissolved TCE concentration distribution inside the clay sample is unobservable throughout the experiment. The only means of measuring the dissolved TCE concentration is through the water flushing below the lower end of the clay cylinder, as of now called a breakthrough curve. Furthermore, the total TCE concentration can only be measured at the end of the exper-

Table 5: Parameter values of various clay core samples for the laboratory experiment.

Soil parameters				
Parameter	Unit	Core #1	Core #2	Core #2B
D	m^2/day	2.00×10^{-5}	2.00×10^{-5}	2.78×10^{-5}
ϕ	-	0.288	0.288	0.288
ρ_s	kg/m^3	1957	1957	1957
Simulation domain				
Parameter	Unit	Core #1	Core #2	Core #2B
L	m	0.0254	0.02604	0.105
r	m	0.02375	0.02375	N/A
T	days	38.81	39.82	48.88
Q	m^3/day	1.01×10^{-4}	1.04×10^{-4}	N/A
c_{sol}	kg/m^3	1.4	1.6	1.4

iment, by cutting the clay specimen into slices to enable direct (but destructive) measurement of the total TCE concentration. Hence, the dissolved concentration data is available as a breakthrough curve, which has only a single data point at each time step; and the spatial distribution of the total concentration data is available only at the final time step, and at coarse spatial resolution. Additionally, the observation data obtained from the experiment is very sparse and also noisy. All these challenges associated with the use of real-world data prompt the implementation of uncertainty quantification methods on FINN to provide honest and reliable predictions that would be useful for aiding critical decision making processes or hypothesis testing.

For this real-world application, three core samples are retrieved from the same geographical area, namely core samples #1, #2, and #2B (Nowak, 2000). Consequently, similar soil parameters can be assumed for all three samples, which are summarized in Table 5. The breakthrough curve of core sample #2 is the least noisy, and hence is chosen as the training data; whereas the breakthrough curve of core #1 is chosen to test the trained model. Additional test is also performed with the data from core sample #2B. However, core #2B is significantly longer than the other samples and the bottom of the setup is closed (no flushing). By the end of the experiment, no measurable TCE has yet arrived at the bottom end of the sample. Numerically, the experiment is modeled with the setup as described in section 2.1, with the initial condition written in Equation (5) and the boundary conditions written in Equations (3) and (4) for both core samples #1 and #2. For core sample #2B, because it is closed on the bottom, a no-flow Neumann boundary condition is used instead:

$$\left.\frac{\partial c}{\partial x}\right|_{x=L} = 0 \quad \forall t : 0 \leq t \leq T. \tag{32}$$

The breakthrough curve of core #2 used as training data only serve for model training using the Cauchy boundary condition described in equation (4). As a consequence, no other benchmark models can be used, since all of them, except for PINN, have no means of properly implementing numerical boundary conditions other than Dirichlet or periodic. PINN also cannot be applied in this example, because the test dataset from core sample #1 and #2B have different boundary conditions, and therefore the PINN model trained on core sample #2 no longer holds for the other samples. Moreover, one of the most interesting goals of this experiment is to learn the retardation factor, whereas all the considered ML models have no capability to do so explicitly. Therefore, we assess

the performance of FINN using a comparison to the PDE-based physical model—the same as used to generate synthetic data in section 4.1—calibrated to the experimental data as a benchmark. The best fit of the physical model is found with the retardation factor modeled using the Freundlich sorption isotherm, with $K_f = 5.20 \times 10^{-4}$ $(\mathrm{m}^3/\mathrm{kg})^{\mathrm{n_f}}$ and $n_f = 0.35$.

For this application, FINN is implemented with the use of the module $\varphi_{\mathcal{D}}$. Here, the module $\varphi_{\mathcal{D}}$ is defined as a feedforward network with the size of [1, 10, 20, 10, 1] that takes c as an input and outputs the retardation factor $R(c)$. The diffusion coefficient is assumed to be known and measurable for all the core samples (Nowak, 2000), and therefore is not learned by FINN. As in the synthetic case, FINN is trained with the objective to minimize the deviation between the model predictions of c as a breakthrough curve and the total concentration c_t at the end of the experiment. Also, and in contrast to the synthetic data scenario, FINN is now trained using the Bayes-by-backprop method as outlined in section 2.4 using Equation 19 as the loss function formulation. Even though the Bayes-by-backprop method manages to provide reasonable uncertainty quantification of FINN's prediction, it fails to learn the standard deviation parameter σ sufficiently (i.e. the learned σ values do not differ much from the initial values).

Due to the limitations of the Bayes-by-backprop method, we alternatively use three different MCMC methods, namely the random walk Metropolis-Hastings (MH), Metropolis-adjusted Langevin algorithm (MALA), and Barker proposal (Barker). Starting with random initial values of FINN's parameters, samplings are performed with these three methods for 102 000 iterations, with the 2 000 initial iterations discarded as the burn-in period, resulting in 100 000 effective iterations. This number is chosen as the upper limit, to investigate which MCMC method provides decent convergence and offers the most efficient sampling under acceptable computational time.

Out of the 100 000 iterations, we thin out the samples by saving only every 10^{th} iteration, resulting in a total of 10 000 samples. The step size h is chosen so that the acceptance rate amounts to approximately 23% (Reuschen et al., 2021). This corresponds to $h = 10^{-2}$ for MH, $h = 7 \times 10^{-6}$ for MALA, and $h = 4 \times 10^{-3}$ for Barker. As shown by the trace plot (the left plot in Figure 8), all methods improve the log posterior substantially after only a few iterations. However, looking at the zoomed-in plot (the right plot in Figure 8), none of the used MCMC methods converge well to an equilibrium distribution, as evidenced by the downward trend of the log posterior even until the last iteration. Even though MALA has better convergence compared to the other methods, the acceptance rate quickly deteriorates. This can be attributed to the fact that MALA is less robust to high step size (Livingstone and Zanella (2019), although the step size used for MALA is already very low in this case).

To improve the performance of the MCMC chains, we start the sampling with optimized parameter values of FINN. To obtain this, FINN is first trained deterministically as in the synthetic data scenario, and the parameter values of the trained FINN model are used as the starting point of the MCMC chain. Samplings are then performed for 100 000 iterations, without the burn-in period (since we start with optimized values), again thinning by a factor of 10, resulting in a total of 10 000 samples. The corresponding step sizes are $h = 10^{-2}$ for MH, $h = 3 \times 10^{-5}$ for MALA, and $h = 5 \times 10^{-3}$ for Barker.

As shown by the log posterior trace plot (the left plot in Figure 9), there is a downward trend in the log posterior of the samples. This is fundamentally caused by using an excellent starting point with minimized error (statistically too good to be a representative sample), resulting in less good (but as of then statistically valid) subsequent samples. When zooming in after 5 000 iterations, we observe that the performance of both the gradient-based MCMC methods, namely MALA and Barker, is better than that of MH. Among these two methods, however, Barker shows the best behavior with the highest and most stable log posterior values, indicating proper sampling from the desired pos-

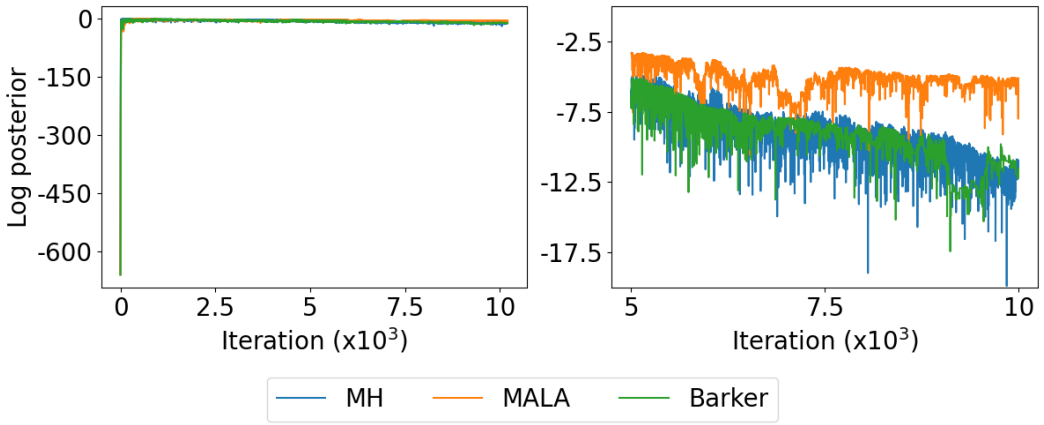

Figure 8: Trace plot of the log posterior starting with random initial values (left) and the zoomed-in plot after 5 000 iterations (right) for the Metropolis Hastings (MH, blue), Metropolis-adjusted Langevin algorithm (MALA, orange), and Barker (green) MCMC methods.

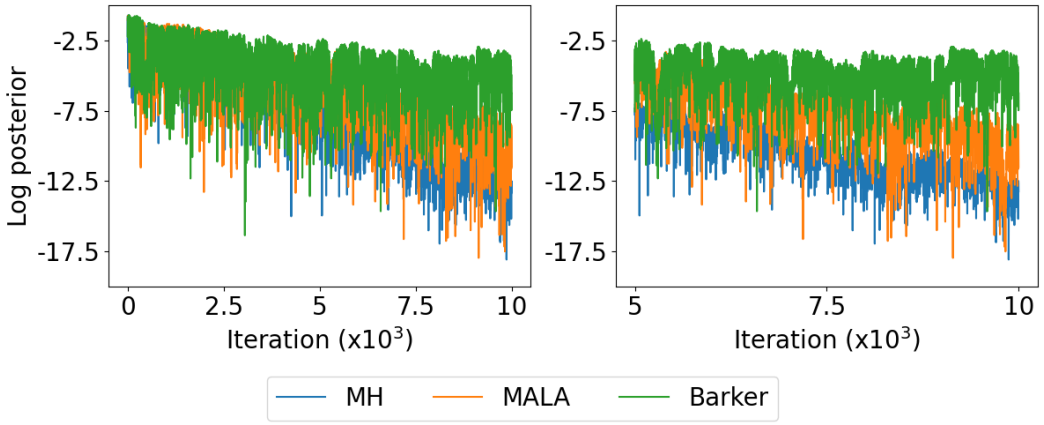

Figure 9: Trace plot of the log posterior starting with optimized initial values (left) and the zoomed-in plot after 5 000 iterations (right) for the Metropolis Hastings (MH, blue), Metropolis-adjusted Langevin algorithm (MALA, orange), and Barker (green) MCMC methods.

terior. On the other hand, the log posterior value of the samples obtained using MALA is still slightly decreasing. This result shows that Barker scales well for higher dimensionality, especially when the chain is properly initialized. All in all, only the optimized Barker MCMC finishes its burn-in properly and reaches an equilibrium distribution, that is the desired posterior, within the 100 000 iterations window.

Another way to quantify the predictive performance/sharpness of the MCMC methods is to plot the reliability curve as defined in Jospin et al. (2022), which is calculated using the cumulative distribution function (CDF) across all samples, compared to its observed probability (i.e. ordered against the actual data). The reliability plot enables evaluation of the model's predictive performance. The model is described as underconfident if the reliability curve lies above the baseline, and overconfident otherwise. As shown in Figure 10, all the methods with random initialization (left plot) lie further from the ideal

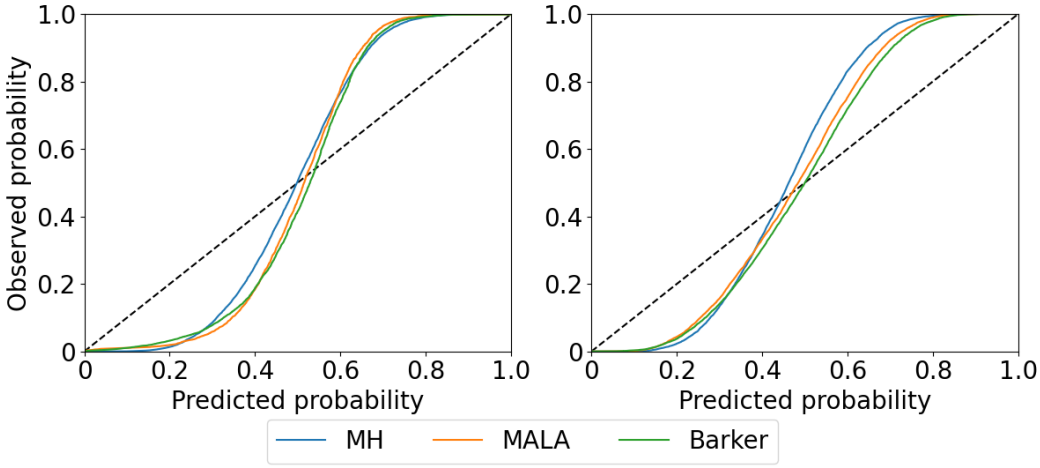

Figure 10: Reliability curves of the MH (blue), MALA (orange), and Barker (green) MCMC methods initialized randomly (left) and with optimized values (right). The baseline for the ideal condition is shown by the black dashed line.

condition compared to the methods with optimized starting point (right plot). Among all the methods, Barker MCMC with optimized initialization lies the closest to the ideal condition, confirming further that the samples generated by the optimized Barker MCMC provides the best predictive performance. It is also interesting to note that the models are overconfident for lower extremes and underconfident for higher extremes. One possible explanation is that the data error is not Gaussian.

The predictions obtained using the optimized Barker MCMC are shown in Figure 11. The MCMC method augment FINN's prediction with a confidence interval, which captures most of the noisy observation data inside, showing sufficient uncertainty quantification. Quantitatively, FINN achieves lower training error on core sample #2 data with $\text{MSE} = 5.43 \times 10^{-4}$, compared to the physical model with $\text{MSE} = 1.06 \times 10^{-3}$. During testing with data from core sample #1, FINN also outperforms the physical model with $\text{MSE} = 1.41 \times 10^{-3}$ compared to $\text{MSE} = 2.50 \times 10^{-3}$, because the calibrated physical model underestimates the TCE breakthrough curve. When tested against data from core sample #2B, which has a different type of numerical boundary condition implemented, FINN again achieves lower prediction error with $\text{MSE} = 1.16 \times 10^{-3}$ compared to the calibrated physical model that overestimates the TCE concentration with $\text{MSE} = 2.73 \times 10^{-3}$. Because there is no breakthrough curve data available for this specific sample, we compare the prediction against the total concentration profile $c(x, t = T)$ at the end of the experiment.

Moreover, we also plot FINN's learned retardation factor in comparison to the calibrated Freundlich retardation factor, which shows that the best-fitting available sorption isotherm model fails to capture the retardation factor shape as learned by FINN, possibly leading to the higher prediction error of the calibrated physical model in both cases of training and testing. Overall, FINN outperforms the calibrated physical model by learning the retardation factor better than the parametric sorption isotherm model using only the breakthrough curve of core sample #2 (i.e. only 55 data points) and successfully applies it to the other samples with relatively high accuracy.

As a side note, this real-world application example adopted in this work was performed in a small-scale laboratory experiment. With the corresponding scale, homogeneity could be assumed for the modeled soil parameters. For a larger-scale application (i.e.

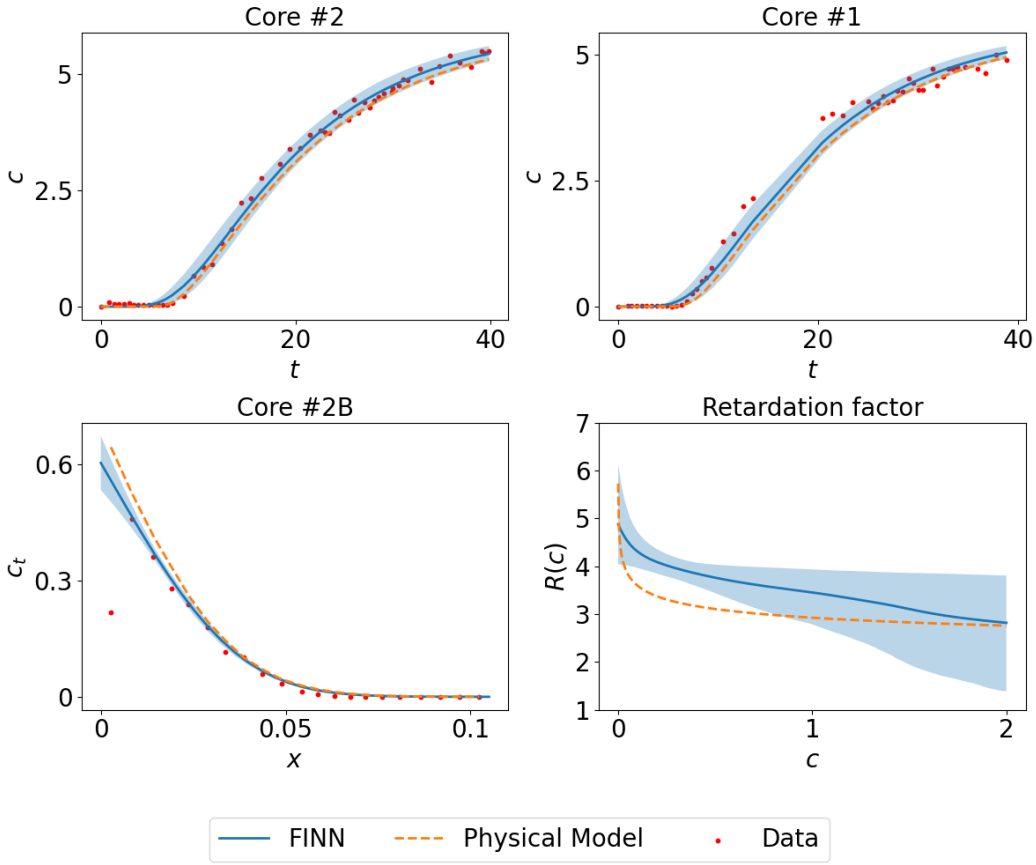

Figure 11: Breakthrough curve average prediction of FINN (blue line) and its 95% confidence interval (blue shade) during training on core sample #2 (top left), during testing on core sample #1 (top right) and total concentration profile of core sample #2B (bottom left). The predictions are compared with the experimental data (red circles) and the results obtained using the physical model (orange dashed line). The learned retardation factor $R(c)$ is shown in the bottom right plot.

field-scale), the assumption might no longer hold, and thus heterogeneity would have to be taken into account. To account for heterogeneity, FINN has to either adopt a geostatistical approach to model the heterogeneous distribution of the parameter, (e.g. the diffusion coefficient), or a graph representation.

Finally, even though we only showed the application of FINN to a subsurface contaminant transport problem, it is also applicable to a range of other problems or equations, such as the 2D Burgers' the diffusion-reaction equation, and the Allen-Cahn equation. For further details, we refer the interested readers to our ML-focused paper (Karlbauer et al., 2022). FINN lays the groundwork for further development of hybrid modeling frameworks in this area, and hopefully can be used for an even wider range of problems in the future, such as weather and climate simulation or investigation of improved constitutive relations in multiphase flow.

5 Summary and Conclusion

In this work, we applied FINN, a hybrid modeling framework that induces physical inductive biases into an ANN learning paradigm. FINN is based on the numerical structure of the FVM for solving PDEs, as well as conditions such as monotonicity and non-negativity of functions or parameters to constrain the model training. FINN learns numerical stencils, unknown constitutional/closure relationships, and/or parameters to predict variables of interest in spatiotemporal physical systems. Using a well-controlled subsurface contaminant transport benchmark study, we showed that FINN is beneficial in comparison to pure ML models as well as physics-motivated and physics-aware ML models for several reasons.

First, FINN demonstrates superior generalization when tested against extrapolated data and data generated with different boundary condition. The other ML models participating in our comparison have the tendency to overfit, while PINN is not even applicable to the same system with a different boundary condition.

Second, FINN allows proper treatment of different boundary condition types, whereas other models are only applicable to boundary condition types with constant values such as Dirichlet or periodic ones, because of the convolutional structure adopted in most of the other models. As a result, only FINN can be trained on data under a Cauchy boundary condition in the form of a diffusive breakthrough curve in the real-world experimental data example.

Third, FINN can be trained with a sparse dataset without compromising its learning ability and its prediction accuracy. It was shown in the real-world data example that FINN was trained with only 55 data points, yet it generalized well to other unseen samples, even one with a different boundary condition type. This also means that FINN is applicable to real-world data that is noisy and sparse. Hence, FINN offers a data-driven modeling approach that goes beyond a surrogate modeling tool.

Fourth, FINN provides flexibility in choosing between different uncertainty quantification methods, especially due to its comparatively low number of parameters. FINN can be paired not only with the variational inference type of uncertainty quantification (i.e. Bayes-by-backprop), but also with MCMC methods. Additionally, the widely available automatic differentiation tools in various ML libraries enable the use of gradient information in both MALA and Barker MCMC, leading to better performance compared to a random walk MH. Furthermore, these automatic differentiation tools also promote finding an optimal starting point for the MCMC chain, by first training FINN deterministically.

Fifth, and probably most importantly, FINN's structure provides a high degree of model explainability. Through its structure, FINN can explicitly learn unknown constitutive/closure relationships or parameters, which are usually the main source of uncertainty in physical systems modeling. The particular example shown in this work is the unknown retardation factor as a function of concentration. Because the available sorption isotherm models describe the retardation factor function using few parameters, they are not flexible enough to be calibrated to learn the "true" shape of the function. FINN, on the other hand, has the full flexibility to learn it.

To re-emphasize, in this work we did not intend to develop FINN as a faster and more efficient surrogate model in place of known equations and their numerical solvers. The only comparison between FINN and a traditional numerical simulation model was presented on the real-world problem example. The purpose of this comparison was to show that, when calibrating the physical model, we are faced with discrete choices of models that lead to difficulties in capturing the "true" functional relationship. FINN, on the other hand, alleviates this problem by providing a flexible way of learning the unknown relationship, as shown in section 4.2.

Table A1: Closed-loop MSE on the *train* data from ten different training runs for each model for the *linear* isotherm.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
8.2×10^{-1}	1.2×10^{-1}	1.1×10^{-5}	1.3×10^{-4}	2.6×10^{-5}	6.3×10^{-6}	3.6×10^{-5}	1.8×10^{-7}
1.1×10^{-1}	1.2×10^{-2}	8.5×10^{-5}	6.0×10^{-5}	8.9×10^{-5}	2.3×10^{-4}	4.2×10^{-5}	5.2×10^{-7}
2.8×10^{-3}	6.9×10^{-2}	1.2×10^{-5}	9.4×10^{-3}	5.7×10^{-5}	3.4×10^{-4}	4.1×10^{-5}	2.3×10^{-8}
7.8×10^{-2}	1.2×10^{-5}	3.2×10^{-5}	2.4×10^{-6}	6.5×10^{-5}	1.4×10^{-5}	5.2×10^{-5}	3.2×10^{-7}
3.0×10^{-1}	1.4×10^{-5}	7.5×10^{-6}	2.4×10^{-5}	1.2×10^{-4}	4.5×10^{-6}	5.9×10^{-5}	1.5×10^{-7}
9.7×10^{-2}	4.4×10^{-2}	4.1×10^{-4}	4.1×10^{-6}	6.4×10^{-5}	1.9×10^{-6}	1.8×10^{-5}	3.0×10^{-7}
1.3×10^{-1}	6.4×10^{-2}	2.2×10^{-3}	7.4×10^{-5}	5.4×10^{-5}	1.3×10^{-6}	3.4×10^{-5}	1.0×10^{-7}
3.9×10^{-2}	4.8×10^{-5}	3.2×10^{-5}	9.0×10^{-4}	1.4×10^{-4}	7.0×10^{-6}	1.4×10^{-5}	3.0×10^{-7}
1.8×10^{-3}	3.8×10^{-3}	2.6×10^{-5}	5.9×10^{-3}	5.5×10^{-5}	1.7×10^{-5}	2.2×10^{-5}	1.5×10^{-7}
8.6×10^{-1}	5.2×10^{-2}	7.5×10^{-6}	4.3×10^{-3}	8.2×10^{-5}	2.8×10^{-6}	1.5×10^{-5}	6.0×10^{-9}

Despite the promising benefits of FINN, we realized that there is still a lot of room for improvement. For instance, the computation time of FINN is still not optimized, because the implementation is highly dependent on the available Neural ODE package. It will not be as fast as PINN, because PINN models the system as an explicit function of x and t, allowing to parallelize the computation. FINN and the other models such as ConvLSTM, DISTANA, and PhyDNet, on the other hand, rely on recurrent structures, which prohibit full parallelization.

Additionally, FINN offers room for possible extension of the underlying physical theory behind the framework. For example, in non-Fickian diffusion or dispersion, where the diffusion coefficient could be a highly complex function of many variables, or if there are neglected processes that contribute significantly to the modeled system such as chemical transformation, evaporation, or advection. In these cases, FINN's structure would need to be modified to allow for a more complicated representation of the unknown functional relationships.

Uncertainty quantification of ML models is still a broad and open research area, due to the complicated nature of ML models, and ANNs in particular. More rigorous analysis of uncertainty quantification on FINN can further improve the predictive performance and foster the interpretability of the model itself. Furthermore, it would also be beneficial to reach a fully accurate total uncertainty quantification over inferred scientific hypotheses. This could include not only the uncertainty from parameters, closures, and stencils as done in this work, but also over entirely different conceptualizations of a system. The latter could result in different structural set-ups of FINN. In order to follow-up on these questions, additional research is necessary.

Appendix A Details of All Model Runs

For all benchmark models that are used in this work and for each sorption isotherm used to generate the synthetic dataset, ten different random initializations are conducted to have a more comprehensive analysis of all the models' performance. Table A1, Table A2, and Table A3 show the results of the linear sorption isotherm applied to the *train*, *in-dis-test*, and *out-dis-test* data, respectively. Table A4, Table A5, and Table A6 show the results of the Freundlich sorption isotherm applied to the *train*, *in-dis-test*, and *out-dis-test* data, respectively. Table A7, Table A8, and Table A9 show the results of the Langmuir sorption isotherm applied to the *train*, *in-dis-test*, and *out-dis-test* data, respectively.

Table A2: Closed-loop MSE on the *in-dis-test* data from ten different training runs for each model for the *linear* isotherm.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
1.1×10^{0}	8.7×10^{-2}	1.3×10^{-3}	7.2×10^{-2}	5.6×10^{-4}	2.8×10^{-4}	2.6×10^{-4}	2.3×10^{-7}
9.6×10^{-2}	2.9×10^{-2}	1.3×10^{-3}	3.5×10^{-2}	1.4×10^{-3}	4.9×10^{-3}	5.1×10^{-4}	6.6×10^{-7}
6.1×10^{-2}	6.1×10^{-2}	1.7×10^{-4}	6.3×10^{-1}	9.6×10^{-4}	2.2×10^{-3}	2.9×10^{-4}	4.0×10^{-8}
6.3×10^{-2}	2.1×10^{-4}	5.8×10^{-4}	3.0×10^{-2}	9.3×10^{-4}	1.5×10^{-5}	5.7×10^{-2}	4.0×10^{-7}
4.6×10^{-1}	1.9×10^{-4}	4.8×10^{-4}	3.9×10^{-2}	1.5×10^{-3}	8.1×10^{-5}	4.7×10^{-4}	1.7×10^{-7}
1.7×10^{-1}	3.8×10^{-2}	2.5×10^{-3}	1.6×10^{-2}	1.1×10^{-3}	2.6×10^{-5}	3.2×10^{-4}	4.1×10^{-7}
9.8×10^{-2}	1.2×10^{-1}	9.2×10^{-3}	6.8×10^{-2}	8.0×10^{-4}	1.2×10^{-5}	5.5×10^{-4}	1.3×10^{-7}
1.0×10^{-1}	1.7×10^{-3}	1.6×10^{-4}	1.3×10^{-1}	1.3×10^{-3}	2.0×10^{-3}	3.1×10^{-4}	4.1×10^{-7}
4.1×10^{-2}	9.3×10^{-3}	2.8×10^{-3}	3.6×10^{-1}	6.4×10^{-4}	2.7×10^{-2}	4.1×10^{-4}	2.1×10^{-7}
1.3×10^{0}	5.3×10^{-2}	8.3×10^{-5}	2.4×10^{-1}	9.0×10^{-4}	2.7×10^{-3}	2.7×10^{-4}	1.5×10^{-8}

Table A3: Closed-loop MSE on the *out-dis-test* data from ten different training runs for each model for the *linear* isotherm.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
8.4×10^{-1}	1.5×10^{-1}	1.5×10^{-3}	7.5×10^{-2}	2.4×10^{-2}	N/A	1.1×10^{-2}	1.6×10^{-7}
1.5×10^{-1}	5.2×10^{-2}	7.6×10^{-3}	3.3×10^{-2}	1.7×10^{-2}	N/A	9.7×10^{-3}	4.5×10^{-7}
7.1×10^{-2}	1.1×10^{-1}	1.2×10^{-3}	6.3×10^{-1}	2.6×10^{-2}	N/A	1.1×10^{-2}	1.6×10^{-8}
1.0×10^{-1}	1.2×10^{-4}	7.9×10^{-3}	3.1×10^{-2}	1.6×10^{-2}	N/A	3.5×10^{-2}	2.8×10^{-7}
3.4×10^{-1}	7.5×10^{-3}	2.7×10^{-3}	3.1×10^{-2}	1.9×10^{-2}	N/A	3.7×10^{-2}	1.2×10^{-7}
1.1×10^{-1}	4.8×10^{-2}	1.9×10^{-3}	1.6×10^{-2}	1.8×10^{-2}	N/A	1.2×10^{-2}	2.7×10^{-7}
1.6×10^{-1}	5.0×10^{-2}	5.2×10^{-3}	8.7×10^{-2}	1.3×10^{-2}	N/A	1.2×10^{-2}	9.5×10^{-8}
1.3×10^{-1}	7.1×10^{-3}	3.6×10^{-3}	1.4×10^{-1}	1.9×10^{-2}	N/A	1.2×10^{-2}	2.7×10^{-7}
5.9×10^{-2}	8.2×10^{-3}	4.7×10^{-3}	2.5×10^{-1}	1.7×10^{-2}	N/A	1.2×10^{-2}	1.4×10^{-7}
9.4×10^{-1}	9.9×10^{-2}	2.4×10^{-3}	2.4×10^{-1}	2.2×10^{-2}	N/A	1.2×10^{-2}	6.0×10^{-9}

Table A4: Closed-loop MSE on the *train* data from ten different training runs for each model for the *Freundlich* isotherm.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
1.2×10^{-2}	2.2×10^{-2}	2.4×10^{-6}	4.3×10^{-4}	4.2×10^{-5}	6.3×10^{-6}	3.0×10^{-5}	2.7×10^{-5}
3.1×10^{-1}	2.9×10^{-4}	5.9×10^{-6}	1.4×10^{-5}	6.0×10^{-3}	1.8×10^{-6}	4.5×10^{-5}	3.8×10^{-5}
1.5×10^{-1}	1.4×10^{-4}	1.3×10^{-5}	9.1×10^{-3}	1.1×10^{-4}	3.2×10^{-6}	4.5×10^{-5}	2.6×10^{-5}
1.3×10^{-1}	2.2×10^{-3}	1.6×10^{-5}	4.2×10^{-5}	8.5×10^{-5}	2.6×10^{-6}	6.8×10^{-3}	2.6×10^{-5}
2.7×10^{-4}	5.8×10^{-2}	1.2×10^{-6}	3.0×10^{-2}	2.5×10^{-4}	2.6×10^{-6}	2.7×10^{-5}	3.4×10^{-5}
3.8×10^{-1}	6.6×10^{-3}	3.0×10^{-6}	4.4×10^{-5}	5.6×10^{-5}	2.1×10^{-6}	3.7×10^{-5}	3.1×10^{-5}
8.8×10^{-3}	8.6×10^{-2}	5.3×10^{-6}	2.6×10^{-3}	3.6×10^{-5}	8.3×10^{-6}	3.3×10^{-5}	2.4×10^{-5}
1.5×10^{-2}	1.6×10^{-2}	2.4×10^{-5}	1.0×10^{-4}	4.5×10^{-5}	7.4×10^{-6}	2.9×10^{-5}	2.6×10^{-5}
1.2×10^{-3}	1.6×10^{-5}	6.5×10^{-6}	1.5×10^{-4}	3.4×10^{-5}	2.1×10^{-6}	2.9×10^{-5}	2.9×10^{-5}
1.2×10^{-1}	9.5×10^{-6}	3.2×10^{-6}	2.9×10^{-4}	7.1×10^{-5}	6.6×10^{-6}	3.2×10^{-5}	2.7×10^{-5}

Table A5: Closed-loop MSE on the *in-dis-test* data from ten different training runs for each model for the *Freundlich* isotherm.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
3.0×10^{-2}	2.8×10^{-2}	9.7×10^{-6}	1.1×10^{-1}	4.1×10^{-4}	1.9×10^{-3}	1.0×10^{-3}	2.5×10^{-5}
3.5×10^{-1}	2.1×10^{-2}	2.2×10^{-4}	5.7×10^{-2}	9.4×10^{-3}	6.0×10^{-4}	2.9×10^{-4}	3.6×10^{-5}
2.4×10^{-1}	1.1×10^{-2}	2.9×10^{-4}	3.4×10^{-2}	6.7×10^{-4}	1.0×10^{-4}	4.8×10^{-3}	2.4×10^{-5}
1.5×10^{-1}	5.6×10^{-3}	4.7×10^{-4}	2.3×10^{-2}	4.8×10^{-4}	4.2×10^{-4}	1.3×10^{-2}	2.4×10^{-5}
8.6×10^{-3}	4.7×10^{-2}	8.3×10^{-6}	1.8×10^{0}	1.0×10^{-3}	5.5×10^{-5}	1.2×10^{-3}	3.3×10^{-5}
4.7×10^{-1}	3.5×10^{-2}	7.8×10^{-5}	8.3×10^{-2}	4.2×10^{-4}	3.2×10^{-4}	2.4×10^{-4}	2.9×10^{-5}
1.3×10^{-1}	5.8×10^{-2}	8.5×10^{-5}	2.3×10^{-1}	3.3×10^{-4}	5.6×10^{-3}	1.5×10^{-4}	2.2×10^{-5}
1.6×10^{-2}	3.0×10^{-2}	4.6×10^{-4}	1.2×10^{-1}	1.5×10^{-4}	1.9×10^{-4}	1.2×10^{-4}	2.4×10^{-5}
4.3×10^{-3}	4.5×10^{-4}	4.9×10^{-5}	4.9×10^{-2}	4.5×10^{-4}	1.4×10^{-4}	1.4×10^{-4}	2.7×10^{-5}
2.0×10^{-1}	2.9×10^{-4}	1.4×10^{-4}	6.1×10^{-2}	7.7×10^{-4}	4.6×10^{-4}	1.4×10^{-3}	2.6×10^{-5}

Table A6: Closed-loop MSE on the *out-dis-test* data from ten different training runs for each model for the *Freundlich* isotherm.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
3.4×10^{-3}	6.6×10^{-2}	8.2×10^{-4}	1.0×10^{-1}	1.4×10^{-2}	N/A	1.3×10^{-2}	2.1×10^{-5}
2.9×10^{-1}	2.8×10^{-3}	1.0×10^{-3}	4.3×10^{-2}	3.6×10^{-3}	N/A	1.1×10^{-2}	3.0×10^{-5}
1.3×10^{-1}	2.4×10^{-2}	2.1×10^{-3}	2.5×10^{-2}	1.1×10^{-2}	N/A	1.1×10^{-2}	2.1×10^{-5}
1.5×10^{-1}	1.5×10^{-2}	6.2×10^{-4}	2.0×10^{-2}	1.6×10^{-2}	N/A	1.1×10^{-2}	2.0×10^{-5}
1.2×10^{-2}	8.8×10^{-2}	5.8×10^{-4}	1.5×10^{0}	2.2×10^{-2}	N/A	1.2×10^{-2}	2.7×10^{-5}
4.2×10^{-1}	6.1×10^{-2}	1.6×10^{-3}	9.3×10^{-2}	1.5×10^{-2}	N/A	1.1×10^{-2}	2.4×10^{-5}
9.5×10^{-2}	1.1×10^{-1}	2.0×10^{-3}	2.2×10^{-1}	1.4×10^{-2}	N/A	1.2×10^{-2}	1.9×10^{-5}
3.4×10^{-2}	6.2×10^{-2}	5.2×10^{-3}	9.2×10^{-2}	1.6×10^{-2}	N/A	1.2×10^{-2}	2.1×10^{-5}
1.1×10^{-2}	1.2×10^{-3}	1.2×10^{-3}	4.6×10^{-2}	1.0×10^{-2}	N/A	1.2×10^{-2}	2.3×10^{-5}
1.3×10^{-1}	4.2×10^{-5}	7.1×10^{-5}	5.5×10^{-2}	1.5×10^{-2}	N/A	1.5×10^{-2}	2.1×10^{-5}

Table A7: Closed-loop MSE on the *train* data from ten different training runs for each model for the *Langmuir* isotherm.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
2.2×10^{-1}	6.6×10^{-3}	2.6×10^{-6}	1.0×10^{-4}	2.1×10^{-5}	2.0×10^{-6}	2.1×10^{-5}	1.3×10^{-4}
1.4×10^{-1}	8.1×10^{-2}	1.3×10^{-5}	1.7×10^{-5}	1.8×10^{-4}	1.1×10^{-6}	1.5×10^{-4}	7.7×10^{-7}
1.3×10^{-1}	1.3×10^{-5}	2.3×10^{-5}	6.0×10^{-5}	8.3×10^{-5}	4.0×10^{-6}	3.3×10^{-5}	3.1×10^{-7}
7.7×10^{-2}	8.5×10^{-2}	1.4×10^{-5}	1.1×10^{-3}	7.1×10^{-5}	1.6×10^{-6}	2.0×10^{-5}	1.4×10^{-4}
1.3×10^{-1}	3.3×10^{-2}	3.0×10^{-6}	3.4×10^{-5}	1.1×10^{-4}	1.9×10^{-6}	1.7×10^{-5}	2.1×10^{-7}
1.2×10^{-1}	4.3×10^{-6}	1.1×10^{-5}	1.5×10^{-4}	8.1×10^{-5}	1.2×10^{-5}	5.1×10^{-5}	6.0×10^{-8}
2.4×10^{-1}	6.0×10^{-4}	9.4×10^{-5}	2.7×10^{-4}	2.5×10^{-3}	3.0×10^{-4}	1.5×10^{-5}	1.6×10^{-4}
9.0×10^{-2}	6.0×10^{-2}	2.5×10^{-5}	8.7×10^{-5}	3.8×10^{-4}	2.4×10^{-6}	2.6×10^{-5}	1.5×10^{-4}
7.5×10^{-3}	3.7×10^{-2}	3.9×10^{-5}	4.0×10^{-6}	5.7×10^{-5}	6.2×10^{-6}	3.7×10^{-5}	2.8×10^{-6}
1.1×10^{-1}	8.3×10^{-2}	7.7×10^{-6}	4.1×10^{-5}	2.6×10^{-5}	2.7×10^{-6}	9.0×10^{-5}	1.5×10^{-4}

Table A8: Closed-loop MSE on the *in-dis-test* data from ten different training runs for each model for the *Langmuir* isotherm.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
1.2×10^{-1}	2.0×10^{-2}	3.9×10^{-5}	3.2×10^{-2}	2.3×10^{-4}	5.3×10^{-4}	4.9×10^{-4}	1.5×10^{-4}
1.2×10^{-1}	6.8×10^{-2}	1.2×10^{-3}	8.3×10^{-2}	1.1×10^{-3}	1.0×10^{-4}	6.6×10^{-4}	1.1×10^{-6}
8.2×10^{-2}	6.4×10^{-4}	3.9×10^{-4}	4.4×10^{-1}	5.6×10^{-4}	1.1×10^{-4}	1.0×10^{-3}	2.2×10^{-7}
1.1×10^{-1}	4.7×10^{-2}	4.8×10^{-4}	2.3×10^{-1}	6.1×10^{-4}	1.4×10^{-3}	1.3×10^{-4}	1.5×10^{-4}
1.2×10^{-1}	4.5×10^{-2}	3.2×10^{-4}	3.8×10^{-2}	7.9×10^{-4}	8.9×10^{-4}	9.1×10^{-5}	1.6×10^{-7}
1.7×10^{-1}	6.3×10^{-4}	1.0×10^{-4}	4.3×10^{-2}	3.6×10^{-4}	1.6×10^{-3}	1.5×10^{-3}	9.2×10^{-8}
2.9×10^{-1}	3.2×10^{-3}	5.2×10^{-3}	1.1×10^{-1}	5.0×10^{-3}	5.8×10^{-2}	3.1×10^{-4}	1.7×10^{-4}
5.6×10^{-2}	3.9×10^{-2}	9.3×10^{-4}	1.2×10^{-1}	1.8×10^{-3}	8.5×10^{-4}	1.2×10^{-3}	1.6×10^{-4}
2.1×10^{-2}	3.6×10^{-2}	7.7×10^{-4}	9.2×10^{-3}	5.9×10^{-4}	5.3×10^{-5}	3.6×10^{-3}	3.9×10^{-6}
7.5×10^{-2}	5.2×10^{-2}	3.9×10^{-4}	4.5×10^{-2}	2.6×10^{-4}	1.7×10^{-4}	4.2×10^{-3}	1.6×10^{-4}

Table A9: Closed-loop MSE on the *out-dis-test* data from ten different training runs for each model for the *Langmuir* isotherm.

TCN	ConvLSTM	DISTANA	CNN-NODE	FNO	PINN	PhyDNet	FINN
2.0×10^{-1}	4.9×10^{-2}	3.6×10^{-4}	4.4×10^{-2}	1.4×10^{-2}	N/A	9.4×10^{-3}	1.1×10^{-4}
1.5×10^{-1}	1.2×10^{-1}	2.2×10^{-3}	5.6×10^{-2}	2.6×10^{-2}	N/A	1.3×10^{-2}	5.9×10^{-7}
1.4×10^{-1}	1.6×10^{-4}	2.1×10^{-3}	3.8×10^{-1}	1.7×10^{-2}	N/A	1.3×10^{-2}	2.9×10^{-7}
1.2×10^{-1}	9.9×10^{-2}	3.7×10^{-3}	2.0×10^{-1}	1.1×10^{-2}	N/A	1.2×10^{-2}	1.2×10^{-4}
1.8×10^{-1}	8.3×10^{-2}	3.4×10^{-4}	3.9×10^{-2}	1.3×10^{-2}	N/A	1.2×10^{-2}	2.0×10^{-7}
1.6×10^{-1}	2.6×10^{-3}	9.8×10^{-4}	1.6×10^{-2}	2.8×10^{-2}	N/A	1.2×10^{-2}	4.5×10^{-8}
2.4×10^{-1}	1.6×10^{-3}	1.3×10^{-2}	9.9×10^{-2}	4.2×10^{-3}	N/A	1.6×10^{-2}	1.3×10^{-4}
1.1×10^{-1}	8.8×10^{-2}	3.4×10^{-3}	8.2×10^{-2}	1.6×10^{-2}	N/A	1.5×10^{-2}	1.2×10^{-4}
3.3×10^{-2}	7.6×10^{-2}	6.2×10^{-3}	8.8×10^{-3}	2.4×10^{-2}	N/A	1.5×10^{-2}	2.3×10^{-6}
1.2×10^{-1}	1.1×10^{-1}	1.1×10^{-3}	4.4×10^{-2}	1.3×10^{-2}	N/A	1.4×10^{-2}	1.2×10^{-4}

Appendix B Complete Plots of Results

In this appendix, the complete plots for all predictions are presented to complement the discussion in section 4.1.2. For the dataset generated with the linear sorption isotherm, Figure B1 and Figure B2 show the plots of the best model predictions of the *in-dis-test* data for the dissolved and total concentration, respectively. Figure B3 and Figure B4 show the plots of the best model predictions of the *out-dis-test* data for the dissolved and total concentration, respectively. Figure B5 and Figure B6 show the plots of the model predictions with the confidence interval of the *in-dis-test* data for the dissolved and total concentration, respectively. Figure B7 and Figure B8 show the plots of the model predictions with the confidence interval of the *out-dis-test* data for the dissolved and total concentration, respectively.

For the dataset generated with the Freundlich sorption isotherm, Figure B9 and Figure B10 show the plots of the best model predictions of the *in-dis-test* data for the dissolved and total concentration, respectively. Figure B11 and Figure B12 show the plots of the best model predictions of the *out-dis-test* data for the dissolved and total concentration, respectively. Figure B13 and Figure B14 show the plots of the model predictions with the confidence interval of the *in-dis-test* data for the dissolved and total concentration, respectively. Figure B15 and Figure B16 show the plots of the model predictions with the confidence interval of the *out-dis-test* data for the dissolved and total concentration, respectively.

For the dataset generated with the Langmuir sorption isotherm, Figure B17 shows the plots of the best model predictions of the *in-dis-test* data for the total concentration. Figure B18 shows the plots of the best model predictions of the *out-dis-test* data for the total concentration. Figure B19 shows the plots of the model predictions with the confidence interval of the *in-dis-test* data for the total concentration. Figure B20 show the plots of the model predictions with the confidence interval of the *out-dis-test* data for the total concentration.

Acknowledgments

This work is funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 – 390740016 as well as EXC 2064 – 390727645. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). Moreover, we thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Matthias Karlbauer. All codes necessary to generate data and reproduce results can be found in our repository `https://github.com/CognitiveModeling/finn`.

References

Al-Ghouti, M. A., & Da'ana, D. A. (2020). Guidelines for the use and interpretation of adsorption isotherm models: A review. *Journal of Hazardous Materials*, *393*, 122383. Retrieved from `https://www.sciencedirect.com/science/article/pii/S030438942030371X` doi: https://doi.org/10.1016/j.jhazmat.2020.122383

Allen, S. M., & Cahn, J. W. (1979). A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta Metallurgica*, *27*(6), 1085-1095. Retrieved from `https://www.sciencedirect.com/science/article/pii/0001616079901962` doi: https://doi.org/10.1016/0001-6160(79)90196-2

Almqvist, O. (2019). *A comparative study between algorithms for time series forecasting on customer prediction: An investigation into the performance of arima, rnn, lstm, tcn and hmm.*

Arfken, G., Weber, H., & Harris, F. (2013). *Mathematical methods for physicists: A comprehensive guide*. Elsevier Science. Retrieved from https://books.google.de/books?id=qLFo_Z-PoGIC

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Bardenet, R., Doucet, A., & Holmes, C. C. (2017). On Markov chain Monte Carlo methods for tall data. *Journal of Machine Learning Research*, *18*, 47:1-47:43.

Bar-Sinai, Y., Hoyer, S., Hickey, J., & Brenner, M. (2019). Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, *116*(31), 15344–15349.

Battaglia, P. W., Hamrick, J., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., ... others (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D., & Kavukcuoglu, K. (2016). Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, *112*(518), 859-877. Retrieved from https://doi.org/10.1080/01621459.2017.1285773 doi: 10.1080/01621459.2017.1285773

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32nd international conference on international conference on machine learning - volume 37* (p. 1613–1622).

Brandstetter, J., Worrall, D. E., & Welling, M. (2022). Message passing neural PDE solvers. In *International conference on learning representations.*

Brown, G. H., Brooks, M. C., Wood, A. L., Annable, M. D., & Huang, J. (2012). Aquitard contaminant storage and flux resulting from dense nonaqueous phase liquid source zone dissolution and remediation. *Water Resources Research*, *48*(6). Retrieved from https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2011WR011141 doi: https://doi.org/10.1029/2011WR011141

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... others (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Butcher, J. (2008). *Numerical methods for ordinary differential equations*. Wiley.

Chen, R., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*.

Cheng, A. H.-D., & Cheng, D. T. (2005). Heritage and early history of the boundary element method. *Engineering Analysis with Boundary Elements*, *29*(3), 268-302. Retrieved from https://www.sciencedirect.com/science/article/pii/S0955799705000020 doi: https://doi.org/10.1016/j.enganabound.2004.12.001

Chib, S., & Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *The American Statistician*, *49*(4), 327–335. Retrieved from http://www.jstor.org/stable/2684568

Courant, R., Friedrichs, K., & Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, *11*(2), 215-234. doi: 10.1147/rd.112.0215

De Bézenac, E., Pajot, A., & Gallinari, P. (2019). Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, *2019*(12), 124009.

Dwivedi, R., Chen, Y., Wainwright, M. J., & Yu, B. (2019). Log-concave sampling: Metropolis-hastings algorithms are fast. *Journal of Machine Learning Research*, *20*(183), 1–42. Retrieved from http://jmlr.org/papers/v20/19-306.html

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Espeholt, L., Agrawal, S., Sønderby, C., Kumar, M., Heek, J., Bromberg, C., ... Kalchbrenner, N. (2021). Skillful twelve hour precipitation forecasts using large context neural networks. *arXiv preprint arXiv:2111.07470*.

Fetter, C. W. (1999). *Contaminant hydrogeology.* Prentice Hall.

Fornberg, B. (1988). Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, *51*(184), 699–706.

Ghosh, T., Bringedal, C., Helmig, R., & Sekhar, G. R. (2020, November). Upscaled equations for two-phase flow in highly heterogeneous porous media: Varying permeability and porosity. *Advances in Water Resources*, *145*, 103716. Retrieved from `https://doi.org/10.1016/j.advwatres.2020.103716` doi: 10.1016/j.advwatres.2020.103716

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th international conference on machine learning - volume 70* (p. 1263–1272).

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). MIT Press.

Guen, V., & Thome, N. (2020). Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 11474–11484).

Hayduk, W., & Laudie, H. (1974). Prediction of diffusion coefficients for nonelectrolytes in dilute aqueous solutions. *AIChE Journal*, *20*(3), 611-615. Retrieved from `https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690200329` doi: https://doi.org/10.1002/aic.690200329

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

He, Q., Barajas-Solano, D., Tartakovsky, G., & Tartakovsky, A. M. (2020). Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, *141*, 103610. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0309170819311649` doi: https://doi.org/10.1016/j.advwatres.2020.103610

Hendry, M. J., Ranville, J. R., Boldt-Leppin, B. E. J., & Wassenaar, L. I. (2003). Geochemical and transport properties of dissolved organic carbon in a clay-rich aquitard. *Water Resources Research*, *39*(7). Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2002WR001943` doi: https://doi.org/10.1029/2002WR001943

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.* A field guide to dynamical recurrent neural networks. IEEE Press In.

Huang, J., & Goltz, M. N. (2015). Semianalytical solutions for transport in aquifer and fractured clay matrix system. *Water Resources Research*, *51*(9), 7218-7237. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014WR016073` doi: https://doi.org/10.1002/2014WR016073

Höge, M., Guthke, A., & Nowak, W. (2019). The hydrologist's guide to bayesian model selection, averaging and combination. *Journal of Hydrology*, *572*, 96-107. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0022169419301532` doi: https://doi.org/10.1016/j.jhydrol.2019.01.072

IPCC. (2013). *Climate change 2013: The physical science basis. contribution of working group i to the fifth assessment report of the intergovernmental panel on climate change* (T. Stocker et al., Eds.). Cambridge University Press.

Isaacson, E., & Keller, H. (1994). *Analysis of numerical methods.* Dover Publications.

Jin, X., Cai, S., Li, H., & Karniadakis, G. (2021). Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equa-

tions. *Journal of Computational Physics*, *426*, 109951.

Johnson, G. R., Zhang, Z., & Brusseau, M. L. (2003). Characterizing and quantifying the impact of immiscible-liquid dissolution and nonlinear, rate-limited sorption/desorption on low-concentration elution tailing. *Water Resources Research*, *39*(5). Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2002WR001435` doi: https://doi.org/10.1029/2002WR001435

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., & Bennamoun, M. (2022). Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, *17*(2), 29-48. doi: 10.1109/MCI.2022.3155327

Joyce, J. M. (2011). Kullback-Leibler Divergence. In M. Lovric (Ed.), *International encyclopedia of statistical science* (pp. 720–722). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-04898-2_327

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A., Graves, A., & Kavukcuoglu, K. (2016). Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Karlbauer, M., Menge, T., Otte, S., Lensch, H., Scholten, T., Wulfmeyer, V., & Butz, M. (2020). Hidden latent state inference in a spatio-temporal generative model. *arXiv preprint arXiv:2009.09823*.

Karlbauer, M., Otte, S., Lensch, H., Scholten, T., Wulfmeyer, V., & Butz, M. (2019). A distributed neural network architecture for robust non-linear spatio-temporal prediction. *arXiv preprint arXiv:1912.11141*.

Karlbauer, M., Praditia, T., Otte, S., Oladyshkin, S., Nowak, W., & Butz, M. V. (2022). Composing partial differential equations with physics-aware neural networks. In *Proceedings of the 39th international conference on machine learning*. Baltimore, USA.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, *3*(6), 422–440.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. Retrieved from `http://arxiv.org/abs/1412.6980`

Klaasen, G., & Troy, W. (1984). Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. *SIAM Journal on Applied Mathematics*, *44*(1), 96-110. doi: 10.1137/0144008

Koch, J., & Nowak, W. (2015). Predicting dnapl mass discharge and contaminated site longevity probabilities: Conceptual model and high-resolution stochastic simulation. *Water Resources Research*, *51*(2), 806-831. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014WR015478` doi: https://doi.org/10.1002/2014WR015478

Koch, T., Weishaupt, K., Müller, J., Weigand, B., & Helmig, R. (2021, 5). A (dual) network model for heat transfer in porous media. *Transport in Porous Media*. Retrieved from `https://doi.org/10.1007/s11242-021-01602-5` doi: 10.1007/s11242-021-01602-5

Kochenderfer, M., & Wheeler, T. (2019). *Algorithms for optimization*. MIT Press. Retrieved from `https://books.google.de/books?id=uBSMDwAAQBAJ`

Kochkov, D., Smith, J., Alieva, A., Wang, Q., Brenner, M., & Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, *118*(21).

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097–1105.

Kutta, W. (1901). Beitrag zur naherungsweisen integration totaler differentialgle-

ichungen. *Z. Math. Phys.*, *46*, 435–453.

Lake, B. (2019). Compositional generalization through meta sequence-to-sequence learning. In *Advances in neural information processing systems 32* (pp. 9791–9801).

Lake, B., Ullman, T., Tenenbaum, J., & Gershman, S. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*. doi: 10.1017/S0140525X16001837

Lea, C., Vidal, R., Reiter, A., & Hager, G. D. (2016). Temporal convolutional networks: A unified approach to action segmentation. In *European conference on computer vision* (pp. 47–54).

Li, K., & Horne, R. N. (2006). Comparison of methods to calculate relative permeability from capillary pressure in consolidated water-wet porous media. *Water Resources Research*, *42*(6). Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2005WR004482` doi: https://doi.org/10.1029/2005WR004482

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020a). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020b). Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*.

Lienen, M., & Günnemann, S. (2022). Learning the dynamics of physical systems from sparse observations with finite element networks. In *International conference on learning representations.*

Limousin, G., Gaudet, J.-P., Charlet, L., Szenknect, S., Barthès, V., & Krimissa, M. (2007). Sorption isotherms: A review on physical bases, modeling and measurement. *Applied Geochemistry*, *22*(2), 249-275. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0883292706002629` doi: https://doi.org/10.1016/j.apgeochem.2006.09.010

Livingstone, S., & Zanella, G. (2019). *The barker proposal: combining robustness and efficiency in gradient-based mcmc.* Retrieved from `https://arxiv.org/abs/1908.11812` doi: 10.48550/ARXIV.1908.11812

Logan, D. (1992). *A first course in the finite element method.* PWS-Kent Publishing Company. Retrieved from `https://books.google.de/books?id=qCiRQgAACAAJ`

Long, Z., Lu, Y., Ma, X., & Dong, B. (2018). Pde-net: Learning pdes from data. In *International conference on machine learning* (pp. 3208–3216).

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th conference on natural language learning - volume 20* (p. 1–7). USA: Association for Computational Linguistics. Retrieved from `https://doi.org/10.3115/1118853.1118871` doi: 10.3115/1118853.1118871

Marchuk, G. (1974). *Numerical Methods in Weather Prediction.* Elsevier.

Moghadasi, L., Guadagnini, A., Inzoli, F., & Bartosek, M. (2015). Interpretation of two-phase relative permeability curves through multiple formulations and model quality criteria. *Journal of Petroleum Science and Engineering*, *135*, 738-749. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0920410515301534` doi: https://doi.org/10.1016/j.petrol.2015.10.027

Morton, K., & Mayers, D. (1994). *Numerical solution of partial differential equations: An introduction.* Cambridge University Press. Retrieved from `https://books.google.de/books?id=h-s-nwEACAAJ`

Moukalled, F., Mangani, L., & Darwish, M. (2016). *The finite volume method in computational fluid dynamics* (1st ed.). Springer. doi: 10.1007/978-3-319-16874-6

National Toxicology Program, US Department of Health and Human Services. (2021). *15th report on carcinogens* (Tech. Rep.). National Toxicology Program, US Department of Health and Human Services.

Nowak, W. (2000). *Age determination of a tce source zone using solute transport profiles in an underlying clayey aquitard* (Unpublished master's thesis). University of Waterloo.

Nowak, W., & Guthke, A. (2016). Entropy-based experimental design for optimal model discrimination in the geosciences. *Entropy*, *18*(11). doi: 10.3390/e18110409

Oliphant, T. E. (2006). *A Bayesian perspective on estimating mean, variance, and standard-deviation from data.* Brigham Young University.

Pankow, J., & Cherry, J. (1996). *Dense chlorinated solvents and other dnapls in groundwater: History, behavior, and remediation.* Waterloo Press. Retrieved from `https://books.google.de/books?id=OiYfAQAAIAAJ`

Parker, B. L., Cherry, J. A., & Chapman, S. W. (2004). Field study of tce diffusion profiles below dnapl to assess aquitard integrity. *Journal of Contaminant Hydrology*, *74*(1), 197-230. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0169772204000415` doi: https://doi.org/10.1016/j.jconhyd.2004.02.011

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`

Praditia, T., Walser, T., Oladyshkin, S., & Nowak, W. (2020). Improving thermochemical energy storage dynamics forecast with physics-inspired neural network architecture. *Energies*, *13*(15), 3873. doi: 10.3390/en13153873

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686-707. doi: 10.1016/j.jcp.2018.10.045

Rasp, S., Dueben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., & Thuerey, N. (2020). Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, *12*(11), e2020MS002203.

Reuschen, S., Jobst, F., & Nowak, W. (2021). Efficient discretization-independent bayesian inversion of high-dimensional multi-gaussian priors using a hybrid mcmc. *Water Resources Research*, *57*(8), e2021WR030051. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021WR030051` doi: https://doi.org/10.1029/2021WR030051

Runge, C. (1895). Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, *46*(2), 167–178.

Salehi, Y., & Giannacopoulos, D. (2021). Physgnn: A physics-driven graph neural network based model for predicting soft tissue deformation in image-guided neurosurgery. *arXiv preprint arXiv:2109.04352*.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., & Battaglia, P. W. (2020). Learning to simulate complex physics with graph networks. In *International conference on machine learning* (pp. 8459–8468).

Seo, S., Meng, C., & Liu, Y. (2019). Physics-aware difference graph networks for sparsely-observed dynamics. In *International conference on learning representations*.

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., & Woo, W. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., & Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, *33*.

Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., & Barajas-Solano, D. (2020). Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, *56*(5), e2019WR026731. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019WR026731` doi: https://doi.org/10.1029/2019WR026731

Thuerey, N., Holl, P., Mueller, M., Schnell, P., Trost, F., & Um, K. (2021). *Physics-based deep learning*. WWW. Retrieved from `https://physicsbaseddeeplearning.org`

Timms, W. A., Acworth, R. I., Crane, R. A., Arns, C. H., Arns, J.-Y., McGeeney, D. E., ... Cuthbert, M. O. (2018). The influence of syndepositional macropores on the hydraulic integrity of thick alluvial clay aquitards. *Water Resources Research*, *54*(4), 3122-3138. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2017WR021681` doi: https://doi.org/10.1029/2017WR021681

Tran, G., & Ward, R. (2017). Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling & Simulation*, *15*(3), 1108–1129.

Turing, A. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, *237*, 37–72.

Wei, J., & Winter, M. (2017). Stable spike clusters for the one-dimensional gierer–meinhardt system. *European Journal of Applied Mathematics*, *28*(4), 576–635. doi: 10.1017/S0956792516000450

Wilke, C. R., & Chang, P. (1955). Correlation of diffusion coefficients in dilute solutions. *AIChE Journal*, *1*(2), 264-270. Retrieved from `https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690010222` doi: https://doi.org/10.1002/aic.690010222

Winsberg, E. (2003). Simulated experiments: Methodology for a virtual world. *Philosophy of Science*, *70*(1), 105-125. Retrieved from `https://doi.org/10.1086/367872` doi: 10.1086/367872

Wöhling, T., Schöniger, A., Gayler, S., & Nowak, W. (2015). Bayesian model averaging to explore the worth of data for soil-plant model selection and prediction. *Water Resources Research*, *51*(4), 2825-2846. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014WR016292` doi: https://doi.org/10.1002/2014WR016292

World Health Organization, Regional Office for Europe. (2000). 5.15 trichloroethylene [Publications]. In *Air quality guidelines for europe* (2nd ed., p. 115-117). World Health Organization, Regional Office for Europe.

Xu, T., Reuschen, S., Nowak, W., & Hendricks Franssen, H.-J. (2020). Preconditioned crank-nicolson markov chain monte carlo coupled with parallel tempering: An efficient method for bayesian inversion of multi-gaussian log-hydraulic conductivity fields. *Water Resources Research*, *56*(8), e2020WR027110. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020WR027110` doi: https://doi.org/10.1029/2020WR027110

Yin, Y., Guen, V., Dona, J., Ayed, I., de Bézenac, E., Thome, N., & Gallinari, P. (2020). Augmenting physical models with deep networks for complex dynamics forecasting. *arXiv preprint arXiv:2010.04456*.

Zhuang, J., Kochkov, D., Bar-Sinai, Y., Brenner, M., & Hoyer, S. (2021). Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids*, *6*(6), 064605.

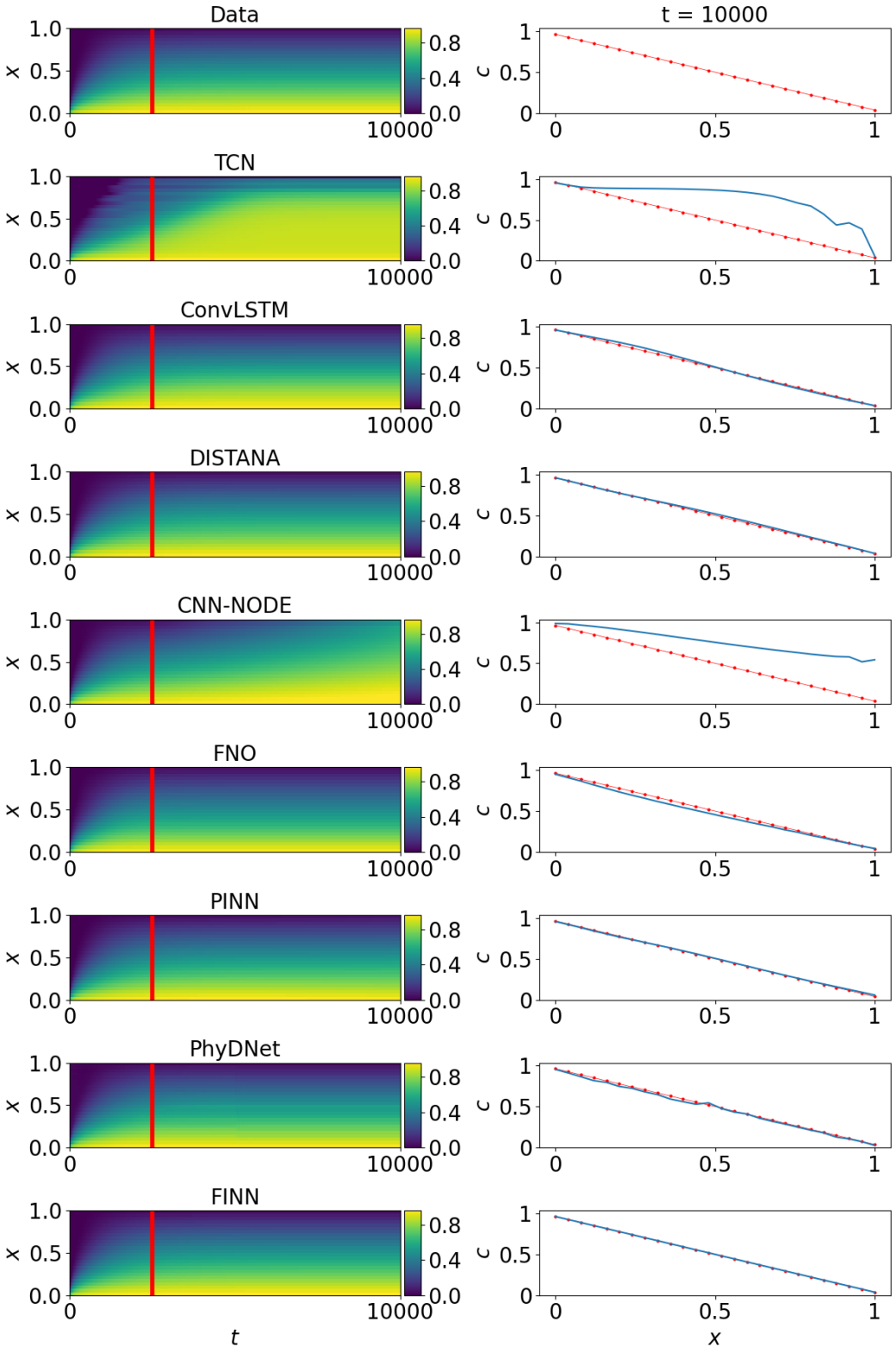

Figure B1: Plots of the dissolved concentration data generated with the linear isotherm (red) and *in-dis-test* prediction (blue) using different models. The left column shows the solution over x and t (red lines mark the transition from *train* to *in-dis-test*), the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

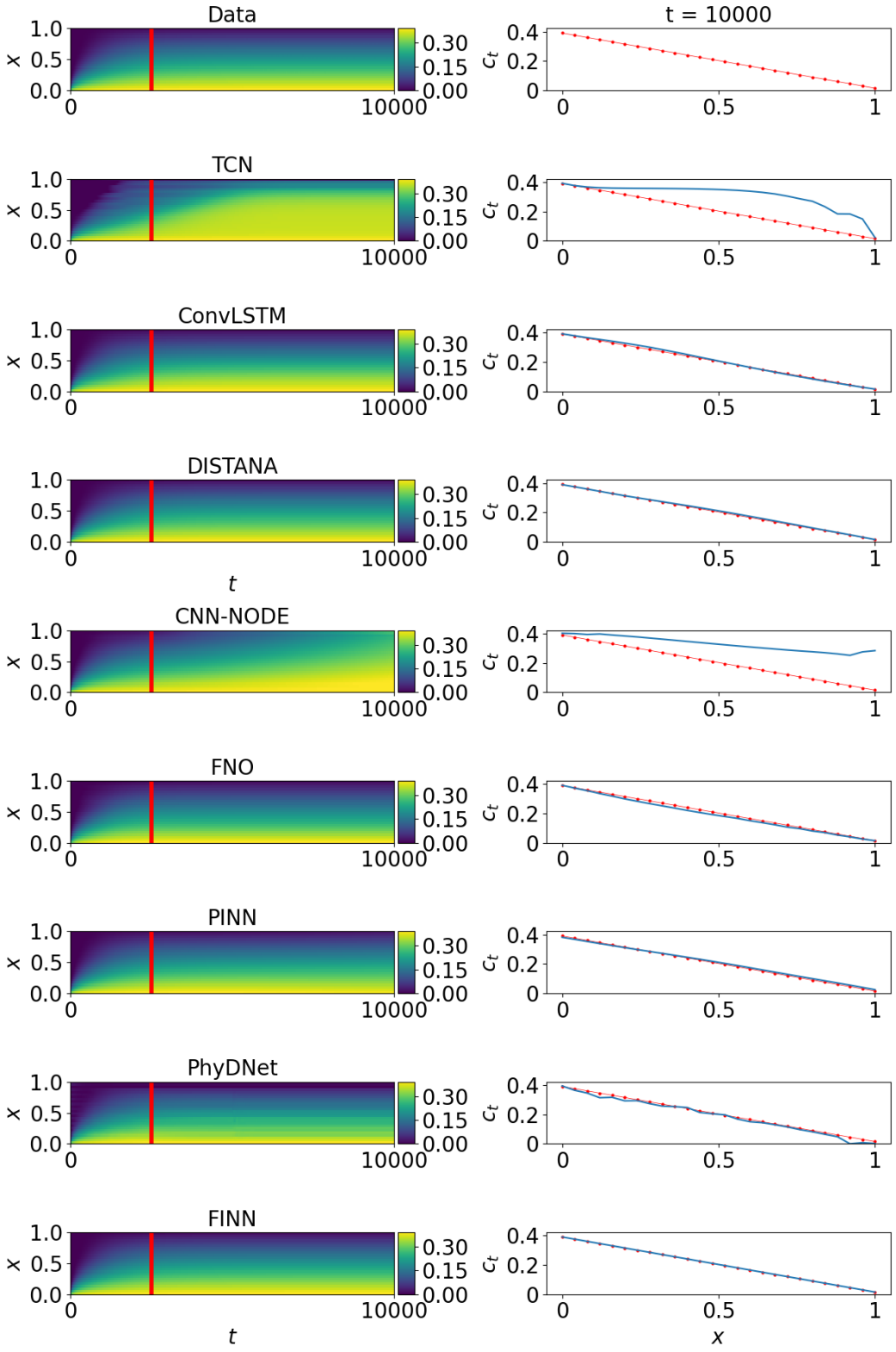

Figure B2: Plots of the total concentration data generated with the linear isotherm (red) and *in-dis-test* prediction (blue) using different models. The left column shows the solution over x and t (red lines mark the transition from *train* to *in-dis-test*), the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

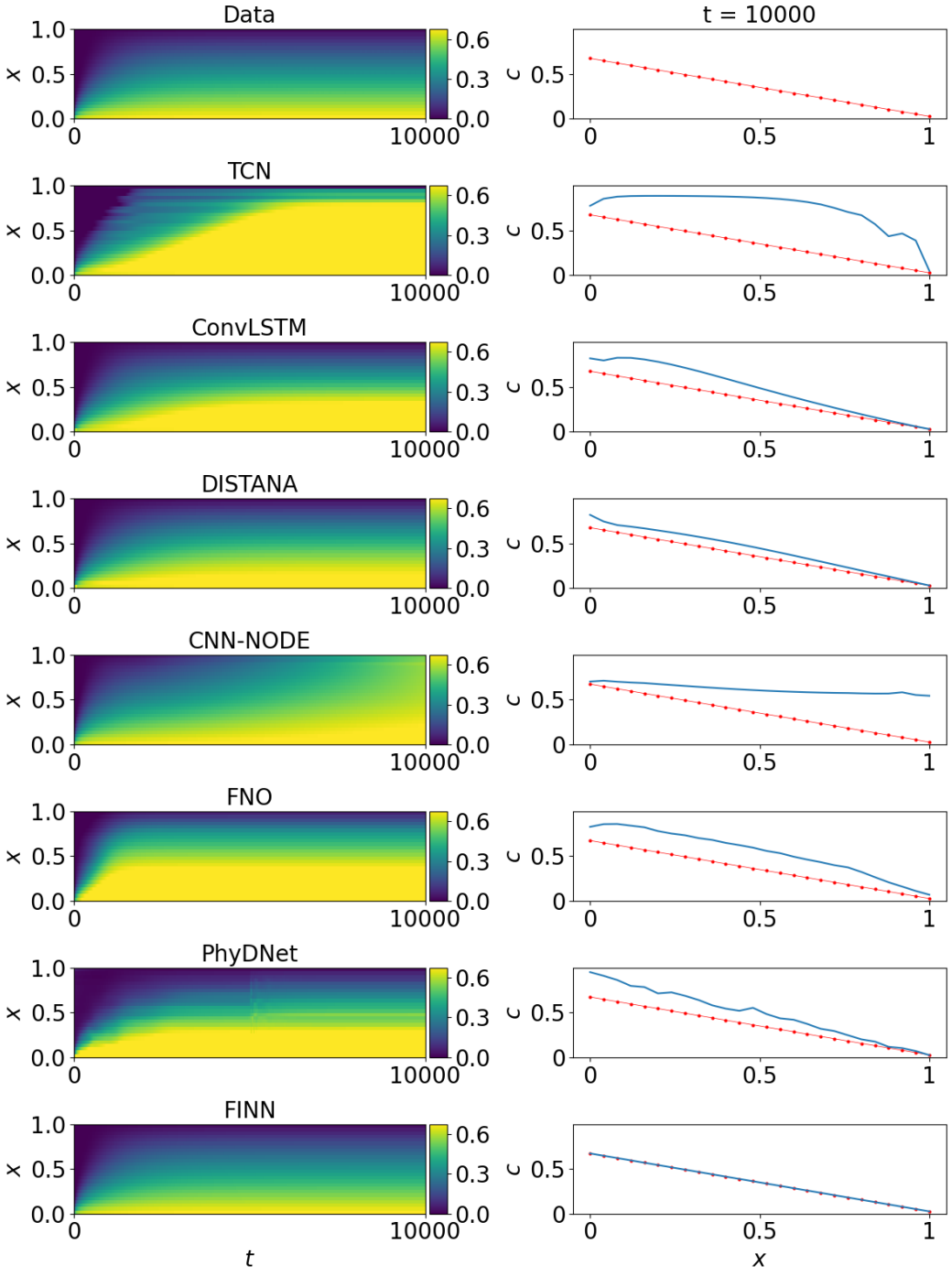

Figure B3: Plots of the dissolved concentration data generated with the linear isotherm (red) and *out-dis-test* prediction (blue) using different models. The left column shows the solution over x and t, the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

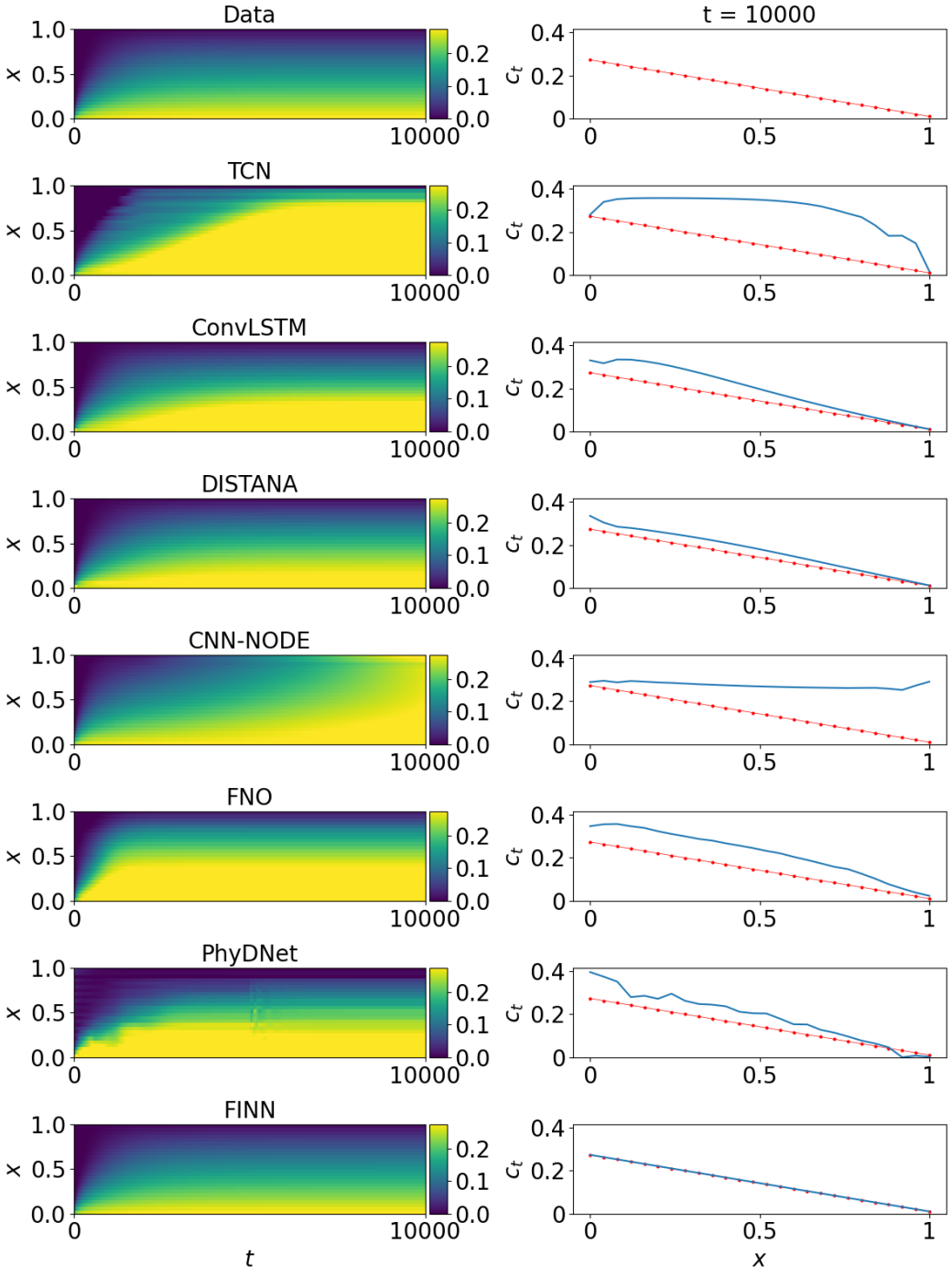

Figure B4: Plots of the total concentration data generated with the linear isotherm (red) and *out-dis-test* prediction (blue) using different models. The left column shows the solution over x and t, the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

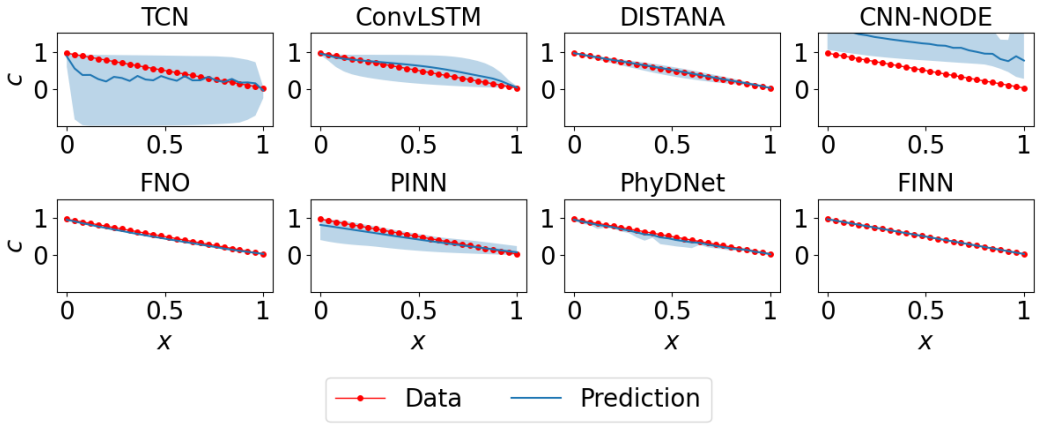

Figure B5: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration generated using the linear isotherm at $t = 10\,000$ for the *in-dis-test* dataset.

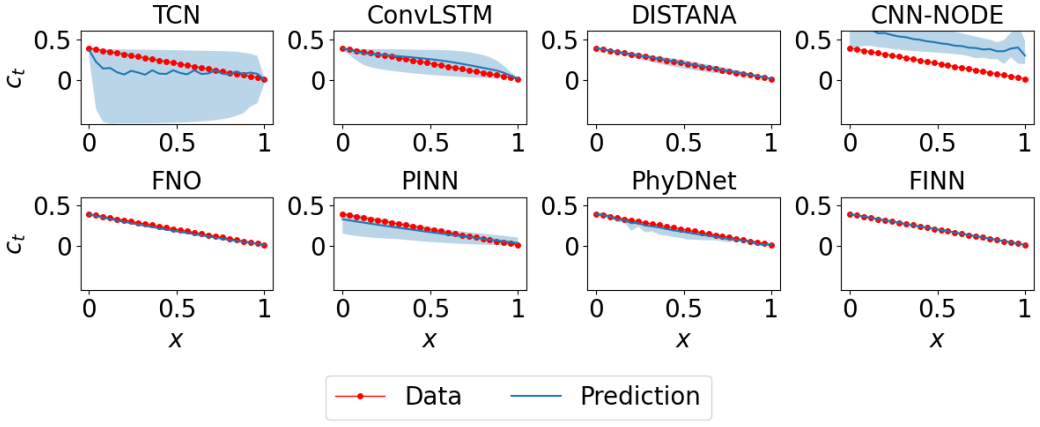

Figure B6: Prediction mean over ten different trained models (with 95% confidence interval) of the total concentration generated using the linear isotherm at $t = 10\,000$ for the *in-dis-test* dataset.

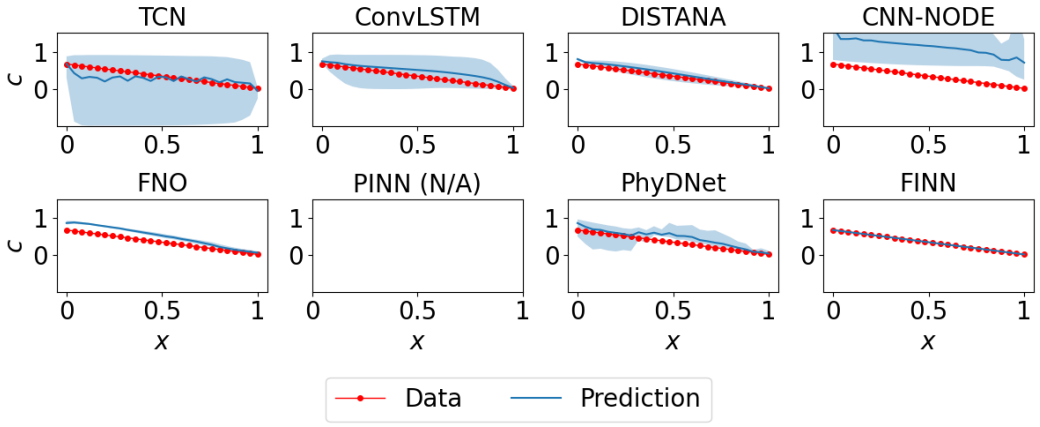

Figure B7: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration generated using the linear isotherm at $t = 10\,000$ for the *out-dis-test* dataset.

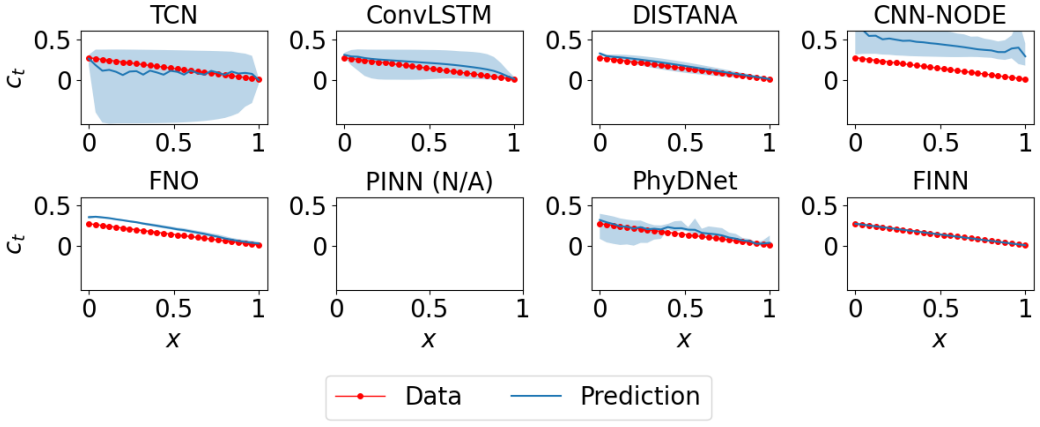

Figure B8: Prediction mean over ten different trained models (with 95% confidence interval) of the total concentration generated using the linear isotherm at $t = 10\,000$ for the *out-dis-test* dataset.

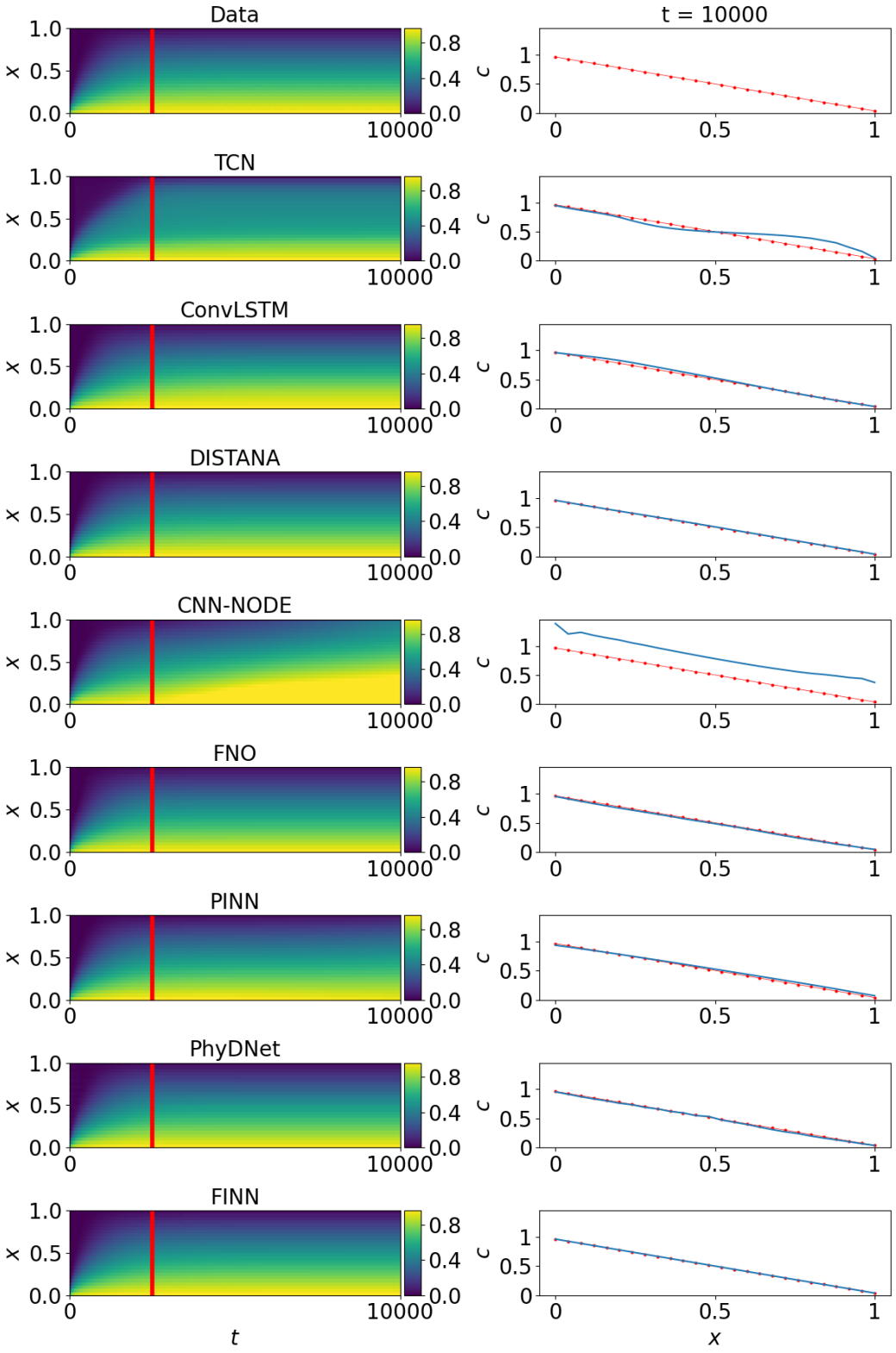

Figure B9: Plots of the dissolved concentration data generated with the Freundlich isotherm (red) and *in-dis-test* prediction (blue) using different models. The left column shows the solution over x and t (red lines mark the transition from *train* to *in-dis-test*), the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

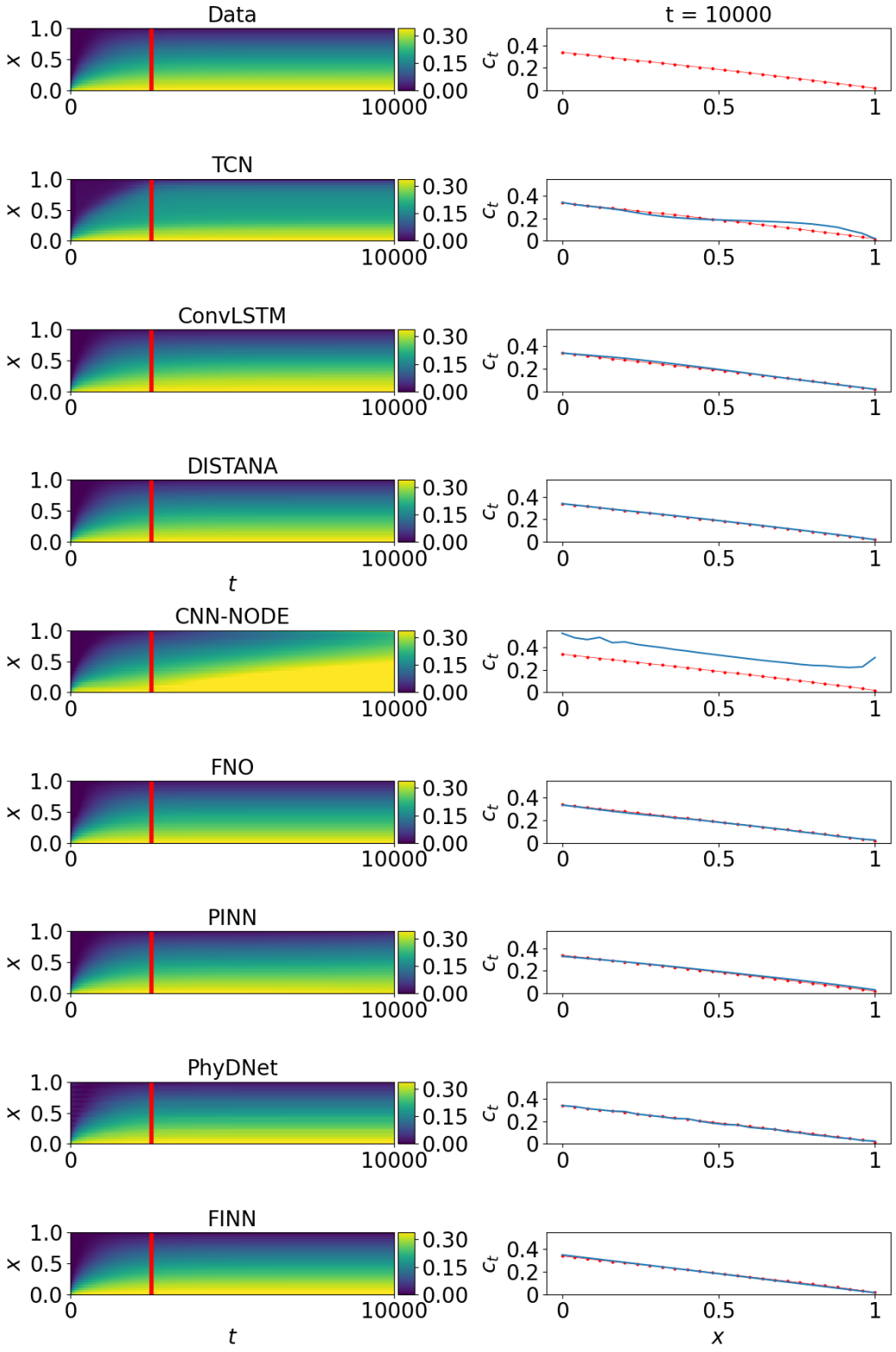

Figure B10: Plots of the total concentration data generated with the Freundlich isotherm (red) and *in-dis-test* prediction (blue) using different models. The left column shows the solution over x and t (red lines mark the transition from *train* to *in-dis-test*), the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

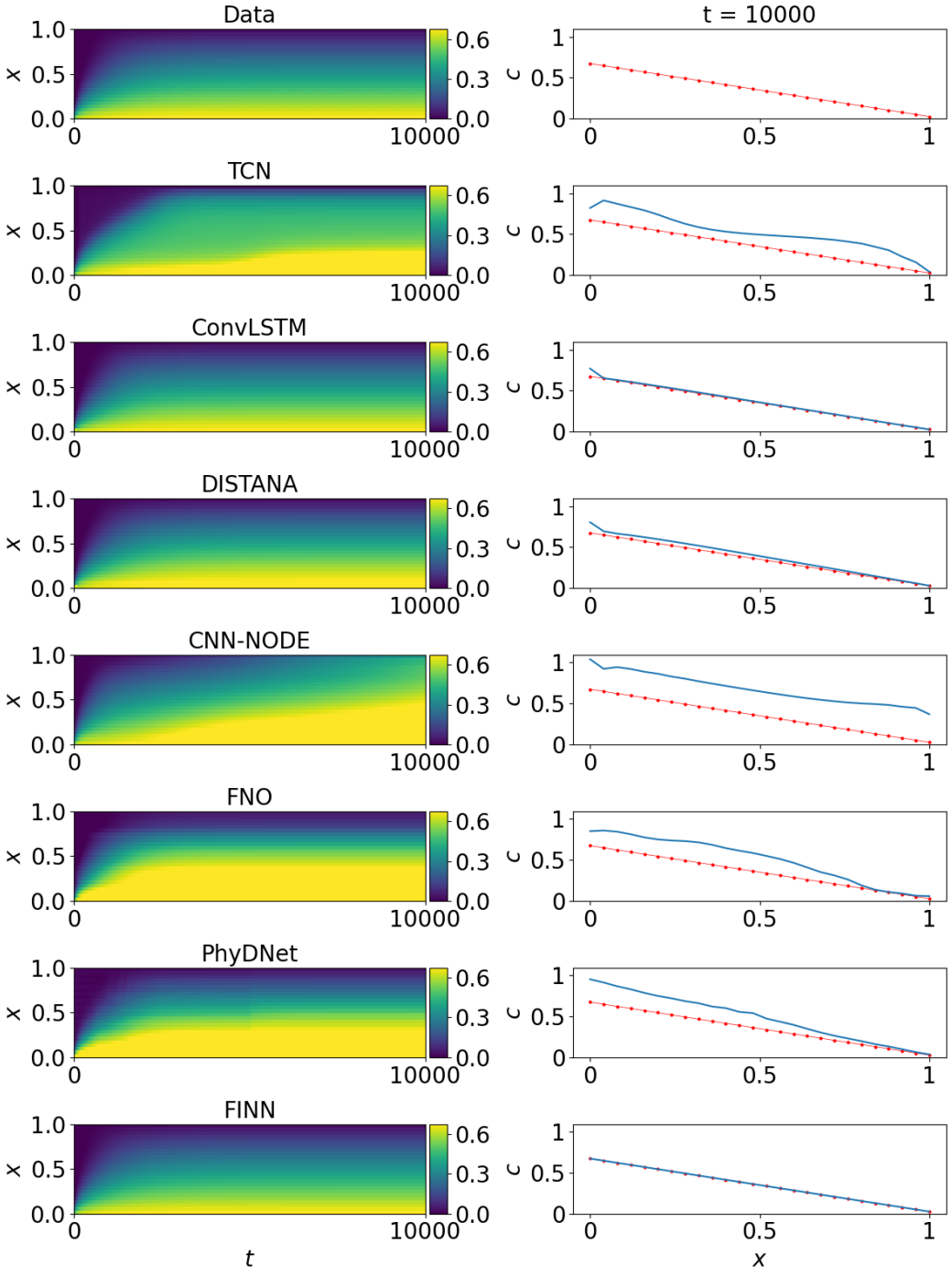

Figure B11: Plots of the dissolved concentration data generated with the Freundlich isotherm (red) and *out-dis-test* prediction (blue) using different models. The left column shows the solution over x and t, the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

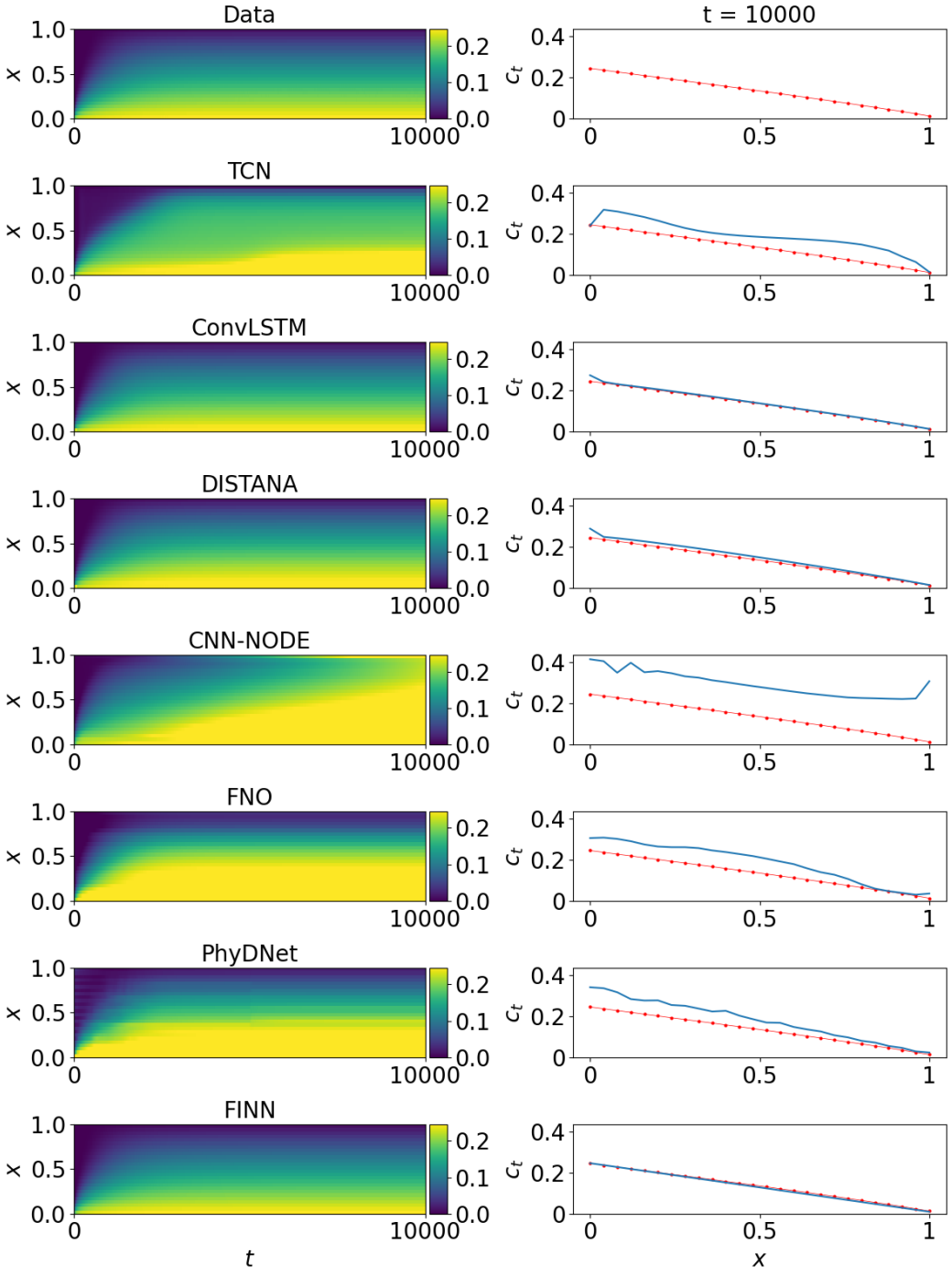

Figure B12: Plots of the total concentration data generated with the Freundlich isotherm (red) and *out-dis-test* prediction (blue) using different models. The left column shows the solution over x and t, the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

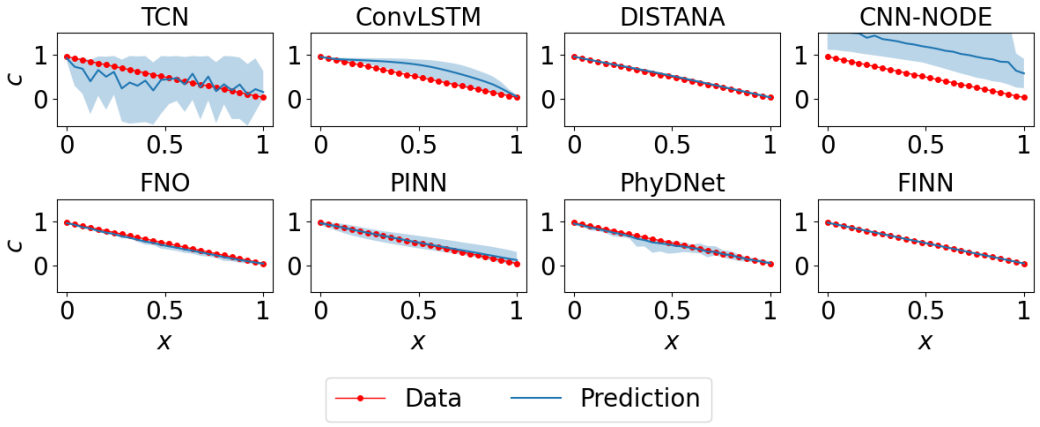

Figure B13: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration generated using the Freundlich isotherm at $t = 10\,000$ for the *in-dis-test* dataset.

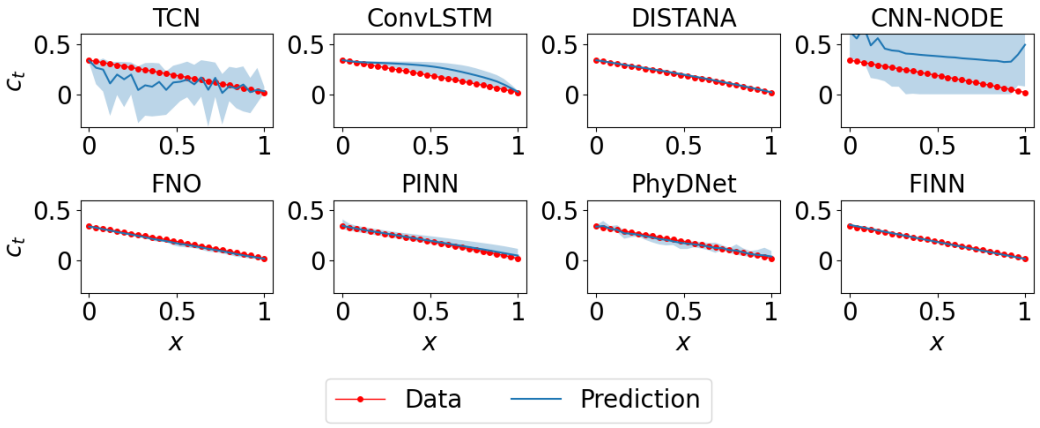

Figure B14: Prediction mean over ten different trained models (with 95% confidence interval) of the total concentration generated using the Freundlich isotherm at $t = 10\,000$ for the *in-dis-test* dataset.

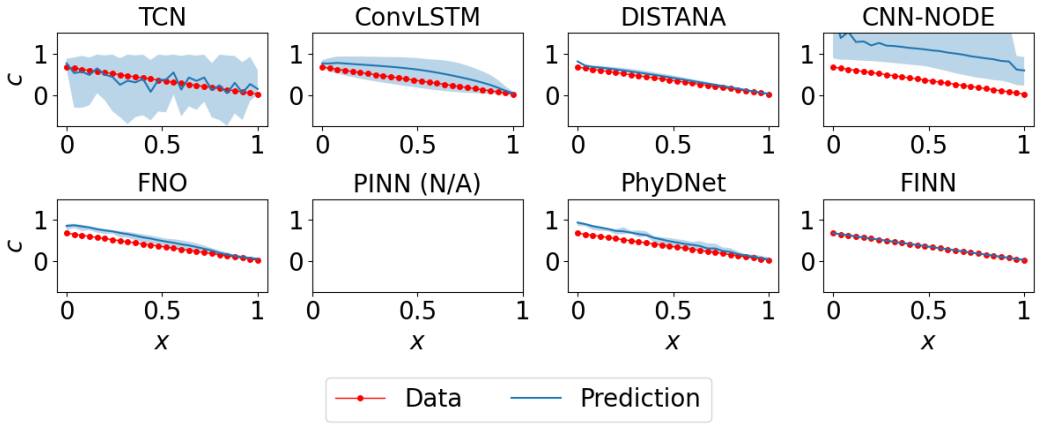

Figure B15: Prediction mean over ten different trained models (with 95% confidence interval) of the dissolved concentration generated using the Freundlich isotherm at $t = 10\,000$ for the *out-dis-test* dataset.

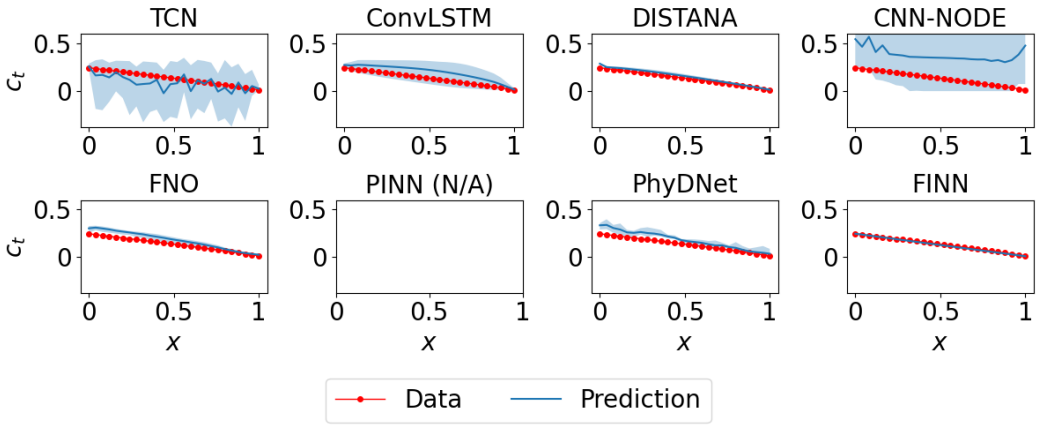

Figure B16: Prediction mean over ten different trained models (with 95% confidence interval) of the total concentration generated using the Freundlich isotherm at $t = 10\,000$ for the *out-dis-test* dataset.

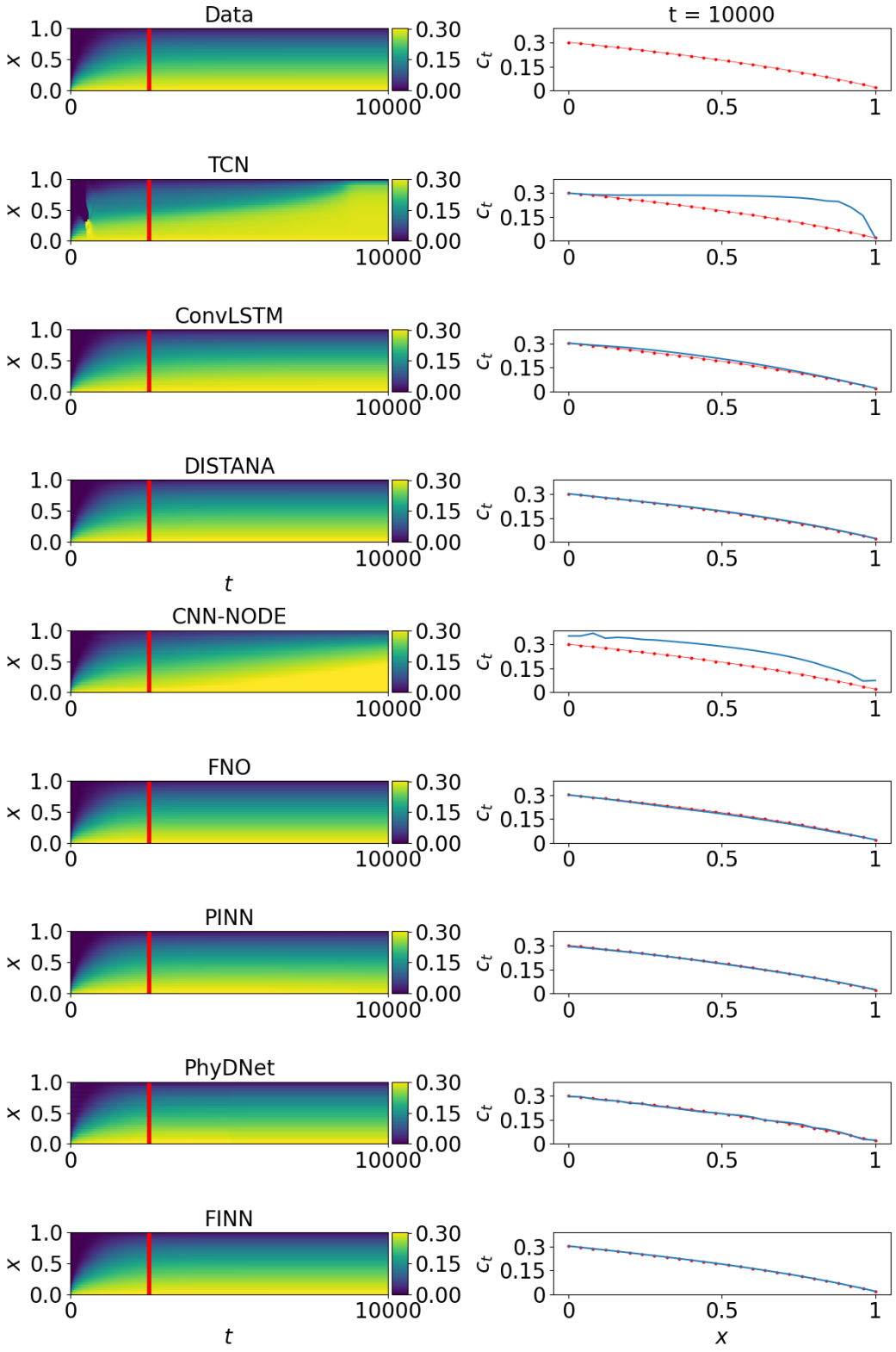

Figure B17: Plots of the total concentration data generated with the Langmuir isotherm (red) and *in-dis-test* prediction (blue) using different models. The left column shows the solution over x and t (red lines mark the transition from *train* to *in-dis-test*), the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

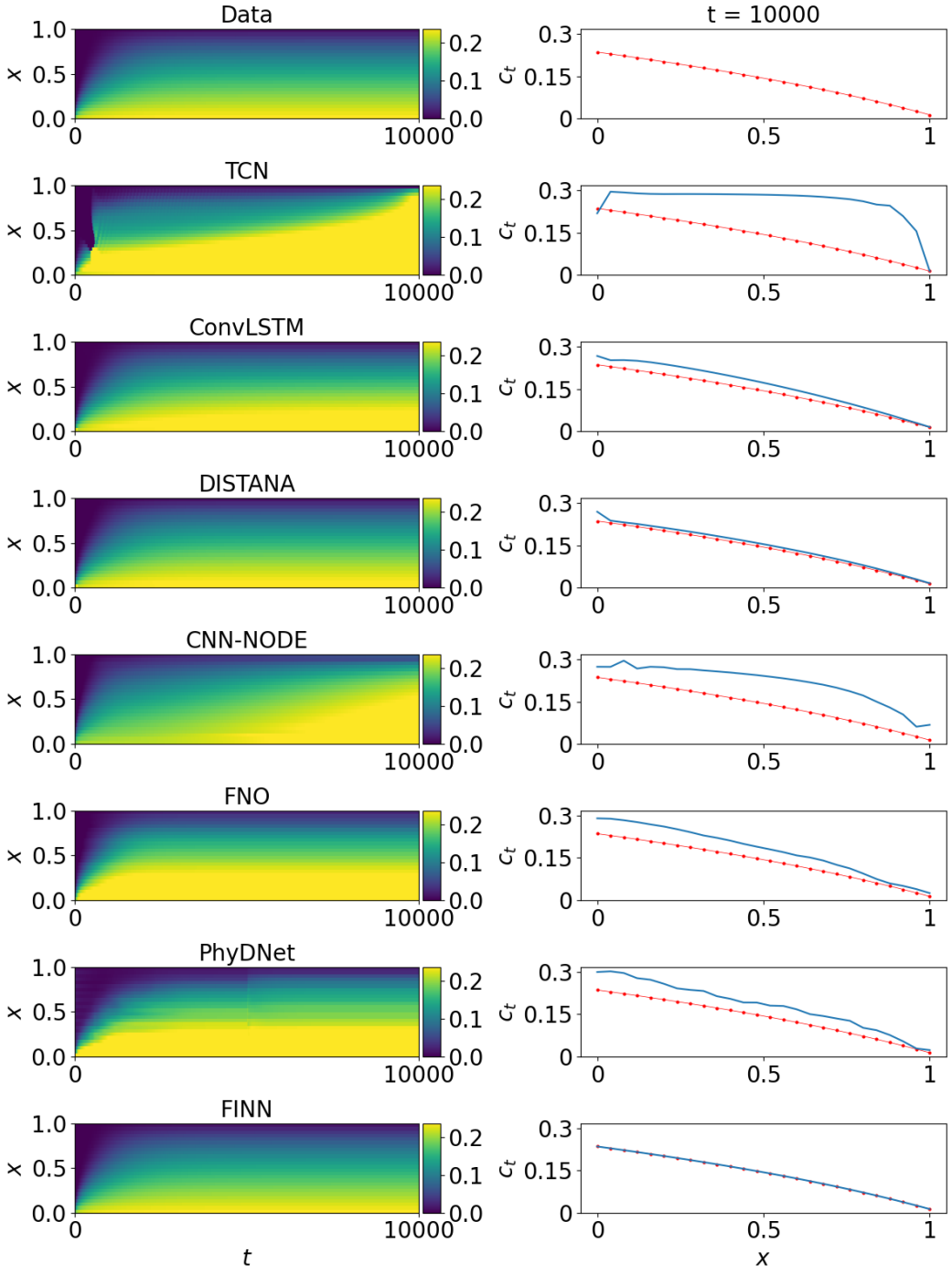

Figure B18: Plots of the total concentration data generated with the Langmuir isotherm (red) and *out-dis-test* prediction (blue) using different models. The left column shows the solution over x and t, the right column visualizes the best solution of each model distributed in x at $t = 10\,000$.

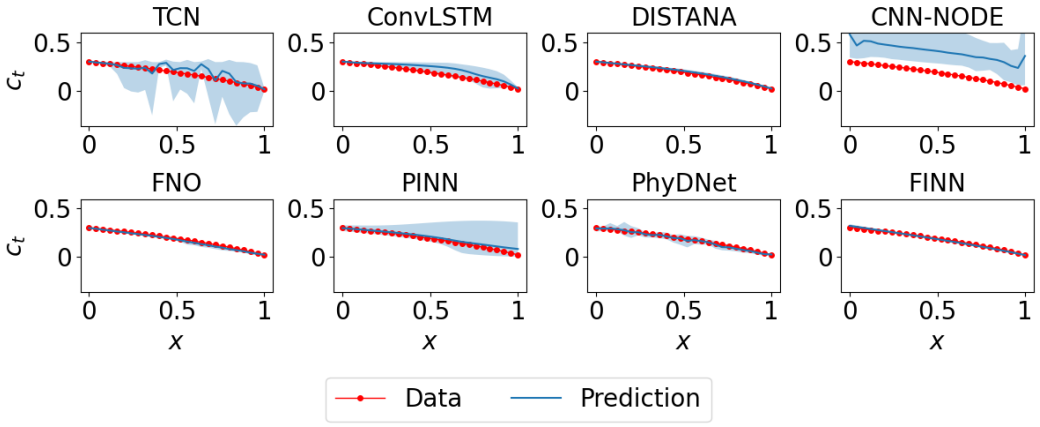

Figure B19: Prediction mean over ten different trained models (with 95% confidence interval) of the total concentration generated using the Langmuir isotherm at $t = 10\,000$ for the *in-dis-test* dataset.

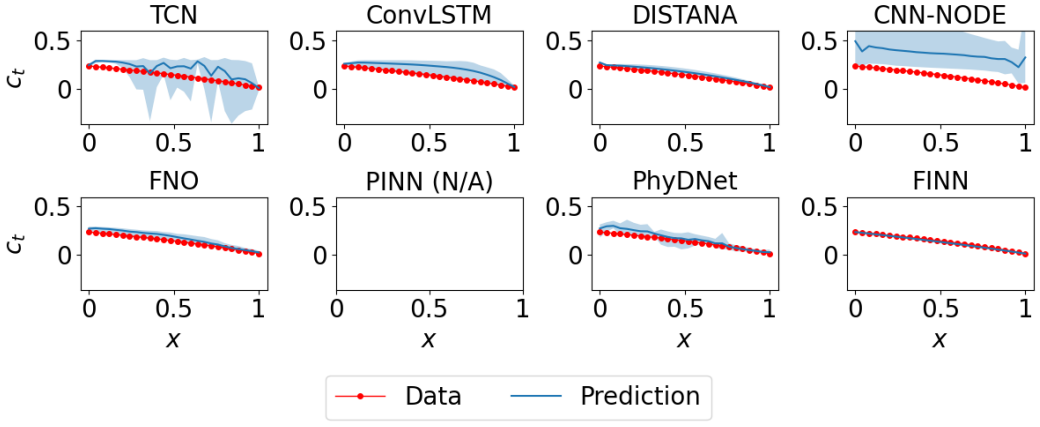

Figure B20: Prediction mean over ten different trained models (with 95% confidence interval) of the total concentration generated using the Langmuir isotherm at $t = 10\,000$ for the *out-dis-test* dataset.