FINITE VOLUME NEURAL NETWORK: MODELING SUBSURFACE CONTAMINANT TRANSPORT

Anonymous authors
Paper under double-blind review

ABSTRACT

Data-driven modeling of spatiotemporal physical processes with general deep learning methods is a highly challenging task. It is further exacerbated by the limited availability of data, leading to poor generalizations in standard neural network models. To tackle this issue, we introduce a new approach called the Finite Volume Neural Network (FINN). The FINN method adopts the numerical structure of the well-known Finite Volume Method for handling partial differential equations, so that each quantity of interest follows its own adaptable conservation law, while it concurrently accommodates learnable parameters. As a result, FINN enables better handling of fluxes between control volumes and therefore proper treatment of different types of numerical boundary conditions. We demonstrate the effectiveness of our approach with a subsurface contaminant transport problem, which is governed by a non-linear diffusion-sorption process. FINN does not only generalize better to differing boundary conditions compared to other methods, it is also capable to explicitly extract and learn the constitutive relationships (i.e. the retardation factor). More importantly, FINN shows excellent generalization capability not only when applied to synthetic datasets, but also on real and sparse experimental data, thus underlining its relevance as a data-driven modeling tool.

1 INTRODUCTION

Training neural networks with additional physical information has been shown to improve their generalization capabilities, particularly when predicting physical processes. In the Physics Informed Neural Network (PINN) framework (Karpatne et al., 2017; 2018; Tartakovsky et al., 2018; Raissi et al., 2019; Wang et al., 2020), the neural network prediction $u(x,t)$ is defined to be an explicit function of space x and time t. Furthermore, calculations of derivatives, such as $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial t}$ are required for formulating the loss function. However, when the available training data are concentrated on a single location x or time t, the approximation of the derivatives $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial t}$ in current techniques deteriorates due to (a) insufficient information provided by the data and (b) the lack of structural explainability of the framework itself. To address these issues from a structural point of view, several works have been conducted in the literature recently. One architecture, namely the Distributed Spatiotemporal Artificial Neural Network (DISTANA, Karlbauer et al., 2019), uses translational invariant Prediction Kernels (PKs) and Transition Kernels (TKs) to process the temporal and spatial correlation of the data, respectively. Another method, called the Universal Differential Equation method (UDE, Rackauckas et al., 2020), combines Convolutional Neural Networks (CNNs, LeCun et al., 1999) with Neural Ordinary Differential Equations (NODEs, Chen et al., 2018), to process spatial correlations and temporal correlations, respectively. Despite promising results shown by these methods, they still suffer from unreliable flux handling (i.e. the physical fluxes are not guaranteed to be conservative). Consequently, the methods mentioned above lack means to properly treat different types of boundary conditions.

To handle fluxes more robustly and improve generalization within a physical domain, we propose a Finite Volume Neural Network (FINN). The FINN method is a hybrid model, which capitalizes on the structural knowledge of the well-known Finite Volume Method (FVM, Moukalled et al., 2016), and the flexibility as well as the learning abilities of Artificial Neural Networks (ANNs), more specifically NODEs. As a consequence, the FINN structure can properly treat different types of boundary conditions and ensures conservation of the quantities of interest. Moreover, we show that FINN is able to reconstruct the full field solution (for all x and t) even when trained with only

partial information (e.g. at a single point x or t). Additionally, the structure of FINN facilitates learning and extracting constitutive relationships and/or reaction terms, and, consequently, shows exceptional generalization capabilities and develops explainable knowledge structures.

2 METHODS

In this work, we focus on modeling spatiotemporal physical processes, namely processes that scientists try to model with Partial Differential Equations (PDEs), such as diffusion-type problems. They can be generally written mathematically as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D(u)\frac{\partial u}{\partial x}\right) + q(u), \tag{1}$$

where u is the quantity of interest, t is time, D is the diffusion coefficient and q is the source/sink term. Usually, the FVM discretizes Equation 1 implicitly in space and explicitly in time, leading to a simplified definition:

$$\frac{\partial u_i^{(t+1)}}{\partial t} = f\left(u_{i-1}^{(t)}, u_i^{(t)}, u_{i+1}^{(t)}, t\right), \tag{2}$$

where $u_i^{(t)}$ denotes u at control volume i and time step t. In other words, the time derivative of u depends on its own value, the values at the neighboring control volumes, and time. For brevity, we drop the time index in later definitions.

In FINN, we introduce Flux Kernels $\mathcal{F}$, which take the input of u_{i-1}, u_i, and u_{i+1} to approximate the divergence part in Equation 1 for each control volume i:

$$\mathcal{F}_i = \Phi_\theta(u_{i-1}, u_i, u_{i+1}) = \sum_s fk_s \approx \oint_S \left(D(u)\frac{\partial u}{\partial x}\cdot\hat{n}\right) ds. \tag{3}$$

Here, fk_s is the flux calculated at the boundary surface s between control volume i and one of its neighboring control volumes (see Figure 1), being learned by an ANN Φ with parameters θ. Calculation of fluxes at each boundary surface s signifies that the numerical stencil at each s should be $[-1, 1]$, which corresponds to $[u_i, u_{i-1}]$ and $[u_i, u_{i+1}]$ in one-dimensional problems. When the fluxes are integrated in each control volume, the summation of the numerical stencil will lead to the classical one-dimensional numerical Laplacian with $[1, -2, 1]$ corresponding to $[u_{i-1}, u_i, u_{i+1}]$. Furthermore, the structure of the Flux Kernel also enables straightforward handling of different types of boundary conditions. With a Dirichlet boundary condition u_b, we can pad its value $u = u_b$ next to the domain boundary. With a Neumann boundary condition ν, we can easily set the flux at the corresponding domain boundary fk_s to be equal to ν. Additionally, the numerical stencil and the diffusion coefficient D can be set as learnable parameters in the Flux Kernels. Moreover, if the diffusion coefficient depends on the quantity of interest u according to a hidden constitutive relationship, Flux Kernels are also capable of learning this relationship.

The State Kernels $\mathcal{S}$ then take the output of the Flux Kernels and approximate $\frac{\partial u}{\partial t}$ while also learning the source/sink term q (whenever necessary). The source/sink term can be further defined as a function of u, which is also learnable by the State Kernels. Formally, each State Kernel can be written as an approximation of time derivative in Equation 1 for each control volume i:

$$\mathcal{S}_i = \mathcal{F}_i + \Phi_\psi(u) \approx \frac{\partial u_i}{\partial t}, \tag{4}$$

where Φ is parameterized by ψ. The outputs of State Kernels are then integrated by an ODE solver to solve for $u^{(t+1)}$. One of the benefits of State Kernels is to enable separate calculations for different quantities of interest, while the divergence (flux) can be calculated based on the same variable. This ensures that each quantity of interest follows its own conservation law.

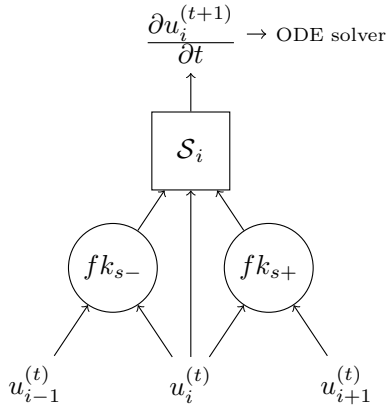

Figure 1: Illustration of the Flux and State Kernels in the FINN.

3 EXPERIMENT

For demonstration purposes, we choose an application with a subsurface contaminant transport problem. We assess the performance of FINN not only using synthetic simulation data, but also real experimental data. The contaminant transport is characterized by the non-linear diffusion-sorption equation in a homogeneous porous medium:

$$R\frac{\partial c}{\partial t} = D_e\frac{\partial^2 c}{\partial x^2}, \tag{5}$$

where c denotes the concentration of trichloroethylene (TCE) dissolved in the fluid, D_e denotes the effective diffusion coefficient, and R denotes the retardation factor (representing sorption), which is a function of c. Equation 5 is subject to a Dirichlet boundary condition at $x = 0$ and a Cauchy boundary condition at $x = L$ (i.e. the top and bottom ends of the field) and an initial condition $c(t = 0) = 0$. Using the definition of retardation factors, we can also calculate the total TCE concentration c_t (both in the fluid and adsorbed in the solid)

$$\frac{\partial c_t}{\partial t} = D_e\phi\frac{\partial^2 c}{\partial x^2}, \tag{6}$$

where ϕ is the porosity of the medium (i.e. the core samples). More detailed information on the experiment can be found elsewhere (Nowak & Guthke, 2016).

4 RESULTS AND DISCUSSION

As the first step, we train and test FINN using numerically generated synthetic datasets. Both train and test datasets are discretized into 26 control volumes and 2 000 time steps. We train FINN using time step $1 - 501$ of the train dataset, and to test the generalization capability, we use the trained network to extrapolate until time step 2 000. Additionally, we test the trained FINN prediction against a completely unseen test dataset obtained at different boundary conditions. The Dirichlet boundary condition values at the top boundary c_s are 1.0 kg/m^3 and 0.7 kg/m^3 for the train and test dataset, respectively. We also compare the train and test Mean Squared Error (MSE) value with other known methods, such as TCN (Kalchbrenner et al., 2016), ConvLSTM (Shi et al., 2015), and DISTANA (Karlbauer et al., 2019).

The comparison in Table 1 shows that FINN appropriately generalizes when tested against a different boundary condition. Even though all methods have comparable performance during training, the predictions of TCN, ConvLSTM, and DISTANA deteriorate when the knowledge gained from the training data needs to be extrapolated and to predict unseen test data. This appears to be caused by the fact that they lack proper boundary condition handling. More detailed information can be found in the appendices regarding the benchmark test dataset (Appendix A), the training and test results (Appendix B), as well as the model setup (Appendix C).

As the second step, we apply FINN to real experimental data. The experimental data were collected from three different core samples, namely core samples #1, #2, and #2B (see Appendix D). In this setup, we train FINN using data that originate exclusively from core sample #2. More specifically, we only use the breakthrough curve data of c in the tailwater to train FINN. This means that the data only provides partial information at $x = L|_{0\leq t\leq t_{end}}$. The trained FINN has higher accuracy with $\mathrm{MSE} = 4.84 \times 10^{-4}$ compared to the calibrated numerical PDE model with $\mathrm{MSE} = 1.06 \times 10^{-3}$. Further, we test and validate the trained model using different core samples (i.e. #1 and #2B), which originate from the same geographical area and therefore can be assumed to have similar soil parameters. In Figure 2, we show that the predictions match the experimental data with reasonable accuracy.

Table 1: Comparison of MSE values between different deep learning architectures.

Method	Training	Extrapolated training	Test unseen	Parameters
TCN	$(7.9 \pm 5.4) \times 10^{-6}$	$(5.9 \pm 4.1) \times 10^{-3}$	$(3.0 \pm 1.2) \times 10^{-2}$	1 386
ConvLSTM	$(5.5 \pm 1.6) \times 10^{-6}$	$(4.9 \pm 5.7) \times 10^{-2}$	$(6.6 \pm 7.9) \times 10^{-2}$	1 496
DISTANA	$(1.9 \pm 1.1) \times 10^{-6}$	$(1.0 \pm 2.9) \times 10^{-2}$	$(1.6 \pm 4.0) \times 10^{-2}$	1 350
FINN	$(4.7 \pm 4.9) \times 10^{-5}$	$(1.1 \pm 1.2) \times 10^{-4}$	$(4.1 \pm 4.0) \times 10^{-5}$	465

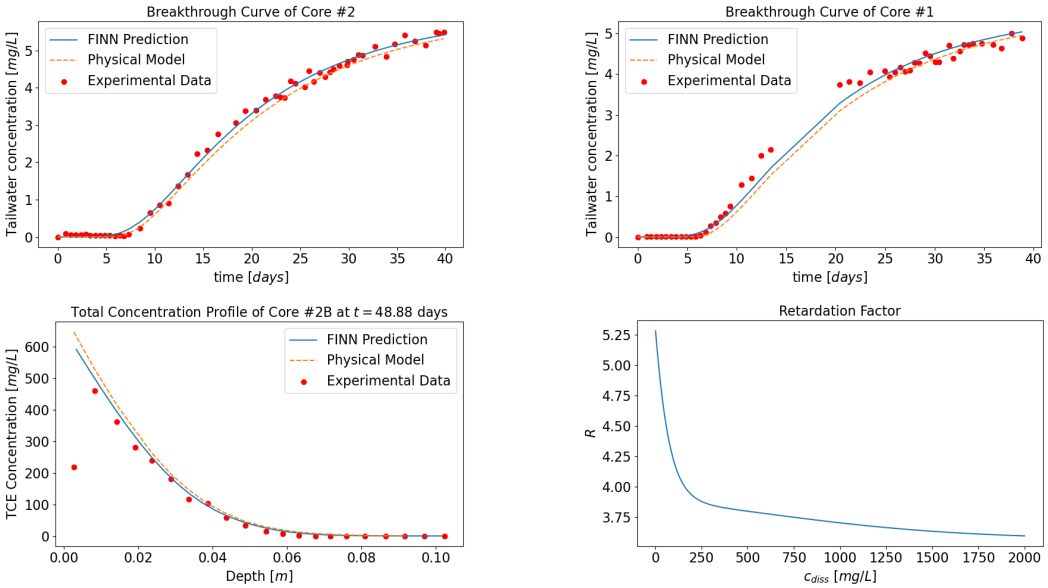

Figure 2: Breakthrough curve prediction of the FINN method (blue line) during training using data from core sample #2 (top left), during testing using data from core sample #1 (top right) and total concentration profile prediction using data from core sample #2B (bottom left). The predictions are compared with the experimental data (red circles) and the results obtained using the physical model (orange dashed line). The extracted retardation factor as a function of c is shown on the bottom right

We also compare the predictions with the output of the numerical model, with the retardation factor also calibrated using the data from the same core sample #2. The plots show that FINN's prediction accuracy is comparable to the numerical model, even beating it in some instances. For core sample #1, the MSE of FINN prediction is 1.37×10^{-3}, while the numerical model underestimates the breakthrough curve with $\mathrm{MSE} = 2.50 \times 10^{-3}$.

Specifically for core sample #2B, which is significantly longer than the other samples, to model the diffusion-sorption process, we can assume a zero-flux Neumann boundary condition at the bottom of the core. As a consequence, there are no breakthrough curve data available anymore. Instead, we compare the prediction against the total concentration profile data obtained from a destructive sampling (i.e. core slicing) at the end of the experiment. Here, FINN produces predictions with $\mathrm{MSE} = 1.16 \times 10^{-3}$, while the numerical model overestimates the total concentration profile with $\mathrm{MSE} = 2.73 \times 10^{-3}$. The results show that, even when applied to a different type of boundary condition, FINN's predictions remain accurate. Moreover, we can extract the retardation factor as a function of c using FINN, which is plotted in Figure 2, thus explaining a soil property.

5 Conclusion and Future Work

We have shown that including physical knowledge in the form of the Finite Volume structure produces excellent generalization capabilities and improves the explainability of the applied neural network structure. Our novel Finite Volume Neural Network (FINN) permits proper calculations for conservative fluxes and for different types of boundary condition. FINN outperforms other neural network methods for spatiotemporal modeling such as Temporal Convolutional Network, Convolutional LSTM, and DISTANA, especially when tested with different boundary conditions. Moreover, we show that FINN is suitable for experimental data processing, rendering it relevant as a data-driven modeling tool. Overall, recent development in fusing numerical methods with deep learning to aid physical processes simulation shows promising results to keep continuing the trend in this direction. It is very exciting to see how far we can push the boundaries between numerical methods and deep learning and to see the benefit when combining both approaches.

ACKNOWLEDGMENTS

For anonymity purpose, acknowledgments will be filled in after acceptance, because they contain personal information of the authors and their affiliations.

REFERENCES

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint*, 2016. URL https://arxiv.org/abs/1610.10099.

Matthias Karlbauer, Sebastian Otte, Hendrik P. A. Lensch, Thomas Scholten, Volker Wulfmeyer, and Martin V. Butz. A distributed neural network architecture for robust non-linear spatio-temporal prediction. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1912.11141.

A. Karpatne, G. Atluri, J.H. Faghmous, M. Steinbach, A. Banerjee, A.R. Ganguly, S. Shekhar, N.F. Samatova, and V. Kumar. Theory-guided Data Science: A New Paradigm for Scientific Discovery from Data. *arXiv preprint*, 2017. URL http://arxiv.org/abs/1612.08544v2.

A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling. *arXiv preprint*, 2018. URL http://arxiv.org/abs/1710.11431v2.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. *Object recognition with gradient-based learning*, pp. 319–345. Springer, 1999. ISBN 978-3-540-46805-9. doi: 10.1007/3-540-46805-6_19.

F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 1 edition, 2016. ISBN 9783319168746. doi: 10.1007/978-3-319-16874-6.

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

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint*, 2020. URL https://arxiv.org/abs/2001.04385.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. 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, 2019. doi: 10.1016/j.jcp.2018.10.045.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint*, 2015. URL https://arxiv.org/abs/1506.04214.

A.M. Tartakovsky, C.O. Marrero, P. Perdikaris, G.D. Tartakovsky, and D. Barajas-Solano. Learning Parameters and Constitutive Relationships with Physics Informed Deep Neural Networks. *arXiv preprint*, 2018. URL https://arxiv.org/abs/1808.03398.

Nanzhe Wang, Dongxiao Zhang, Haibin Chang, and Heng Li. Deep learning of subsurface flow via theory-guided neural network. *Journal of Hydrology*, 584:124700, 2020. doi: 10.1016/j.jhydrol.2020.124700.

A SOIL PARAMETERS AND SIMULATION DOMAINS FOR THE BENCHMARK TEST

In this appendix, we present the soil parameters and the simulation domains used to generate the benchmark dataset. Table 2 summarizes the parameters used to generate the training and test dataset. The retardation factor function is generated with the Freundlich sorption isotherm, written mathematically as

$$R = 1 + \frac{1-\phi}{\phi}\rho_s K_f n_f c^{n_f - 1}. \tag{7}$$

Table 2: Soil parameters and simulation domains for training and testing dataset generation.

Soil parameters		
Parameter	Unit	Value
D_e	m^2/day	5×10^{-4}
ϕ	-	0.29
ρ_s	kg/m^3	2880
K_f	(m^3/kg)^{n_f}	3.53×10^{-4}
n_f	-	0.874
Simulation domain		
Parameter	Unit	Value
L	m	1.0
Δx	m	0.04
t_{end}	days	10^4
Δt	days	2×10^3

Here, D_e is the effective diffusion coefficient, ϕ is the porosity, ρ_s is the bulk density, K_f is the Freundlich K coefficient, n_f is the Freundlich exponent, L is the length of the sample, Δx is the discrete control volume size, t_{end} is the simulation time, and Δt is the time step.

For the training dataset, the upper boundary condition ($x = 0$) is set to be a Dirichlet boundary condition, with the maximum solubility of TCE $c_s = 1.0$ kg/m^3. The testing dataset is generated with the same soil parameters and simulation domain, but with upper boundary condition $c_s = 0.7$ kg/m^3. The lower boundary condition ($x = L$) is set to be a Cauchy boundary condition: $c_L = \frac{D_e}{Q}\frac{\partial c}{\partial x}|_{x=L}$, where Q is the flow rate in the bottom reservoir. In the benchmark dataset, we assume that $Q = 1.0$.

B BENCHMARK TEST RESULTS

In this appendix, we present the results and compare different methods for the benchmark test results. For each method, we train 10 models with different initialization. The MSE values of the predictions are then calculated compared with the training dataset at $t = 0 - 2\,500$ days, the extrapolated training dataset at $t = 2\,500 - 10\,000$ days, and the whole unseen test dataset. We train the models with noisy data. The noise is normally distributed with standard deviation $\sigma = 1 \times 10^{-5}$, i.e. $\mathcal{N} \sim (0.0, 1 \times 10^{-5})$. Detailed information of the test MSE for every individual model is shown in Table 3 for seen data and in Table 4 for unseed data.

The prediction mean and confidence interval are plotted in Figure 3, Figure 4, Figure 5, and Figure 6. Even though the prediction mean of each method is not far from the synthetic data, clear instabilities and inconsistencies can be seen from the wide range of confidence intervals in the TCN, ConvLSTM, and DISTANA predictions. This instability is mainly caused by the improper handling of boundary conditions by these methods. FINN, on the other hand, produces very precise prediction along with high accuracy.

For the prediction of c_t using FINN, higher errors exist close to $x = 0$. This ensues from higher errors in the predicting the retardation factor by FINN at higher c values, and therefore c_t, where c_t

is proportial to R multiplied with c, also suffer from higher error when the value of c is high (i.e. close to $x = 0$).

Table 3: Test MSE on seen data (extrapolated training) coming from ten different training runs for each model

Run	TCN	ConvLSTM	DISTANA	FINN
1	5.2×10^{-3}	2.1×10^{-3}	6.3×10^{-5}	2.7×10^{-4}
2	4.9×10^{-3}	3.4×10^{-3}	3.6×10^{-4}	3.0×10^{-5}
3	4.1×10^{-3}	8.3×10^{-2}	9.7×10^{-2}	2.7×10^{-4}
4	4.1×10^{-3}	1.5×10^{-1}	4.0×10^{-4}	9.0×10^{-5}
5	8.1×10^{-4}	4.4×10^{-3}	4.2×10^{-5}	8.6×10^{-6}
6	1.2×10^{-2}	8.3×10^{-3}	4.5×10^{-4}	4.1×10^{-5}
7	1.5×10^{-2}	2.4×10^{-3}	8.2×10^{-5}	3.2×10^{-5}
8	4.5×10^{-3}	5.0×10^{-3}	9.5×10^{-4}	2.8×10^{-4}
9	2.6×10^{-3}	1.0×10^{-1}	5.2×10^{-5}	2.4×10^{-5}
10	5.6×10^{-3}	1.3×10^{-1}	1.8×10^{-4}	3.5×10^{-5}

Table 4: Test MSE on unseen data coming from ten different training runs for each model

Run	TCN	ConvLSTM	DISTANA	FINN
1	3.8×10^{-2}	1.1×10^{-2}	1.5×10^{-3}	9.7×10^{-5}
2	3.3×10^{-2}	1.1×10^{-3}	8.9×10^{-4}	1.5×10^{-5}
3	3.0×10^{-2}	1.0×10^{-1}	1.4×10^{-1}	9.5×10^{-5}
4	2.7×10^{-2}	1.2×10^{-1}	8.6×10^{-3}	3.4×10^{-5}
5	2.5×10^{-2}	7.0×10^{-3}	7.0×10^{-5}	4.9×10^{-6}
6	5.1×10^{-2}	5.6×10^{-4}	3.6×10^{-3}	1.9×10^{-5}
7	2.9×10^{-2}	2.6×10^{-2}	3.0×10^{-4}	1.5×10^{-5}
8	3.9×10^{-3}	3.1×10^{-4}	8.6×10^{-3}	1.0×10^{-4}
9	2.3×10^{-2}	1.9×10^{-1}	3.4×10^{-3}	1.2×10^{-5}
10	4.3×10^{-2}	2.2×10^{-1}	3.7×10^{-4}	1.6×10^{-5}

C MODEL DETAILS OF TCN, CONVLSTM AND DISTANA

In this appendix, we provide additional information about the TCN, ConvLSTM and DISTANA models: bias neurons were used in all layers of all architectures and ADAM was used for optimization with a learning rate of $\eta = 1 \times 10^{-3}$. As fair comparison, FINN results for the benchmark test case are obtained also using the ADAM optimizer with the same learning rate. While TCN, ConvLSTM and DISTANA are always provided with the real data in the first ten timesteps (i.e. teacher forcing), FINN only receives an initial condition in the first timestep along with the information about the boundary in all timesteps. For better comparison, we also provide boundary condition information for TCN, ConvLSTM, and DISTANA. Note that in this experiment, TCN, ConvLSTM and DISTANA are provided with more information than FINN, which nevertheless outperforms the other models.

TCN Two input channels are followed by two hidden layers with four and eight channels, respectively, which are processed by two output channels. A convolution kernel size of $k = 3$ was chosen and the standard dilation rate of TCN was applied ($d = l^2$, where l is the index of the layer), leading to a time horizon of 28 time steps. Code was taken and modified from [1].

ConvLSTM Two input feature maps, followed by ten channels in one hidden layer and two output channels, were used. The convolution kernel size was set to $k = 3$ and zero-padding was applied to

[1] https://github.com/locuslab/TCN

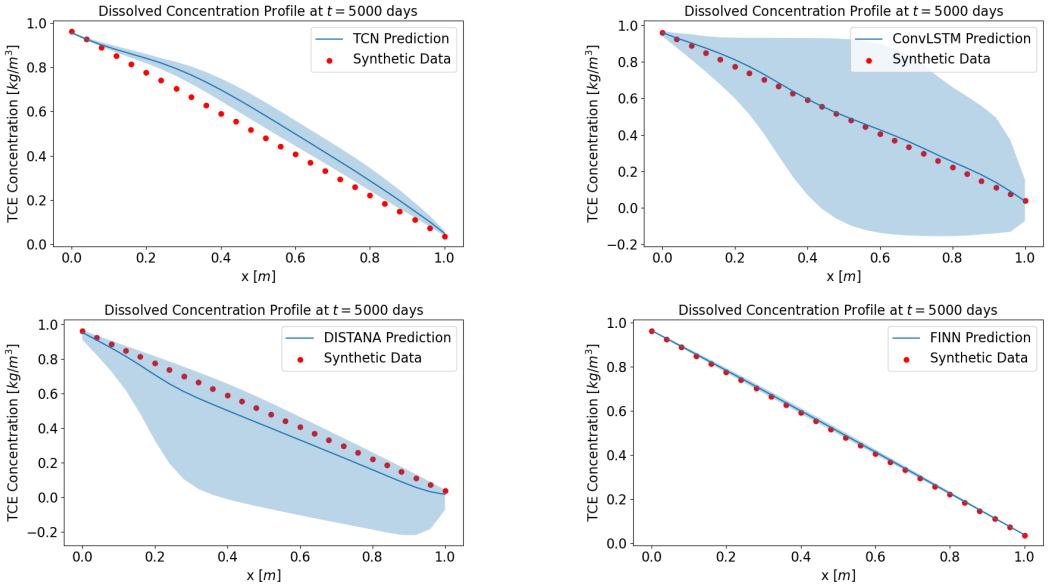

Figure 3: Dissolved concentration profile prediction mean (with confidence interval) at $t = 5\,000$ days compared with the extrapolated training dataset obtained using TCN (top left), ConvLSTM (top right), DISTANA (bottom left), and FINN (bottom right).

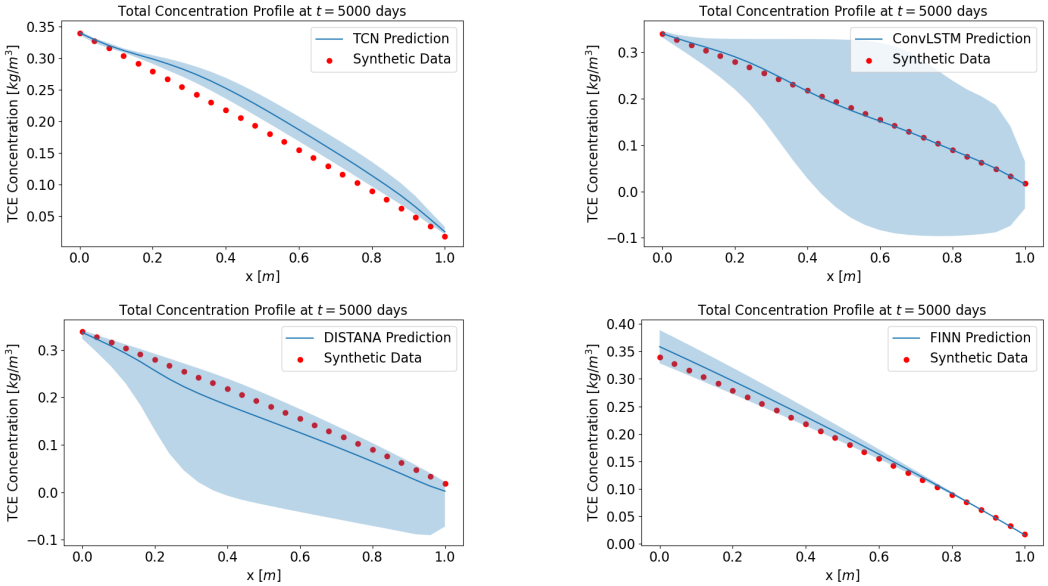

Figure 4: Total concentration profile prediction mean (with confidence interval) at $t = 5\,000$ days compared with the extrapolated training dataset obtained using TCN (top left), ConvLSTM (top right), DISTANA (bottom left), and FINN (bottom right).

preserve data dimensions. PyTorch code was taken from [2] and adapted to be applicable to spatially one-dimensional data.

[2] https://github.com/ndrplz/ConvLSTM_pytorch

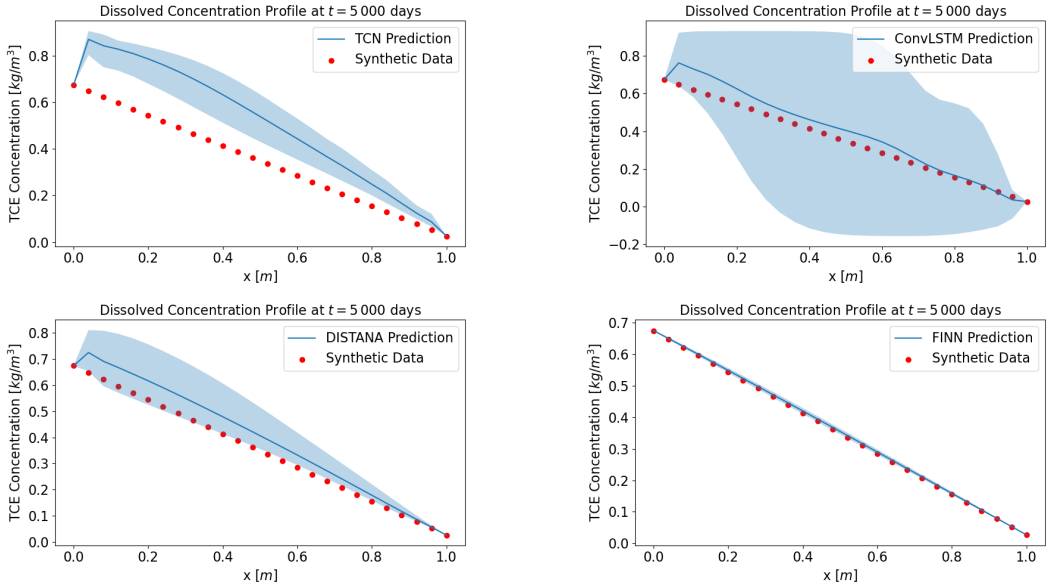

Figure 5: Dissolved concentration profile prediction mean (with confidence interval) at $t = 5\,000$ days compared with the test dataset obtained using TCN (top left), ConvLSTM (top right), DISTANA (bottom left), and FINN (bottom right).

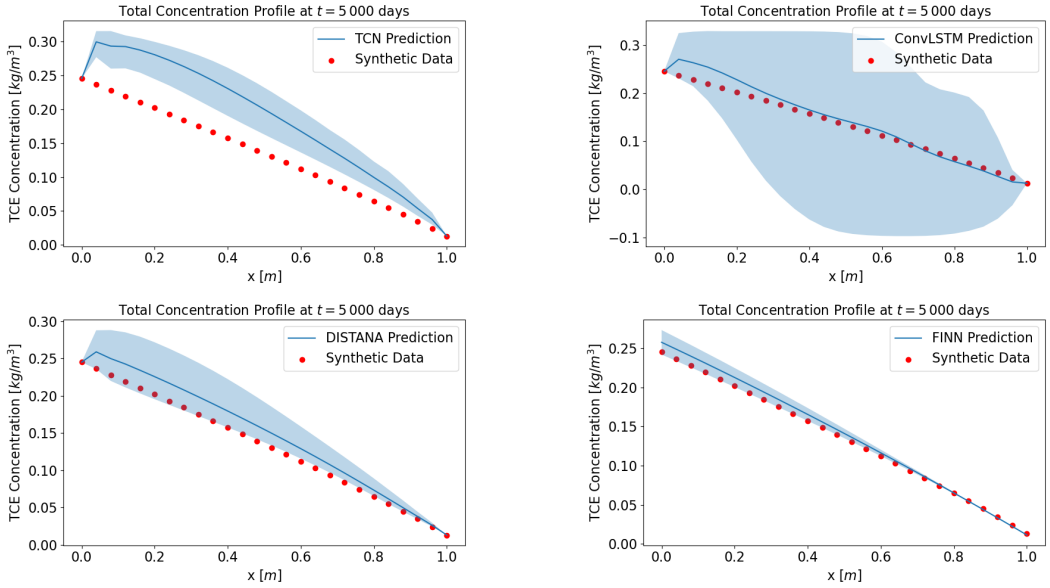

Figure 6: Total concentration profile prediction mean (with confidence interval) at $t = 5\,000$ days compared with the test dataset obtained using TCN (top left), ConvLSTM (top right), DISTANA (bottom left), and FINN (bottom right).

DISTANA Two input channels map to four preprocessing convolution channels, which feed forward into a ConvLSTM layer with eight channels which are processed by two postprocessing convolution channels. The lateral information processing convolution layer was set to two channels.

D SOIL PARAMETERS AND SIMULATION DOMAINS FOR THE EXPERIMENT

In this appendix, we present the soil parameters and the simulation domains of the core samples used in the experiment. Table 5 summarizes the parameters of core sample #1, #2, and #2B.

Table 5: Soil and experimental parameters of core samples #1, #2, and #2B.

Soil parameters				
Parameter	Unit	Core #1	Core #2	Core #2B
D_e	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_{end}	days	38.81	39.82	48.88
Q	m^3/day	1.01×10^{-4}	1.04×10^{-4}	N/A
c_s	kg/m^3	1.4	1.6	1.4

For all experiments, the core samples are subjected to a constant TCE concentration at the top c_s, which amounts to a Dirichlet boundary condition. Notice that, for core sample #2, we set c_s to be slightly higher to compensate for the fact that there might be fractures at the top of core sample #2, so that the TCE can break through the core sample faster.

For core samples #1 and #2, Q is the flow rate of clean water at the bottom reservoir that determines the Cauchy boundary condition at the bottom of the core samples. For core sample #2B, note that the sample length is significantly longer than the other samples. Therefore, for this particular sample, given t_{end} to be approximately in the same order with the other samples, we assume the bottom boundary condition to be a no-flow Neumann boundary condition.