Fluid Flow-based Deep Learning (FFDL) for geologic CO_2 Storage

Zhen Qin[1], Yingxiang Liu[2], Fangning Zheng[1], Behnam Jafarpour[1*]

[1]Mork Family Department of Chemical Engineering and Materials Science, University of Southern California, 925 Bloom Walk, Los Angeles, 90089, CA, USA
[2]Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, 3740 McClintock Avenue, Los Angeles, 90089, CA, USA

Key Points:

- **Novel and generic approach**: Physics-encoded model to predict the evolution of pressure and saturation fields during geologic CO_2 storage under dynamic injection rates and permeability variations
- **Physics-encoded architecture**: A novel architecture featuring a fluid flow physics-based encoder and a residual-based processor.
- **Physics-based operator**: A novel nonlinear activation function, the physics-based operator, is applied to capturing physical relationships.
- **Generalization and extrapolation**: Robust generalizable approach with accurate predictions over extended time frames and strong extrapolation capabilities.

*Corresponding author

Corresponding author: Behnam Jafarpour, `behnam.jafarpour@usc.edu`

Abstract

Carbon capture and storage (CCS) is one of the few strategies for reducing CO_2 emissions by injecting it into deep geologic formations. The injection of CO_2 into heterogeneous rock formations triggers a set of complex coupled flow and transport processes that are not trivial to describe and predict. Advanced numerical simulation is often used as a standard tool to predict the spatial-temporal evolution of the CO_2 plume and the induced pressure changes. However, numerical simulation is computationally demanding and can limit the use of standard field management workflows, such as risk assessment workflows, and hinder real-time analysis and decision-making for risk mitigation. Standard deep learning models provide powerful alternative prediction tools. However, they also have important limitations, including lack of interpretability, extensive data needs, and physical inconsistency.

To overcome these limitations, a Fluid Flow-based Deep Learning (FFDL) architecture is presented for spatial-temporal prediction of the injected CO_2 in storage formations. The architecture of FFDL consists of a physics-based encoder to construct physically meaningful latent variables, and a residual-based processor to predict the evolution of the state variables. The FFDL model uses physical operators that serve as nonlinear activation functions and impose hard constraints to respect the general structure of the fluid flow equations. A comprehensive investigation of FFDL, based on a field-scale saline aquifer, is used to demonstrate its superior performance compared to standard deep learning models. The results show that FFDL exhibits strong generalization capability and provides more reliable and physically consistent predictions of CO_2 plume migration. The flexibility of FFDL makes it suitable for various applications, including decision-making, optimization, and inverse modeling.

1 Introduction

Carbon capture and storage (CCS) is an important component in reducing the CO_2 emissions by capturing from point sources and injecting it into deep geologic formations. Although geologic CO_2 storage has significant potential, it is still in the early stages. Concerns about CO_2 migration or leakage into shallow aquifers call for robust monitoring and risk management technologies (Zheng et al., 2022, 2021; Celia et al., 2015). While current monitoring methods can track the movement of the injected CO_2 plume, accurately quantifying its volume and migration path remains a challenge for monitoring and verification purposes (Bui et al., 2018). Moreover, predicting the dynamics of pressure buildup and CO_2 migration is essential for guiding decision-making (Zheng et al., 2021) and for assessing real-time risks throughout the life cycle of a project (Ajayi et al., 2019).

Reliable prediction of the CO_2 plume migration often requires spatial and temporal analysis, potentially involving detailed simulation models. The injection of CO_2 into subsurface formations triggers a complex multi-component, multiphase flow system. The complex relationships involving miscibility, capillary pressure, and relative permeability, as well as coupled physics lead to nonlinear coupled systems of partial differential equations (PDEs) that are not trivial to solve (Bandilla et al., 2015). Furthermore, field-scale geologic CO_2 storage (GCS) projects span extensive spatial and temporal scales, including both injection and post-injection periods (Ajayi et al., 2019; X. Jiang, 2011). Therefore, numerical simulation for GCS at the field scale can become computationally prohibitive, particularly for complex tasks such as optimization and uncertainty quantification. Another significant challenge is estimating the time-varying storage capacity of the geologic formations, which is influenced by geologic conditions, injectivity and field development plans (Gorecki et al., 2015). Accurate quantification of uncertainty and potential risks typically involves multiple simulation runs, which can impede the implementation of real-time analysis and risk assessments in GCS projects. Consequently, there is a growing need for innovative approaches that can provide accurate and efficient real-

time monitoring and forecasting of CO_2 plume migration and pressure buildup during GCS operations.

In recent years, deep learning (DL)-based approaches have emerged as promising alternatives to traditional numerical simulations for predicting the spatial-temporal evolution of fluid dynamics in the subsurface. Specifically, convolutional neural networks (CNNs) that have demonstrated a strong capability for processing image data have found widespread application in predicting the spatial and temporal evolution of subsurface flow systems. Zhu and Zabaras (2018) proposed a fully convolutional encoder-decoder architecture to approximate the mapping from permeability to pressure and velocity maps for a 2-dimensional steady-state Darcy flow problem. Mo, Zhu, et al. (2019) extended the work in Zhu and Zabaras (2018) to predict the responses from a dynamic multiphase flow problem at different time steps. Y. Wang and Lin (2020) designed a custom architecture tailored for single- and two-phase flow systems, incorporating sparsely connected layers to account for the inherent sparse input-output interaction.

Among the family of CNNs, U-Net (Ronneberger et al., 2015) was originally designed for biomedical image segmentation and has emerged as a mainstream model for prediction tasks in the subsurface domain (Wen, Tang, & Benson, 2021; Z. Jiang et al., 2021; H. Tang et al., 2021; Yan, Harp, Chen, & Pawar, 2022). U-Net excels in capturing complex patterns by retaining multi-scale details of input images through skip connections. This feature of U-Net facilitates the integration of high-resolution details, reduces the search space of model parameters, and mitigates the gradient vanishing issue often encountered in deep architectures. Yan, Harp, Chen, and Pawar (2022) proposed a physics-constrained smoother to enhance the pressure prediction generated by U-Net. Their approach incorporates the time step as an additional input, allowing the U-Net model to predict pressure or saturation at different time steps. This method does not explicitly capture the dynamics of the flow system. Instead, it approximates a static mapping conditioned on the time step. Alternatively, spatial-temporal predictions can also be achieved using auto-regressive architecture (Mo, Zabaras, et al., 2019; Z. Jiang et al., 2021), recurrent architecture (M. Tang et al., 2020) or by jointly predicting all time frames (Wen, Hay, & Benson, 2021; Wen et al., 2022, 2023). M. Tang et al. (2020) introduced a combined model called Recurrent R-U-Net that integrates Residual U-Net (R-U-Net) (Wen, Tang, & Benson, 2021) with convolutional long short-term memory (ConvLSTM) network (Shi et al., 2015) to capture the evolution of saturation and pressure in the 2D two-phase waterflooding problem. This combined architecture was later extended to address 3D flow problems in waterflooding (M. Tang et al., 2021) and CO_2 sequestration (M. Tang et al., 2022). Different from the recurrent architecture, Wen et al. (2023) proposed a nested framework of DL models for 4D spatial-temporal prediction of pressure and saturation in a joint way by utilizing Fourier Neural Operator (FNO) (Li et al., 2020). Their methodology involves dividing a large reservoir into multiple scales (levels), with each level employing an FNO model to predict either pressure or saturation. By combining the FNOs of each level sequentially, the states of the entire reservoir can be predicted. Unlike the recurrent nature of ConvLSTM, the FNO approach predicts all time steps simultaneously, enabling parallelized prediction across the time domain. However, the FNO's 4D architecture results in a significantly larger number of trainable parameters, with approximately 80 to 150 million parameters for each level, making it computationally demanding and potentially resource-intensive when extending it to more complex problems.

Despite the demonstrated effectiveness of DL models, the approaches mentioned above have limitations and are tailored to specific scenarios. Notably, in GCS projects that can span decades for injection and potentially much longer for the post-injection period, the predictive model's capability and flexibility to forecast over arbitrary time frames are crucial. However, existing DL models are constrained to predicting within fixed periods and encounter challenges in long-term predictions (which require extrapolating

power). Deep learning models have limited extrapolation power and do not provide reliable predictions beyond the data distribution defined by the training set (Willard et al., 2022), leading to the out-of-distribution (OOD) generalization problem. As reflected in Mo, Zhu, et al. (2019), the prediction over the OOD time steps becomes an extrapolation task that can challenge deep learning models, as the saturation predictions during the extrapolation may become physically inconsistent. This is particularly important for GCS projects that require long-term prediction horizons. Another important feature is the flexibility of the model in terms of providing predictions over different development scenarios. For instance, the models should be able to capture the response of the storage formations to time-varying controls, which is important in optimizing the performance of the field by dynamically managing the injection strategies based on the in-situ reservoir conditions and monitoring data (Pawar et al., 2015).

To enhance predictive capabilities and overcome limitations, an emerging and active research field focuses on integrating physical principles and domain knowledge into neural networks (NNs) (Willard et al., 2022; Faroughi et al., 2023). A flexible approach is to incorporate governing equations into the loss function to constrain the neural networks during training, known as Physics-informed Neural Networks (PINNs) (Raissi et al., 2019). The PINN framework offers a flexible implementation across various physical systems and has found successful applications in the subsurface domains (N. Wang et al., 2020; Shokouhi et al., 2021; Yan, Harp, Chen, Hoteit, & Pawar, 2022). However, physics-informed approaches often struggle with adhering to boundary conditions, as they implement physical constraints in a "soft" manner. Furthermore, the inherent complexity of multiphase flow in heterogeneous porous media presents challenges in implementing the closed-form residuals of nonlinear PDEs as loss functions (Yan, Harp, Chen, Hoteit, & Pawar, 2022). The training of deep learning models using physics-informed loss functions without labeled data can be affected by the complexity and nonlinearity of the integrated physics. The presence of complex governing equations can lead to highly nonlinear and non-convex loss functions, complicating the training process (Fuks & Tchelepi, 2020). These issues may challenge the use of data-free approaches in more complex problems, including high-dimensional and highly nonlinear systems (Cuomo et al., 2022; Muther et al., 2023).

Another fit-for-purpose alternative involves hard-encoding the underlying physics into the architecture of neural networks, endowing the resulting models with an inductive bias tailored to specific physical problems (Faroughi et al., 2023; Karniadakis et al., 2021). In contrast to physics-informed approaches, physics-encoded architectures impose hard constraints. By capturing the underlying physical dependencies among variables, these architectures demonstrate their connections to Ordinary Differential Equations (ODEs) (E, 2017; E et al., 2017; Lu et al., 2018) and PDEs (Long et al., 2018, 2019; Ruthotto & Haber, 2020; Rao et al., 2021). Long et al. (2018, 2019) proposed PDE-Net to learn PDEs from data. In their work, the CNN is combined with Residual Network (ResNet) (He et al., 2016) to approximate the evolution of PDE with the forward Euler as the temporal discretization. Rao et al. (2021) introduced the product block to emulate the governing terms in PDEs. Their study utilizes convolutional layers to learn spatial dependencies and employs a recurrent form of ResNet for approximating temporal evolution. The resulting DL model has shown good performance in extrapolation tasks. More recently, Dulny et al. (2022) proposed NeuralPDE to combine Neural Ordinary Differential Equations (NeuralODEs) (Chen et al., 2018) with the Method of Lines (MOL) using CNNs to approximate the spatial component in PDEs. They state that CNNs can approximate the MOL, a numerical method of solving time-dependent PDEs by representing them as systems of ODEs through spatial discretization. Nonetheless, these studies tend to simplify the governing equations and approximate the dynamics in an explicit form.

In this study, we propose a novel physics-encoded DL model, named Fluid Flow-based Deep Learning (FFDL), for predicting the spatial-temporal evolution of the pressure and saturation in geologic CO_2 storage. To bridge the gaps in the existing models, the FFDL model is designed to handle time-varying well controls and to provide long-term predictions. The architecture of FFDL primarily comprises a physics-based encoder for constructing physically meaningful latent variables, a residual-based processor for the recurrent prediction of latent variables, a control encoder for constructing a latent representation of sink or source term, and a decoder for approximating the mapping from latent variables to outputs, namely pressure and saturation. The physics-based encoder, along with the control encoder, can construct different governing terms in the PDEs for multiphase flow, including accumulation, advection, and sink/source terms. Similar to the work in (Long et al., 2019; Rao et al., 2021; Dulny et al., 2022), the convolutional layers are employed to capture the spatial dependencies. However, our model extends beyond these by 1) introducing physics-based operators as the activation functions, 2) constructing latent representations of the governing terms to better approximate the dynamics, and 3) updating the latent governing terms in a coupled and implicit form. We also present a modified Recurrent R-U-Net, based on the work in M. Tang et al. (2022), as a baseline model due to its similar architecture and capability for extension to time-varying control and arbitrary time steps. The predictive performance of FFDL is investigated using a field-scale model of GCS in a saline aquifer. Our results show that FFDL outperforms the Recurrent R-U-Net on test sets featuring unseen permeability, well controls, and time frames.

The remaining sections of this paper are organized as follows. Section 2 presents the problem statement for the spatial-temporal prediction task and introduces the architectures of the proposed physics-based encoder and residual-based processor. In Section 3, we provide a brief overview of the experimental setups and describe the modifications made to Recurrent R-U-Net. The experimental results and discussions are presented in Section 4. Finally, Section 5 offers conclusions on this work and highlights the advantages of the proposed model.

2 Methodology

2.1 Problem Statement

This study addresses a spatial-temporal prediction task governed by a set of coupled PDEs within a 3D reservoir composed of grid blocks $\mathcal{D} \subset \mathbb{R}^3$. The prediction task can be formulated as $\mathbf{X} = \mathcal{F}(\boldsymbol{x}^t, \boldsymbol{m}, \mathbf{U})$, given the inputs $\boldsymbol{x}^t$ as the dynamic states at time step t, $\boldsymbol{m}$ as parameters, and $\mathbf{U}$ as future controls. The output sequence $\mathbf{X} = \{\boldsymbol{x}^{t+1}, \boldsymbol{x}^{t+2}, \ldots\}$ represents the dynamic variables over the subsequent time steps, and the operator $\mathcal{F}$ denotes the mapping from inputs to outputs. For each coordinate $\boldsymbol{\omega} \in \mathcal{D}$, the dynamic variable $\boldsymbol{x}^t(\boldsymbol{\omega})$ at time step t consists of pressure $\boldsymbol{p}^t(\boldsymbol{\omega}) \in \mathbb{R}$ and saturation of non-wetting phase $\boldsymbol{S}_n^t(\boldsymbol{\omega}) \in \mathbb{R}$. The input parameters $\boldsymbol{m}(\boldsymbol{\omega}) \in \mathbb{R}^{d_m}$ characterize the model parameters, such as permeability, porosity, grid volume, and elevation, at point $\boldsymbol{\omega}$. The control sequence $\mathbf{U} = \{\boldsymbol{u}^{t+1}, \boldsymbol{u}^{t+2}, \ldots\}$ denotes well controls over future time steps. At any given step t, the control variable $\boldsymbol{u}^t$ can be the bottom-hole pressure (BHP) or flow rate. The value of each grid cell $\boldsymbol{u}^t(\boldsymbol{\omega}) \in \mathbb{R}^{d_u}$ corresponds to the control value (e.g., injection rate) at a well location ω and is zero if the cell does not contain a well. Superscripts d_x, d_m, and d_u indicate the dimensions of dynamic states, input parameters, and control variables at a point $\boldsymbol{\omega}$, respectively. The goal of this work is to approximate the operator $\mathcal{F}$ with the proposed deep learning model $\mathcal{F}_\theta$, where θ represents the trainable parameters. In this paper, variables denoted in bold represent high-dimensional tensors, while those not in bold refer to scalars. For brevity, we primarily focus on the dimensionality of high-dimensional tensors at individual spatial points $\boldsymbol{\omega}$, rather than the entire spatial domain.

In this approach, we avoid the direct learning of the pressure and saturation dynamics. Instead, we convert these variables into physically meaningful latent variables and learn the dynamics in latent space. The latent representation $\boldsymbol{z}^t = \{\boldsymbol{z}^t_{acc}, \boldsymbol{z}^t_{adv}, \boldsymbol{z}^t_{src}\}$ consists of three components representing the accumulation, advection, and sink/source terms of the governing PDEs in the latent space, respectively. As depicted in Figure 1, the proposed deep learning model mainly consists of the encoder, processor, control encoder, and decoder modules. Initially, the encoder predicts the first two latent variables over the next time step, $\boldsymbol{z}^{t+1}_{acc}$ and $\boldsymbol{z}^{t+1}_{adv}$, using $\boldsymbol{x}^t$ and $\boldsymbol{m}$ as inputs. Concurrently, the control encoder produces the latent variables $\boldsymbol{z}^{t+1}_{src}$ given the control variables $\boldsymbol{u}^{t+1}$. The processor receives three latent variables as input and generates the updated latent variables $\boldsymbol{z}^{t+1}_{acc}$ and $\boldsymbol{z}^{t+1}_{adv}$ as outputs. These outputs are then sent to 1) the decoder for the prediction of the dynamic variables $\boldsymbol{x}^{t+1}$, and to 2) the processor itself for the estimation of the new latent variables $\boldsymbol{z}^{t+2}_{acc}$ and $\boldsymbol{z}^{t+2}_{adv}$. The decoder takes the updated latent variables $\boldsymbol{z}^{t+1}_{acc}$ and $\boldsymbol{z}^{t+1}_{adv}$, as well as the static variable $\boldsymbol{m}$, as inputs. By including $\boldsymbol{m}$, the decoder is designed to decouple the effects of parameters from the latent variables, functioning as the inverse of the encoder. In a recurrent manner, the processor utilizes the previous latent variables as initial estimations and updates them for the next time step by integrating $\boldsymbol{z}^{t+1}_{src}$ as external inputs. To articulate the model's function, the process $\mathbf{X} = \mathcal{F}(\boldsymbol{x}^t, \boldsymbol{m}, \mathbf{U})$ can be modularized as follows:

$$\begin{aligned}
\{\boldsymbol{z}^{t+1}_{acc}, \boldsymbol{z}^{t+1}_{adv}\} &= \mathcal{F}_{\theta_{\text{InputToLatent}}}(\boldsymbol{x}^t, \boldsymbol{m}), \\
\{\boldsymbol{z}^{t+n}_{src}\} &= \{\mathcal{F}_{\theta_{\text{ControlToLatent}}}(\boldsymbol{u}^{t+n})\}, n = 1, 2, \ldots, \\
\{\boldsymbol{z}^{t+n}_{acc}, \boldsymbol{z}^{t+n}_{adv}\} &= \{\mathcal{F}_{\theta_{\text{LatentToLatent}}}(\boldsymbol{z}^{t+n}_{acc}, \boldsymbol{z}^{t+n}_{adv}, \boldsymbol{z}^{t+n}_{src})\}, n = 1, 2, \ldots, \\
\{\boldsymbol{x}^{t+n}\} &= \{\mathcal{F}_{\theta_{\text{LatentToOutput}}}(\boldsymbol{z}^{t+n}_{acc}, \boldsymbol{z}^{t+n}_{adv}, \boldsymbol{m})\}, n = 1, 2, \ldots,
\end{aligned} \tag{1}$$

where, the operators $\mathcal{F}_{\theta_{\text{InputToLatent}}}$, $\mathcal{F}_{\theta_{\text{ControlToLatent}}}$, $\mathcal{F}_{\theta_{\text{LatentToLatent}}}$, and $\mathcal{F}_{\theta_{\text{LatentToOutput}}}$ represent the encoder, control encoder, processor, and decoder modules, respectively.

2.2 Physics-Encoded Deep Learning Model

One of the main contributions of this work is the design of the encoder and processor modules, as depicted in Figure 2. In contrast, the control encoder and decoder are implemented using standard convolutional layers with simpler configurations. This subsection delineates 1) the physical operators used in the encoder, and 2) the detailed architectures of the encoder and processor. A comprehensive description of the governing equations for geologic CO_2 storage is provided in Appendix A. The explanation of the control encoder and decoder is provided in Appendix B.

In our model, the encoder is designed to capture the relationship between the latent representations of the governing terms (i.e., accumulation and advection) and various input variables (e.g., pressure, saturation, permeability, etc.) using physical operators. Recent works designed neural operators to either 1) construct the latent variables in the spectral domain (Li et al., 2020; Wen et al., 2022; Xiong et al., 2023) or 2) introduce high-frequency features to the DL model (H. Wu et al., 2023). Although these advanced architectures effectively address the issue of spectral bias (Tancik et al., 2020; Rahaman et al., 2019), the constructed latent variables lack interpretability, resulting in a black-box architecture. In contrast, we adopt the concept of physical operators to map the input variables to physically meaningful latent variables that represent accumulation and advection terms. This design is inspired by the Operator-Based Linearization (OBL) approach (Voskov, 2017), which represents the governing terms as combinations of operators and linearizes them within the parameter space (pressure, saturation, and component) to facilitate the application of numerical solvers.

Two primary motivations underpin the transition from input variables to physics-based latent representations. First, the governing equations of the system involve implicit and nonlinear dependencies on pressure and saturation, challenging the derivation of explicit update forms. Previous studies have utilized ResNet (or skip connections) for

direct updates of variables in explicit form. These methods often rely on simplifications of PDEs (J. Nagoor Kani & Elsheikh, 2019; Y. Wang & Lin, 2020) or are tailored to specific problem characteristics (Rao et al., 2021). In contrast to direct updates of dynamic variables, our method focuses on the dynamics within the latent space by mapping them onto physics-based latent variables. This design circumvents the assumptions required by previous works, offering a more thorough treatment of the underlying physics. Second, predicting the dynamics within the latent space offers our model flexibility and generalizability. The physics-based encoder can flexibly adapt to various physical terms, such as capillary and gravity terms, through the design of physics-based operators. Moreover, the physical operators enable our model to effectively generalize across diverse input variables and learn the physical relationship between diverse inputs and outputs. Consequently, this design potentially extends the applicability of our model to various tasks and scenarios.

2.2.1 Physical Operators

For geologic CO_2 sequestration sites, such as saline aquifers, the system consists of two primary phases: a water-rich phase and a CO_2-rich phase. In this work, we simplify the CO_2-brine system to an immiscible two-fluid-phase system with no internal component gradient. As a result, the mass balance equation can be written in terms of phase-based balance equations:

$$\frac{\partial}{\partial t}\left(\phi S_\xi \rho_\xi\right) + \nabla \cdot \left(\rho_\xi \mathbf{v}_\xi\right) = \rho_\xi \tilde{q}_\xi, \quad \xi \in \{w, n\}, \tag{2}$$

where ϕ is the porosity; S_ξ and ρ_ξ are the saturation and density of phase ξ, respectively; $\mathbf{v}_\xi$ is the volumetric flux vector for phase ξ; $\tilde{q}_\xi$ is external sources or sinks of volumetric rate per unit volume of phase ξ. The phase ξ is n for the non-wetting phase (supercritical CO_2) and w for the wetting phase (brine).

We rewrite the Eq. 2 as a combination of operators in an algebraic form for the whole reservoir in three-dimensional space $\mathcal{D}$:

$$\boldsymbol{r}_\xi(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{u}) = \left(\boldsymbol{z}_{acc,\xi}(\boldsymbol{x}) - \boldsymbol{z}_{acc,\xi}(\boldsymbol{x}^{t-1})\right) - \boldsymbol{z}_{adv,\xi}(\boldsymbol{x}, \boldsymbol{m}) + \boldsymbol{z}_{src,\xi}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{0}, \tag{3}$$

where $\boldsymbol{r}_\xi$ is the residual of the governing equation for phase ξ over the entire reservoir; $\boldsymbol{z}_{acc,\xi}$, $\boldsymbol{z}_{adv,\xi}$, and $\boldsymbol{z}_{src,\xi}$ are the accumulation, advection, and sink/source terms for phase ξ, respectively.

The governing terms are calculated through the physical operators. Detailed derivation of the governing equations and operators are provided in Appendix A. Here, we define the operators as follows

$$\boldsymbol{z}_{acc,\xi}(\boldsymbol{x}) = \boldsymbol{c}_\phi \circ \boldsymbol{S}_\xi \circ \boldsymbol{\rho}_\xi = \left(1 + \boldsymbol{c}_r \circ (\boldsymbol{p} - \boldsymbol{p}_{ref})\right) \circ \boldsymbol{S}_\xi \circ \boldsymbol{\rho}_\xi, \tag{4}$$

$$\boldsymbol{z}_{adv,\xi}(\boldsymbol{x}, \boldsymbol{m}) = \boldsymbol{a}(\boldsymbol{m}) \circ \sum_l \boldsymbol{\beta}_{\xi,l}(\boldsymbol{x}) \circ \boldsymbol{b}_l(\boldsymbol{x}, \boldsymbol{m}), \tag{5}$$

$$\boldsymbol{z}_{src,\xi}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{a}(\boldsymbol{m}) \circ \boldsymbol{\rho}_\xi \circ \boldsymbol{q}_\xi = \boldsymbol{a}(\boldsymbol{m}) \circ \boldsymbol{\rho}_\xi \circ \boldsymbol{V} \circ \tilde{\boldsymbol{q}}_\xi, \tag{6}$$

$$\boldsymbol{a}(\boldsymbol{m}) = \Delta t \; \boldsymbol{I}_{PV}, \tag{7}$$

$$\boldsymbol{\beta}_{\xi,l}(\boldsymbol{x}) = \boldsymbol{\lambda}_\xi \circ \boldsymbol{\rho}_\xi, \tag{8}$$

$$\boldsymbol{b}_l(\boldsymbol{x}, \boldsymbol{m}) = \boldsymbol{T}^l_m \circ \Delta \boldsymbol{\psi}^l_\xi = \boldsymbol{T}^l_m \circ \left(\boldsymbol{\psi}^v_\xi - \boldsymbol{\psi}^u_\xi\right), \tag{9}$$

where the symbol $\circ$ represents the Hadamard product (or element-wise production); $\boldsymbol{c}_\phi$ is defined as an update multiplier for the initial porosity; $\boldsymbol{c}_r$, $\boldsymbol{p}_{ref}$, and $\boldsymbol{V}$ are rock compressibility, reference pressure, and grid volume, respectively; $\boldsymbol{q}_\xi$ is the volumetric flow rate for phase ξ; Δt is a scalar and represents the time interval; $\boldsymbol{I}_{PV}$ is the inverse of the

initial pore volume of a grid cell $\phi_0 V$; $\boldsymbol{\lambda}_\xi$ is the mobility of phase ξ; $\boldsymbol{T}_m^l$ is the geometric part of the transmissibility of interface l between two grids u and v; $\Delta\boldsymbol{\psi}_\xi^l$ is the phase potential difference between the two grid cells u and v, of which the phase potentials are $\boldsymbol{\psi}_\xi^u$ and $\boldsymbol{\psi}_\xi^v$, respectively. For the design of the encoder, the phase potential is simplified by neglecting the capillary pressure and represented as $\boldsymbol{\psi}_\xi = \boldsymbol{p} + \boldsymbol{\rho}_\xi \circ \boldsymbol{D}$. The variable $\boldsymbol{D} = g\boldsymbol{d}$ refers to the gravity term defined as the product of the gravitational acceleration (g) and the elevation (or depth) of grid cells ($\boldsymbol{d}$). The parameter $\boldsymbol{m}$ is defined as a set of three components $\{\boldsymbol{T}_m, \boldsymbol{I}_{PV}, \boldsymbol{D}\}$.

2.2.2 Physics-based Encoder

The encoder consists of three stages in a sequence (See Figure 2 (a)). The first stage of the encoder takes the dynamic variables $\boldsymbol{x}$ and parameter $\boldsymbol{m}$ as inputs. The output from the first stage is a set of physical features $\boldsymbol{f}$ used in the operators and defined as follows:

$$\boldsymbol{f} = \{\boldsymbol{c}_\phi, \boldsymbol{\rho}_\xi, \boldsymbol{\lambda}, \Delta\boldsymbol{p}, \Delta\left(\boldsymbol{\rho}_\xi \circ \boldsymbol{D}\right)\}. \tag{10}$$

The dependencies of $\boldsymbol{c}_\phi$ and $\boldsymbol{\rho}_\xi$ on pressure and the dependency of $\boldsymbol{\lambda}_\xi$ on pressure and saturation are parameterized using fully-connected (FC) NN, which is named state-related operator (Figure 3 (a)). In this work, we parameterize the state-related operator using a two-layer fully connected NN with the hidden unit d_o. The Gaussian Error Linear Unit (GELU) (Hendrycks & Gimpel, 2023) serves as the nonlinear activation function after each layer. Features $\Delta\boldsymbol{p}$ and $\Delta\left(\boldsymbol{\rho}_\xi \circ \boldsymbol{D}\right)$ are the differential pressure and gravity terms used in the potential difference $\Delta\boldsymbol{\psi}$, which are calculated through the spatial-related operator (Figure 3 (b)). In a structured grid system, $\boldsymbol{p}$ and $\boldsymbol{D}$ are first padded with reflection padding around the edges of tensors, which serves as a closed boundary. Given the padded terms, $\Delta\boldsymbol{p}$ and $\Delta\left(\boldsymbol{\rho}_\xi \circ \boldsymbol{D}\right)$ are computed along the x-, y-, and z-directions. The differential operator is implemented by applying the discretization stencil $\frac{1}{2}\times[-1, 0, 1]$ along each of the three directions.

In the second stage, convolutional layers are used to project all the components of dynamic variable $\boldsymbol{x}$, input parameters $\boldsymbol{m}$, and physical feature $\boldsymbol{f}$ into a high-dimensional hidden feature $\boldsymbol{h}$. Each component of the hidden feature $\boldsymbol{h}$ has a dimensionality of d_h at the point $\boldsymbol{\omega}$. Hidden features are then passed to the operator blocks defined in Eqs. (4 - 9). The operator blocks return the high-dimensional representations of the accumulation and advection terms. The goal of extending the feature dimensionality before the operator blocks is to project the physical relationships in a high-dimensional space. Instead of applying nonlinear functions, the Hadamard product is used to introduce nonlinearity in the encoder.

In the third stage, we further map the accumulation and advection terms into a high-dimensional latent space with the dimension of d_l ($d_l > d_h$). Simultaneously, the latent variables are downsampled through the convolutional layers, reducing the spatial dimension by a factor of 2. The outputs of this stage are the latent variables $\boldsymbol{z}_{acc}^t$ and $\boldsymbol{z}_{adv}^t$. The downsampling is employed to increase the receptive field of the convolutional layers while reducing the GPU memory demand.

While the notation of latent variables does not specify the fluid phase, the latent variables are computed separately for the wetting and non-wetting phases and then concatenated. As a result, each latent variable sent to the next layer encompasses the latent representations for the two phases. As shown in Figure 2 (a), the dynamic input $\boldsymbol{x}$ is defined for the previous time step $t-1$, while the latent variable $\boldsymbol{z}$ corresponds to the current time step t. Therefore, the purpose of the encoder is to generate an initial estimate of the latent variables for time step t. These latent variables will be further updated by the processor, taking into account the external effect of the control variable.

The encoder $\mathcal{F}_{\theta_{\text{InputToLatent}}}$ can then be decomposed into three stages as follows:

$$\begin{aligned} \boldsymbol{f} &= \mathcal{F}_{\theta_{\text{InputToFeature}}}(\boldsymbol{x}, \boldsymbol{m}), \\ \boldsymbol{h} &= \mathcal{F}_{\theta_{\text{FeatureToHidden}}}(\boldsymbol{f}, \boldsymbol{x}, \boldsymbol{m}), \\ \{\boldsymbol{z}_{acc}, \boldsymbol{z}_{adv}\} &= \mathcal{F}_{\theta_{\text{HiddenToLatent}}}(\boldsymbol{h}), \end{aligned} \tag{11}$$

where $\mathcal{F}_{\theta_{\text{InputToFeature}}}$, $\mathcal{F}_{\theta_{\text{FeatureToHidden}}}$, and $\mathcal{F}_{\theta_{\text{HiddenToLatent}}}$ refer to the three stages of the physics-based encoder, respectively. The dimensions d_o, d_h, and d_l in these three stages are the hyperparameters defined by users. The selection of these hyperparameters and their feasible ranges are provided in Appendix B.

Neural networks parameterize the state-related operator in the first stage with d_o hidden units and learn the nonlinear dependencies with complicated formulas, such as Equation of State (EoS). In contrast to the parameterization of the state-related operator, the spatial and physical operator blocks are non-parametric and serve as hard constraints to enforce adherence to the underlying physics.

2.2.3 Residual-based Processor

The processor is implemented as a variant of recurrent neural network (RNN) with a customized recurrent unit, as illustrated in Figure 2 (b). The inputs to the processor are the governing terms in the latent space $\boldsymbol{z}^t$ to represent the accumulation $\boldsymbol{z}^t_{acc}$, advection $\boldsymbol{z}^t_{adv}$, and sink/source $\boldsymbol{z}^t_{src}$ for wetting and non-wetting phases. The latent representation of the sink/source term comes from the control encoder $\mathcal{F}_{\theta_{\text{ControlToLatent}}}$. Each customized recurrent unit, also referred to as a processor block, consists of N_{rl} residual layers that iteratively update the latent variables $\boldsymbol{z}^t_{acc}$, $\boldsymbol{z}^t_{adv}$, and $\boldsymbol{z}^t_{src}$, where N_{rl}, denotes the number of residual layers. To make the proposed model more memory-efficient and to increase the receptive field of the convolutional kernel, we further reduce the spatial dimension of the latent variables $\boldsymbol{z}^t_{acc}$, $\boldsymbol{z}^t_{adv}$, and $\boldsymbol{z}^t_{src} \in \mathbb{R}^{\mathcal{D}\times d_l}$ by a factor of 2 and simultaneously project them onto the feature space with a higher dimensionality of d_r ($d_r > d_l$). The resulting latent features are used to calculate the residual term $\delta\boldsymbol{z}^t$, which is then projected back to the original dimension $\mathcal{D} \times d_l$ and added to the latent variable $\boldsymbol{z}^t$.

The residual layer updates the latent variables based on the concept of the residual for governing equations. For a numerical solver, the Newton-Raphson method is commonly applied, which is written as:

$$\boldsymbol{J}(\boldsymbol{x}^k)(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) = -\boldsymbol{r}(\boldsymbol{x}^k), \tag{12}$$

where $\boldsymbol{J}$ is the Jacobian matrix at the inner iteration step k of the nonlinear solver; $\boldsymbol{r}(\boldsymbol{x}^k)$ denotes the residual of the governing equations for iteration k.

In this study, the design of the processor is inspired by the Newton-Raphson method. Instead of iteratively updating the dynamic variable $\boldsymbol{x}^k$, we update the latent variables and parameterize the update procedure using neural networks. First, we rewrite the Eq. (12) as follows:

$$\begin{aligned} \delta\boldsymbol{x}^k = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k &= -\left(\boldsymbol{J}^k\right)^{-1} \boldsymbol{r}^k \\ &\approx \mathcal{F}(\boldsymbol{r}^k, \boldsymbol{x}^k) \approx \mathcal{F}_{\theta_{\text{ResidualToDiff}}}(\boldsymbol{r}^k, \boldsymbol{z}^k). \end{aligned} \tag{13}$$

The dependency of input difference $\delta\boldsymbol{x}^k$ on terms $\boldsymbol{x}^k$ and $\boldsymbol{r}^k$ is parameterized by the operator $\mathcal{F}_{\theta_{\text{ResidualToDiff}}}$. To further facilitate the update, we introduce another operator $\mathcal{F}_{\theta_{\text{DiffToDiff}}}$ to convert the difference $\delta\boldsymbol{x}^k$ to the latent space, which is shown as follows:

$$\delta\boldsymbol{z}^k = \boldsymbol{z}^{k+1} - \boldsymbol{z}^k \approx \mathcal{F}_{\theta_{\text{DiffToDiff}}}(\delta\boldsymbol{x}^k), \tag{14}$$

where $\mathcal{F}_{\theta_{\text{DiffToDiff}}}$ approximates the mapping between the input difference $\delta \boldsymbol{x}^k$ and the latent difference $\delta \boldsymbol{z}^k$. Finally, $\delta \boldsymbol{z}^k$ can be written as:

$$\delta \boldsymbol{z}^k = \mathcal{F}_{\theta_{\text{DiffToDiff}}}(\mathcal{F}_{\theta_{\text{ResidualToDiff}}}(\boldsymbol{r}^k, \boldsymbol{z}^k)). \tag{15}$$

For brevity, we neglect the superscript k in the following derivation of the processor unit. For each residual layer within the processor unit, we first calculate the residual term $\boldsymbol{r}$ as follows:

$$\boldsymbol{r}^t = \boldsymbol{z}^t_{acc} - \boldsymbol{z}^{t-1}_{acc} + \boldsymbol{z}^t_{adv} + \boldsymbol{z}^t_{src}. \tag{16}$$

The above equation is similar to Eq. (3), with the distinction that each term in Eq. (3) corresponds to a specific phase. The latent variables in the above equation encompass representations for both phases. In the first residual layer of each processor block, the terms $\boldsymbol{z}^t_{acc}$ and $\boldsymbol{z}^{t-1}_{acc}$ in Eq. 16 are identical. In other words, the accumulation term $\boldsymbol{z}^{t-1}_{acc}$ sent from the previous processor will serve as both the previous state $\boldsymbol{z}^{t-1}_{acc}$ and the current state $\boldsymbol{z}^t_{acc}$. Consequently, the previous state $\boldsymbol{z}^{t-1}_{acc}$ remains fixed, while the current state $\boldsymbol{z}^t_{acc}$ is updated in subsequent residual layers.

After calculating the latent residual $\boldsymbol{r}^t$, each of the latent variables (i.e., $\boldsymbol{z}^t_{acc}$, $\boldsymbol{z}^t_{adv}$. and $\boldsymbol{z}^t_{src}$) is multiplied by $\boldsymbol{r}^t$, and their products are sent to the operator $\mathcal{F}_{\theta_{\text{ResidualToDiff}}}$ as inputs, expressed as follows:

$$\boldsymbol{r}^t_z = \mathcal{F}_{\theta_{\text{ResidualToDiff}}}\left(\begin{bmatrix} \boldsymbol{z}^t_{acc} \\ \boldsymbol{z}^t_{adv} \\ \boldsymbol{z}^t_{src} \end{bmatrix} \circ \begin{bmatrix} \boldsymbol{r}^t \\ \boldsymbol{r}^t \\ \boldsymbol{r}^t \end{bmatrix}\right). \tag{17}$$

In this step, the output $\boldsymbol{r}^t_z$ refers to the term $-\left(\boldsymbol{J}^k\right)^{-1}\boldsymbol{r}^k$ in the latent space. Then, the latent residual term $\delta \boldsymbol{z}^t$ can be derived as follows:

$$\boldsymbol{d}^t_z = \mathcal{F}_{\theta_{\text{LatentToDiff}}}(\boldsymbol{z}^t), \tag{18}$$

$$\delta \boldsymbol{z}^t = \mathcal{F}_{\theta_{\text{DiffToDiff}}}(\boldsymbol{d}^t_z \circ \boldsymbol{r}^t_z), \tag{19}$$

where the operator $\mathcal{F}_{\theta_{\text{LatentToDiff}}}$ is to transform the latent variable into a new term to introduce nonlinearity. Then, the operator $\mathcal{F}_{\theta_{\text{DiffToDiff}}}$ takes the product $\boldsymbol{d}^t_z \circ \boldsymbol{r}^t_z$ as input and returns the latent residual term $\delta \boldsymbol{z}^t$. Therefore, in each residual layer, the latent variable z^t is updated as

$$\boldsymbol{z}^t = \boldsymbol{z}^t + \delta \boldsymbol{z}^t. \tag{20}$$

In this study, the design of the residual layer is heuristic, as the updating procedure is performed in a high-dimensional latent space and is parameterized by neural networks. The high-dimensional representation enables the latent variable to capture comprehensive information about the input variables. The processor unit in this study is referred to as a residual-based processor, as it is composed of residual layers and draws inspiration from the concept of residuals in the governing equations.

3 Experimental Setup

In this study, we conduct a comprehensive evaluation of the proposed physics-encoded DL model through three distinct experiments. First, we assess the predictive capability of the model using unseen permeability inputs in Section 4.1 and compare our model with the Recurrent R-U-Net over the public dataset proposed by M. Tang et al. (2021). Second, we investigate the model's performance in the presence of unseen pairs of control variables and permeability inputs (Section 4.2). This assessment requires the model to generalize and handle complex and diverse scenarios. Lastly, we explore the model's

extrapolation capability by applying it to the prediction over the post-injection period while the training set only covers the injection period (Section 4.3). This extrapolation task poses a significant challenge due to the distinctive dynamics involved. In addition to these three experiments, we investigate the effect of the physics-based encoder on the latent representations of the governing terms by replacing the physics-based encoder with a traditional convolutional encoder (Section 4.4).

In addition to the comparison based on publicly available datasets (M. Tang et al., 2021), we also applied our model to a set of simulated datasets generated by a synthetic 3D simulation model of deep saline aquifer (Appendix C). The training datasets include simulation models for both single-well and two-well scenarios, introducing sufficient diversity in the dataset. However, we test our model on the two-well scenario only, as it is more complex than the one-well scenario. The simulation model consists of CO_2 injection over 15 years, followed by a post-injection period of 85 years. For our third experiment, we specifically utilize the initial 15 years of the post-injection as our test set. This selection is based on the observation that the pressure and plume dynamics reach a quasi-steady state beyond this time frame and do not show observable changes over time. The simulated dataset consists of 30 steps, where each step represents one year. During the injection, the well controls are randomly perturbed for each year while being constrained to have a total injection amount over 15 years. For detailed information regarding the simulation model and data generation, please refer to Appendix C.

In the examples with the public dataset, the models are trained to predict the whole simulation time with 10 steps by taking the initial state and permeability as inputs, following the setup in M. Tang et al. (2021). In the other experiments conducted in this study, however, the training process involves feeding the models with dynamic states from any time step and predicting only the subsequent 8 years rather than the entire 15 years. Exposing the model to dynamic inputs of different time frames can enable models to prevent overfitting and accurately learn the dynamics. Consequently, each training sample, comprising 16 time frames (15 years plus an initial state), is divided into eight data samples, each spanning 8 time frames. During testing, the DL models predict the entire 15 years over injection (or 30 years over both injection and post-injection periods) without taking any ground truth dynamic state as input.

In the experiment on the public dataset, we propose a modified version of Recurrent R-U-Net by making three key modifications to the architecture of the original model proposed by M. Tang et al. (2021). These modifications were aimed at improving the model's performance and aligning it with the setup of our model to ensure fair and meaningful comparisons. First, we replaced Batch Normalization (BatchNorm) (Ioffe & Szegedy, 2015) in the original model with Group Normalization (GroupNorm) (Y. Wu & He, 2018). This change was motivated by the observation that GroupNorm offers better generalizability and transferability compared to BatchNorm (Kolesnikov et al., 2020). Second, we introduced a dummy control input to the modified Recurrent R-U-Net and our model for the public dataset. The control variable in the public dataset is fixed over time and not included as input in the original Recurrent R-U-Net. This change is to make the use of modified Recurrent R-U-Net and our model consistent throughout this work. The dummy control variable for each time step is a 3D tensor, with values of 0 for grid blocks with no well, 1 for producers, and -1 for injectors. Lastly, we redefined the static variable $\boldsymbol{m}$ by 1) adding $\boldsymbol{I}_{PV}$ and $\boldsymbol{D}$ as part of the input, and 2) replacing permeability $\boldsymbol{K}$ with the term $\boldsymbol{T}_m$.

The pressure and injection rate are normalized using Min-Max normalization. Saturation is not normalized since it typically ranges between 0 and 1. To normalize the inverse grid volume $\boldsymbol{I}_{PV}$, we scale it by multiplying a reference grid volume of $50\times50\times2.5$ m^3. Using a uniform layer thickness, the depth $\boldsymbol{D}$ varies from 0 to 19, representing the depths of grid blocks from the first to the 20th layer. The term $\boldsymbol{T}_m$ contains zero values to represent the closed boundary and exhibits a right-skewed distribution with

a mean significantly higher than the median. Consequently, we apply the inverse hyperbolic sine transformation to transform $\boldsymbol{T}_m$ before feeding it into a deep learning model.

The training in this study employs the Adam optimizer. In our proposed model and the modified Recurrent R-U-Net, we apply the relative ℓ_2-loss (Wen et al., 2022) as the loss function, which is defined as follows:

$$L(\{\boldsymbol{S}, \boldsymbol{p}\}_{t_1:t_2}, \{\hat{\boldsymbol{S}}, \hat{\boldsymbol{p}}\}_{t_1:t_2}) = \frac{\|\boldsymbol{S}_{t_1:t_2} - \hat{\boldsymbol{S}}_{t_1:t_2}\|_2}{\|\boldsymbol{S}_{t_1:t_2}\|_2} + \frac{\|\boldsymbol{p}_{t_1:t_2} - \hat{\boldsymbol{p}}_{t_1:t_2}\|_2}{\|\boldsymbol{p}_{t_1:t_2}\|_2}, \tag{21}$$

where $\{\hat{\boldsymbol{S}}, \hat{\boldsymbol{p}}\}_{t_1:t_2}$ represents the predicted saturation and pressure over time steps from t_1 to t_2. In this work, the variance of the norms $\|\boldsymbol{S}_{t_1:t_2}\|_2$ or $\|\boldsymbol{p}_{t_1:t_2}\|_2$ are considerably high due to the dynamic range of time frames, which can vary widely. Using the mean squared error (MSE) loss function encourages inaccurate predictions at the start of injection. This is mainly because the saturation of the non-wetting phase in the initial time frames is inconsequential. Deep learning models that fail to predict the saturation of these time frames will still generate low values of MSE loss. The choice of relative error as the loss function will mitigate this issue. The details of the training process and configuration are provided in Appendix B.

4 Results and Discussion

4.1 Testing on Unseen Permeability

In this experiment, we first compare three models on the public dataset (M. Tang et al., 2021): the original Recurrent R-U-Net (the baseline model), the modified Recurrent R-U-Net, and our proposed model. The dataset consists of 2923 samples spanning 1000 days and is generated by a numerical simulation model of a two-phase oil reservoir for a waterflooding problem. In this experiment, models are trained to predict oil saturation using different amounts of training samples: 200, 500, 1000, 1500, 2000, and 2500. A separate set of 50 samples is used for model validation, while the remaining 373 samples form the test set.

Figure 4 (a) shows that our model consistently achieves lower test losses across all sizes of training samples compared to the other two models. Notably, our model trained on 1000 samples demonstrates a comparable root mean square error (RMSE) of saturation prediction to the baseline model trained on 2500 samples. Figure 4 (b) demonstrates the distribution of saturation RMSE for each time step, where the baseline model exhibits higher mean and variance in test losses across time steps compared to our model. To further evaluate the saturation predictions, Figure 5 visualizes the 3D saturation prediction at the 10th time step. The baseline model trained with 200 samples shows a significant mismatch in the saturation front. Both modified Recurrent R-U-Net and our model accurately capture the saturation front for both the 200 and 2500 training sample cases with lower RMSE values. In the subsequent experiments, we will continue using the modified Recurrent R-U-Net for comparison and exclude the original version. The original Recurrent R-U-Net has a different setup of input and does not support time-varying control variables.

Next, we proceed to evaluate our model in the simulation model of subsurface CO_2 storage reservoir. Our proposed model and modified Recurrent R-U-Net are trained using 700 simulated samples to predict pressure and CO_2 saturation over a 15-year injection. The 700 simulated samples are then divided into 5600 training data samples, each of which spans eight years. The validation and test sets consist of 100 and 200 simulated samples, respectively. The training and test sets have the same injection controls but different permeability. For brevity, we will refer to the modified Recurrent R-U-Net as RUNET in the following experiments. As shown in Figure 6, our proposed model consistently exhibits lower mean and variance of RMSE compared to RUNET. It is worth noting that

the saturation error exhibits increasing trends over time due to the spreading of the saturation front. In contrast, the pressure error is more stationary, attributed to the presence of an aquifer region surrounding the storage reservoir, which facilitates pressure dissipation and maintains it within a certain range.

Figures 7 and 8 provide 3D visualizations of saturation and pressure distributions, respectively. The errors in both saturation and pressure tend to occur in areas where there are significant changes (or gradients). Specifically, for saturation, the errors are prominent near the front of the CO_2 plume, where the saturation values experience rapid transitions. On the other hand, for pressure, the errors are more noticeable in the vicinity of the wells. The 3D pressure map changes significantly over time due to variations in injection control. As reflected in Figures 7 and 8, our model outperforms the RUNET in capturing these two distinct patterns.

4.2 Testing on Unseen Control and Permeability

In this experiment, we further evaluate the performances of two models (our model and RUNET) on the test set where both control and permeability are unseen during training. The training, validation, and test sets still consist of 700, 100, and 200 simulated samples, respectively. As depicted in Figure 9, the distribution of injection rates differs between the training and test sets, even though both datasets have the same constraint of total injection rate. Notably, the allocation of injection rates in the test set exhibits more significant variations compared to the training set, demonstrating a less balanced allocation of injection rates. The allocation of the injection rate for the test set falls outside the distribution covered by the training set. For control optimization problems, the control variable can evolve towards its extremity during the process of optimization (Qin et al., 2023). The out-of-distribution injection rate resembles the challenges caused by the extremity of the control variable in the optimization task and challenges the model's robustness in handling diverse control scenarios (Qin et al., 2024).

Figure 10 illustrates that the prediction performances of the two models are comparable in predicting pore pressure, while our model is more accurate in predicting saturation than RUNET. Given the variations in control variables in the test set, predicting saturation and pressure becomes more challenging for both models compared to the previous experiment. Figures 11 and 12 provide visualizations of saturation and pore pressure predictions for both models. In Figure 11, it can be observed that the saturation prediction from RUNET shows significant errors around the top layer, leading to physically inconsistent results. In contrast, our model exhibits only slight discrepancies near the saturation front in the top layer, indicating its improved accuracy and ability to maintain consistency with the underlying physics. As shown in Figure 12, the pressure prediction from RUNET exhibits errors that are evenly distributed around the wells, while the prediction error from our model is more concentrated. This distinct behavior between the two models may be attributed to differences in their architectures and will be further investigated in our future work.

4.3 Extrapolation over Post-Injection

In the previous experiments, the deep learning models were trained and tested over a 15-year injection. During testing, DL models predicted the future states over the same period based on different sets of inputs. In this experiment, we extend the application of these trained models to predict 30 years, including both injection and post-injection. Specifically, models trained in previous experiments (Sections 4.1 and 4.2) are directly applied to predict the last 15 years, from the 16th to the 30th year, with the initial state to be predicted saturation and pressure in the 15th year. During post-injection, injection rates are set to zero.

In Figure 13, the prediction performances of two models are evaluated for both injection and post-injection periods. It can be observed that our model consistently exhibits significantly lower prediction errors for saturation than RUNET. Furthermore, our model's prediction errors remain relatively stable during post-injection, whereas errors from RUNET keep increasing for the saturation prediction. On the other hand, the pore pressure of the entire reservoir becomes more uniform and converges to the boundary pressure during post-injection. Therefore, pressure prediction becomes less challenging for the two models, as both exhibit decreasing prediction errors after the 15th year when the post-injection starts. Despite the comparable performance of pressure prediction for both models during the injection period (Figure 13 (d)), our model consistently outperforms RUNET in terms of lower mean and variance of prediction errors during post-injection. This indicates the superior robustness and accuracy of our model in handling the distinct dynamics of the post-injection period.

Figure 14 visualizes three examples of the saturation predictions for two models in the 15th and 30th years. It can be observed from all three examples that, during the post-injection period, CO_2 is dominantly driven by the buoyant force and gets accumulated at the top layer, which is overburdened by cap rock. A common failure of RUNET is its physically inconsistent prediction during extrapolation. As shown in the first two examples, the predicted CO_2 plume from RUNET in the 30th year shows disconnected patterns and deviates significantly from the ground truth. In the last two examples, the prediction from RUNET exhibits a relatively low RMSE in the 15th year, which is close to the error of our model. However, RUNET still fails to capture the CO_2 migration during post-injection and even shrinks the CO_2 plume toward the end of the prediction. On the other hand, our model can accurately extrapolate the CO_2 plume during extrapolation. Over three models, our model accurately predicts the saturation front compared with a slight increase in RMSE.

Figure 15 visualizes the pressure prediction of the 17th layer of the reservoir for two models during post-injection. It can be observed that the pressure dissipates during post-injection and reaches the aquifer boundary, which serves as a constant boundary condition due to its extensive volume. In contrast to RUNET, our model demonstrates an accurate prediction of the pressure dissipation and eventual convergence to the aquifer pressure. These findings underscore the superior performance of our model in capturing long-term behavior and accurately predicting the saturation evolution beyond the injection period.

It is important to note that predicting the post-injection period is an extrapolation task, as it falls outside the time range covered by the training set. Extrapolation tasks can pose challenges for deep learning models, as the statistical input-output relationship during extrapolation may deviate from what was learned from the training set. In the case of CO_2 migration, the dominant driving force changes between the injection and post-injection periods, which can result in a different statistical input-output relationship. This change in driving force can undermine the prediction performance of deep learning models. Therefore, accurate prediction during post-injection requires the models to capture the underlying physics of the system, rather than solely relying on the learned statistical relationship from the training set.

4.4 Investigation of Latent Representation

To further investigate the effect of introducing the physics-based encoder on the prediction performance, we introduce an additional DL model. This model shares the identical modules as the proposed physics-encoded DL model, with the exception that the physics-based encoder is replaced with convolutional encoders. This alteration facilitates a more focused investigation of the effect. The convolutional encoder comprises two separate encoders to produce the latent representations of accumulation and advec-

tion terms, respectively. Each encoder shares a similar architecture as the encoder in the RUNET. In the case of the accumulation term, the input exclusively encompasses the initial states of dynamic variables (i.e., pressure and saturation). As for the advection term, dynamic and static variables are concatenated together and fed into the encoder as input. Subsequently, the latent variables representing the accumulation and advection terms are then sent to the processor blocks, the same as the pathway of the physics-based encoder.

Figure 16 illustrates the comparison between two models: one equipped with the physics-based encoder (referred to as "Ours") and the other with convolutional encoders (referred to as "ConvEnc"). Both models are trained and tested using the same setup as the second experiment (Sec. 4.2). The results show that both models demonstrate similar RMSE values for pressure and saturation during injection. However, our model consistently outperforms ConvEnc during the post-injection period. On the other hand, as shown in Figure 17, ConvEnc significantly outperforms RUNET in predicting pressure and saturation, encompassing both injection and post-injection periods. Given that all three models possess a similar number of trainable parameters (over 3 million), these results exhibit the superior performances of the physics-based encoder and the residual-based processor in contrast to the convolutional encoder and ConvLSTM, respectively.

Since the governing terms follow certain PDE constraints, it is essential to construct a compact latent space. The latent representations of accumulation and advection terms are expected to exhibit similar patterns with the true governing terms to contain physically consistent features. Therefore, we further investigate the latent representations of two governing terms derived from our model and ConvEnc and discuss their physical meanings. Figure 18 illustrates an example of the latent representations of accumulation (Figure 18 (a)) and advection terms (Figure 18 (b)) over the first 15 years for both our model and ConvEnc. The collected latent representations are the outputs of the last processor layer. To reduce the redundancy in latent representations, we first employ Principal Component Analysis (PCA) on d_l latent variables ($d_l = 32$) for each time step separately. Here, only the first four principal components (PCs) are visualized in Figure 18 for brevity. Moreover, Figures 18 (a) and 18 (b) present the actual values of the wetting-phase accumulation and advection in the vertical direction in the 13th year as references.

As shown in Figure 18 (a), the true accumulation term is dominantly affected by saturation (Figure 18 (d)), with only slight variations outside the CO_2 plume as affected by pressure (Figure 18 (c)). This slight variation is because the density of liquid-phase water is insensitive to pressure changes and that reservoir porosity is set to be homogeneous in this work. The components from our model exhibit similar patterns with pressure contour and saturation front, as reflected in the first and third components. The components from ConvEnc also reflect these two features, as shown in the first PC. However, as depicted in the second PCs, the latent accumulation terms in ConvEnc exhibit high-frequency patterns similar to permeability. This arises from the updating process within the processor blocks, where the latent accumulation and advection mutually influence each other. These high-frequency patterns are distinct from the accumulation terms depicted in Figure 18 (a). The residual-based processor model passes the latent accumulation to the next time step to approximate the dynamics of the mass balance governing equation. Latent representations that are not aligned with the true governing terms could hinder the learning process and lead to inefficient learning of the dynamics.

Figure 18 (b) illustrates the PCs of latent representations of the advection term for both models. Notably, the latent advection terms generated by our model closely align with the true values and exhibit high-fidelity patterns. In contrast, ConvEnc produces overly smoothed latent advection patterns. This distinction between the two models may stem from the inclusion of the physics-based encoder. Specifically, in our model, the overall receptive field for permeability spans only three grid blocks. This design choice pre-

vents over-smoothness during the calculation of the latent advection term, thus preserving intricate details. On the other hand, ConvEnc's receptive field follows the default setup of RUNET and spans 12 blocks, resulting in an inconsistent inductive bias with the advection term. While ConvEnc can learn high-fidelity patterns from the data, it requires more data to constrain its behavior, hindering efficient learning. Furthermore, components within ConvEnc's latent advection term exhibit temporal inconsistency, with notable shifts in patterns observed in the third PCs. This behavior presumably correlates with the unexpected updating caused by physically inconsistent latent variables. Despite our model's superiority over ConvEnc, we do not dive into introducing any metrics to evaluate the physical consistency of latent variables. This aspect extends beyond the scope of this work and could potentially be explored in future endeavors.

5 Conclusion

In this work, we propose novel DL architectures that incorporate the physics of the system for accurate spatial-temporal prediction of pressure and saturation in geologic CO_2 storage. Traditional deep learning models, which solely rely on learning statistical input-output relationships, face challenges in extrapolation, require large amounts of training data, and can produce physically inconsistent predictions. Additionally, these models struggle to adapt to general input-output setups involving diverse types of inputs and outputs. To address these limitations, we introduce a general approach that efficiently learns the underlying physics by encoding physics into the encoder and processor. We demonstrate the ability to learn the underlying physics, handle permeability variations and time-varying controls, and make predictions over both the injection and post-injection periods. As one of the main contributions, the physics-based encoder generates physically meaningful latent variables through physical operators and high-dimensional projection. The physical operators constrain the dependencies of latent variables on inputs and reduce the search space for the model's parameter. The operators within the residual-based processor update the high-dimensional latent variables in a coupled manner. The updating process within high-dimensional latent space enables the integration of nonlinear and coupled PDEs, bypassing the need for assumption and simplification that are necessary for direct updating pressure and saturation.

The experimental results validate the effectiveness of our proposed model in addressing the aforementioned limitations faced by traditional deep learning models. Our model exhibits data efficiency, as demonstrated on a public dataset (Section 4.1), and consistently outperforms traditional deep learning models when handling various types of inputs (Sections 4.1 and 4.2). The physical operators enable our model to approximate the underlying physics and learn the complex input-output relationship efficiently. Furthermore, our model exhibits superior performance compared to RUNET in the extrapolation to the post-injection period (Section 4.3), where RUNET fails to extrapolate the saturation front and generates physically inconsistent predictions. The recurrent architecture, along with regular time intervals for each recurrent unit, allows our model and RUNET to effectively predict over arbitrary lengths of time steps. However, in contrast to RUNET which lacks interpretability and remains a black box, our model comprises modular components that serve specific roles in the evolution of dynamics.

The generalizability and applicability of our model extend beyond geologic CO_2 storage and the specific scenarios studied in this work. Notably, the public dataset used in Section 4.1 focuses on a waterflooding problem with producers and injectors controlled by BHP. Therefore, the proposed architecture is not limited to geologic CO_2 storage or a specific type of well control. Instead, it can be adapted and applied to other geoscience and engineering domains where accurate spatial-temporal predictions are crucial, such as waterflooding and geothermal applications. The model's generalizability of handling different reservoir settings and time-varying controls makes it a valuable tool for decision-making and inverse modeling across a wide range of applications. Moreover, the gener-

alizability of the DL model could potentially address the challenge of limited available datasets in scientific domains, where data acquisition is often expensive and relatively scarce compared to domains like computer vision and natural language processing. The available datasets online typically differ in input setups, and traditional deep learning models are only applicable to a certain instance. Deep learning models with flexible and general-purpose architectures, such as ours, can make efficient use of the available data and alleviate the limitations imposed by data scarcity. This is particularly crucial for the development and practical application of powerful yet typically large deep learning models in real-world scenarios.

Although our model primarily focuses on variations in control and permeability, this work provides a general and flexible approach that can be extended to various inputs in future studies (e.g., heterogeneous porosity, different well locations, and different reservoir sizes). Another promising future work is to integrate physics-informed loss functions into our model to enable data-free training or zero-shot learning. PINN demonstrates proficiency in learning simpler tasks but faces limitations with complex tasks (Krishnapriyan et al., 2021). For the geologic CO_2 storage, the nonlinearity and complexity of PDEs could hinder the training of PINN. One potential solution to address this limitation is to first pretrain the model with labels and subsequently fine-tune it using a physics-informed loss function.

Figures

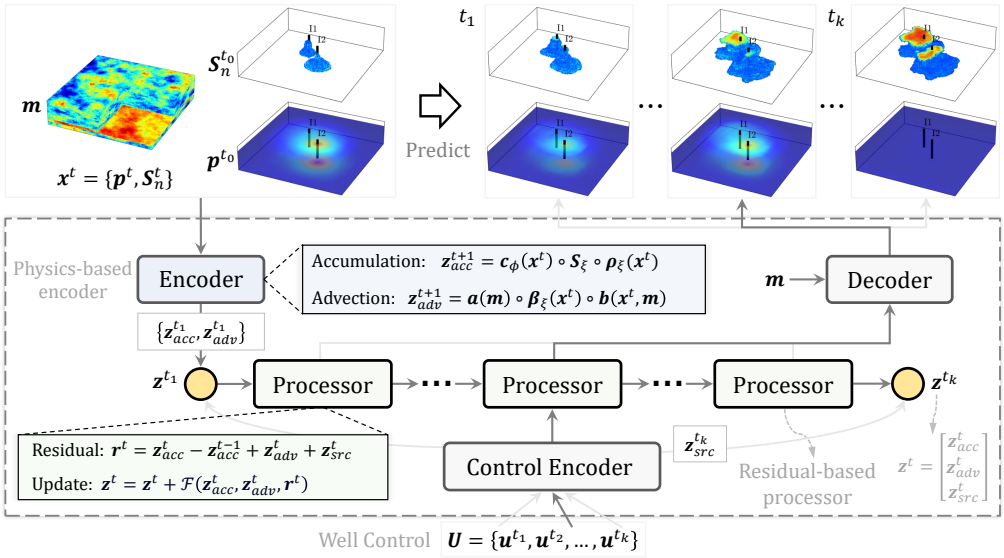

Figure 1: Overview of the proposed Fluid Flow-based Deep Learning (FFDL) model for predicting spatial-temporal pressure and saturation in geologic CO_2 storage. The process is applied to the entire reservoir with four modules: the physics-based encoder and control encoder to project input into physically meaningful latent variables; the residual-based processor to evolve the latent variables given well controls; and the decoder to project latent variables back to pressure and saturation.

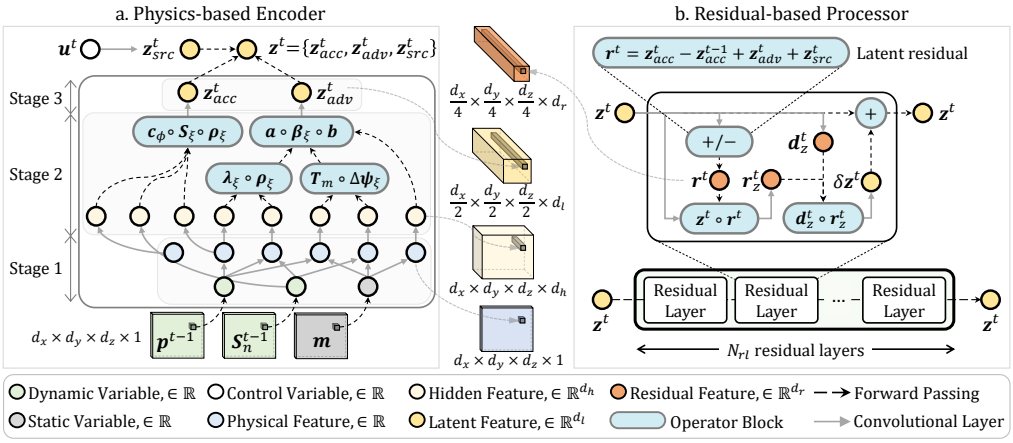

Figure 2: Overview of the physics-based encoder (left) and the residual-based processor (right). Each circle represents a variable corresponding to a single voxel in the 3D representation of the reservoir. The encoder consists of three stages: 1) mapping the inputs (dynamic and static) to physical features, 2) mapping previous features to hidden features, and 3) calculating physical latent variables. For each time step, the processor utilizes a series of residual layers to iteratively update the latent variables. The operator block incorporates non-parametric operators, including Hadamard product, addition, and subtraction. The forward passing operation involves directly passing the variable to the next layer without any additional transformations or modifications.

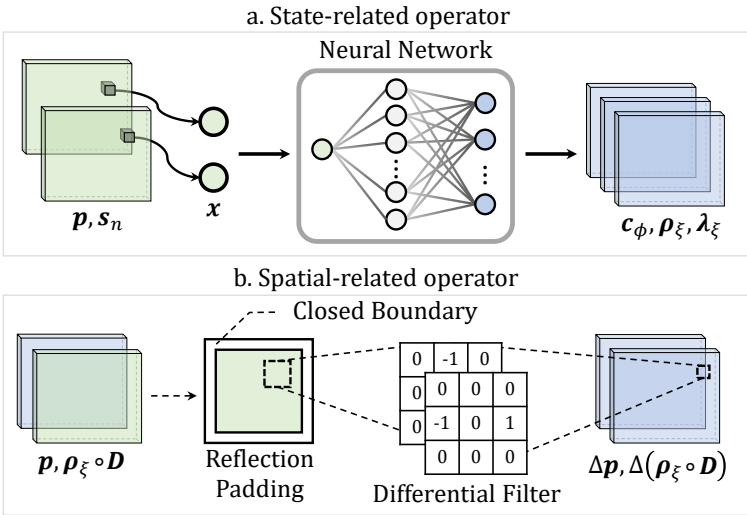

Figure 3: Illustration of (a) state-related operator and (b) spatial-related operator in the first stage of the physics-based encoder. The state-related operator is parameterized by a fully connected neural network. The differential pressure (or fluid potential) is calculated under a closed boundary, which is approximated by reflection padding.

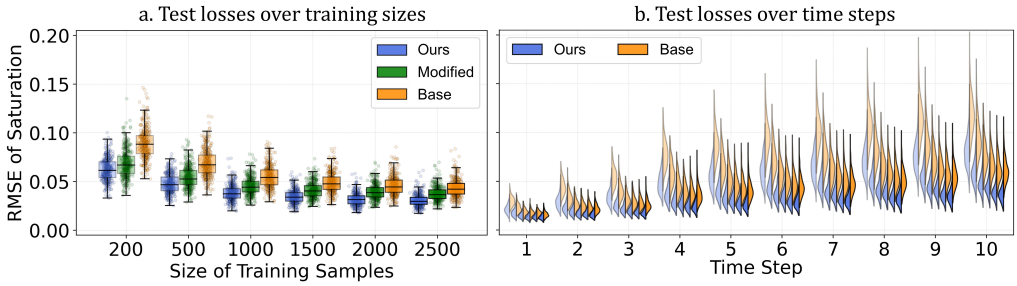

Figure 4: Distributions of test losses of saturation for different models. a. Test losses for different sizes of training samples. b. Test losses of saturation for different time steps. The different transparencies in (a) represent the various sizes of training samples ranging from 200 to 2500 from left to right. Each time step in (b) contains the six bars to represent distributions of RMSE values for the cases of training samples ranging from 200 to 2500 from left to right. "Modified" refers to the modified Recurrent R-U-Net. "Base" refers to the baseline R-U-Net. "Ours" refers to our model.

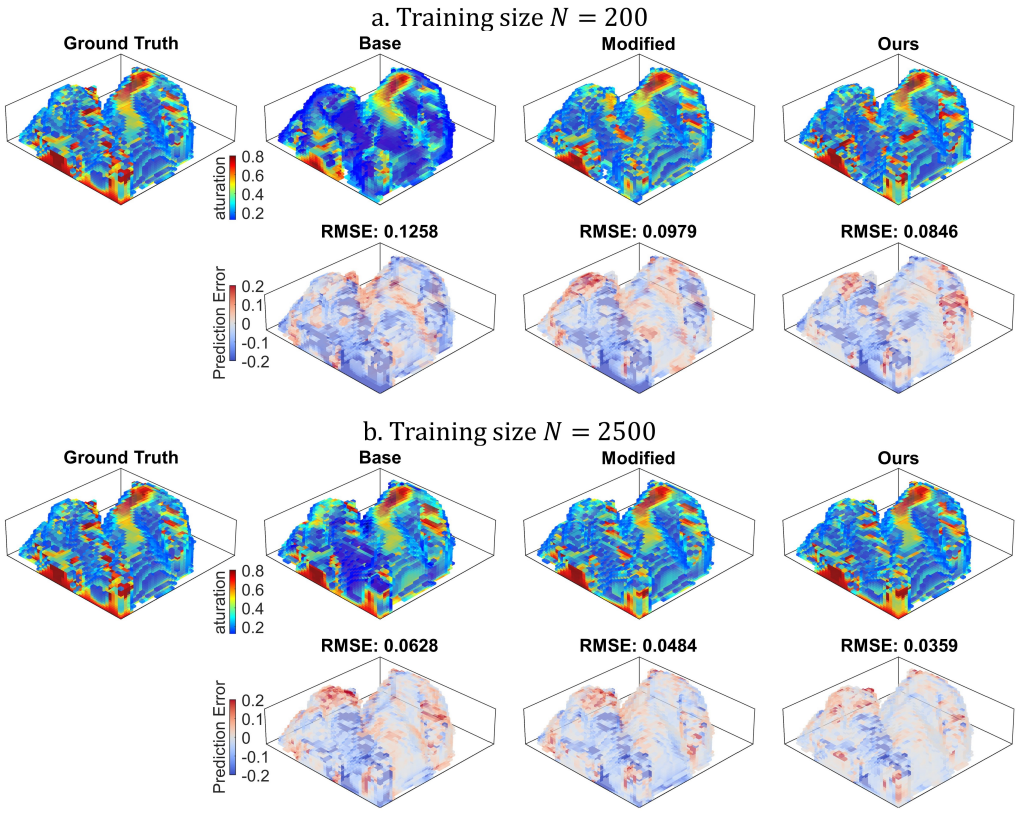

Figure 5: Visualization of predictions of oil saturation exceeding 0.15 at the last time step for three different models trained with varying numbers of training samples: (a) Model trained with 200 samples; (b) Model trained with 2500 samples.

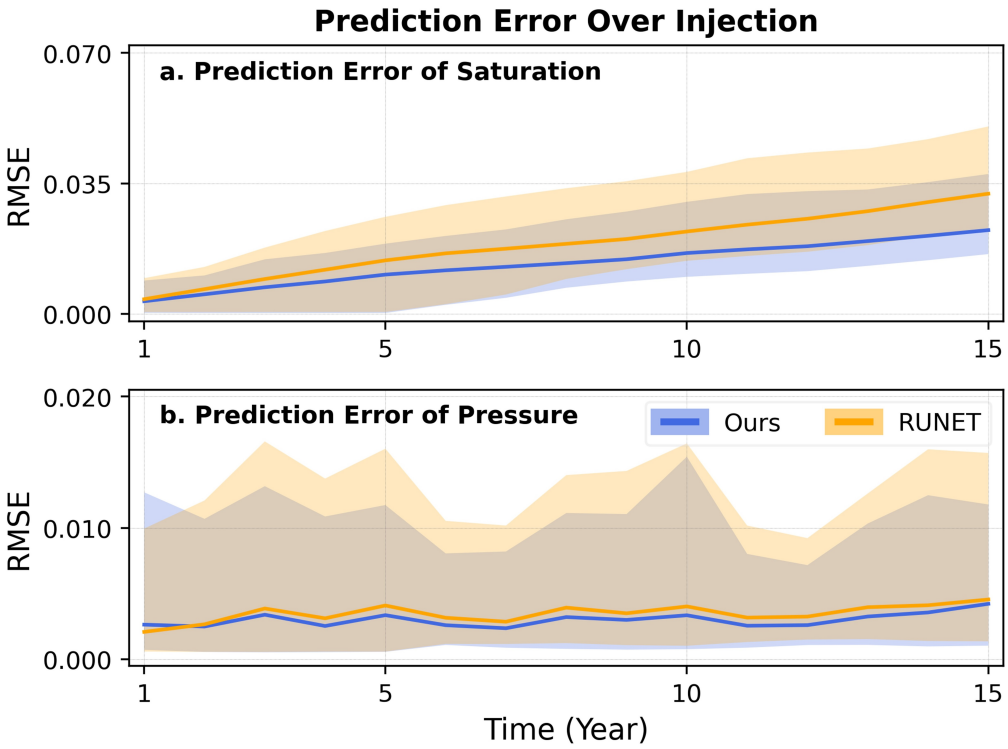

Figure 6: Prediction errors of normalized saturation and normalized pressure over the test set for two models: our proposed model (Ours) and modified Recurrent R-U-Net (RUNET). The error band and solid line represent a 95% confidence interval and the median of RMSE, respectively.

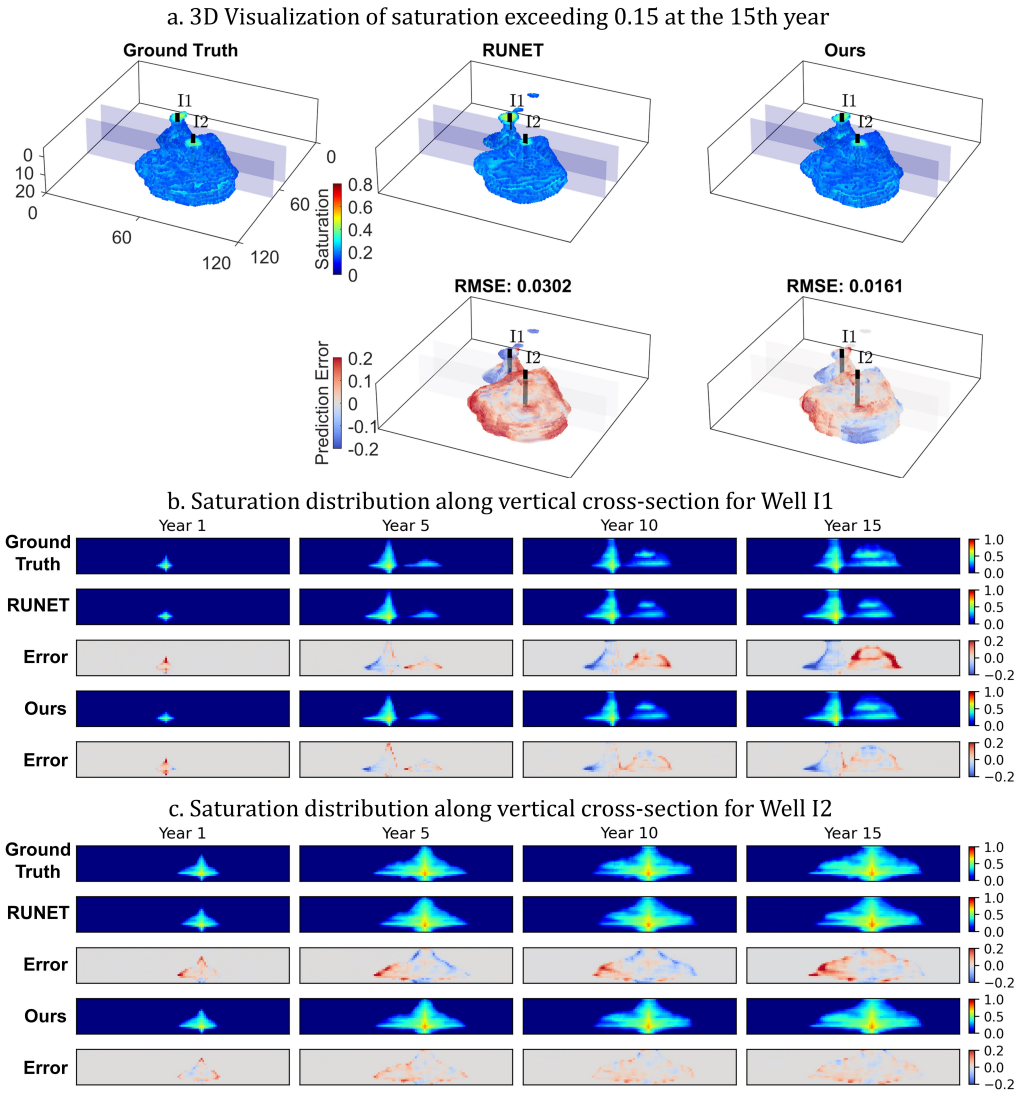

Figure 7: Visualization of saturation prediction over unseen permeability. The 3D visualization in (a) displays the saturation distribution in the 15th year. In (b) and (c), the 2D saturation distributions represent the x-z planes intersecting Wells I1 and I2, respectively. These planes are depicted as transparent cross-sections in (a).

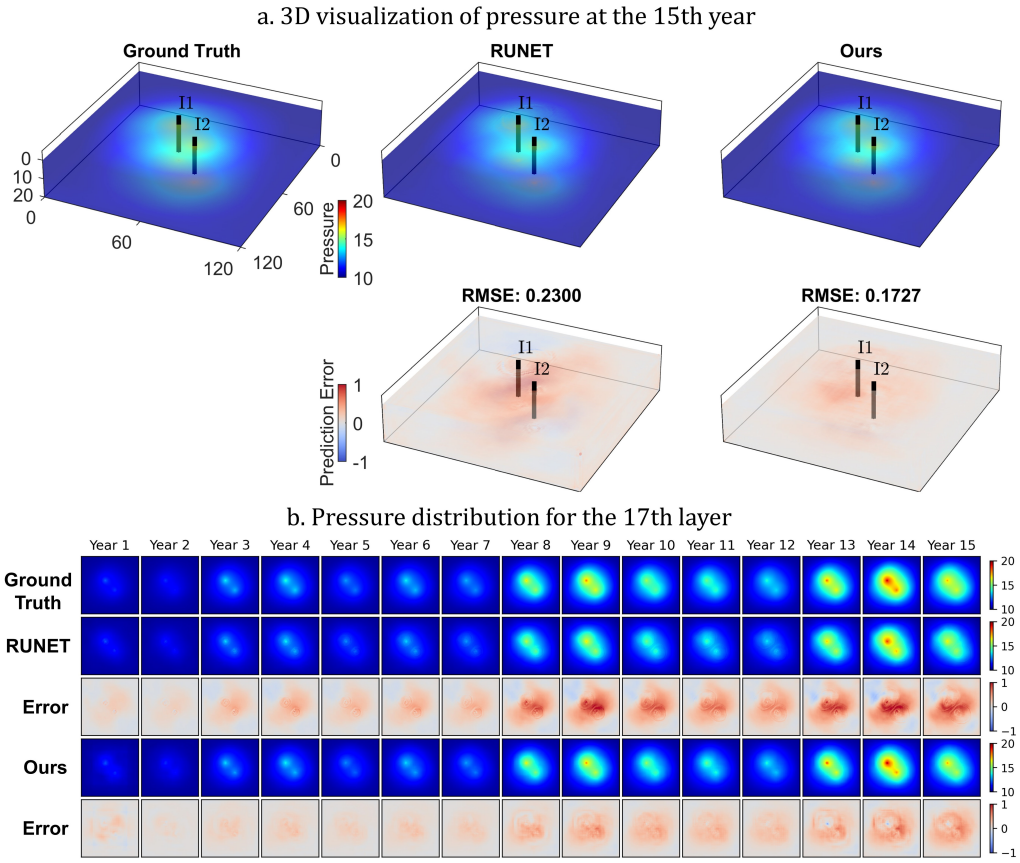

Figure 8: Visualization of pressure prediction over unseen permeability. In (b), the 2D pressure distributions represent the 17th layer of the reservoir, the layer where injection wells are located. The pressure values are measured in MPa, and the RMSE is also reported in MPa.

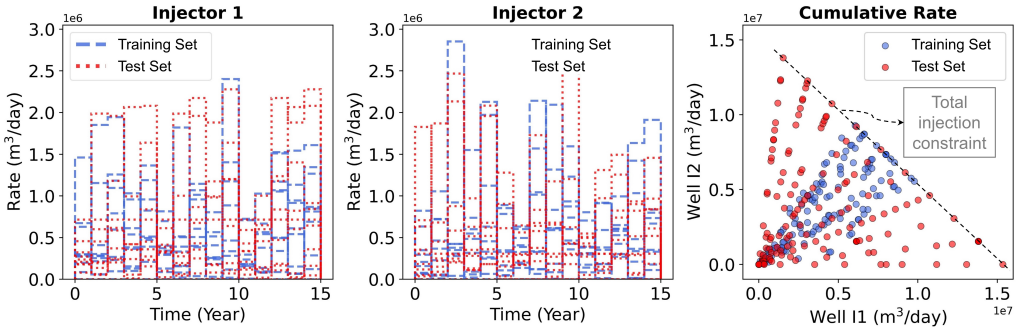

Figure 9: Visualization of injection control in the training and test sets. The left and middle figures depict the time-series injection rates for wells I1 and I2, respectively. The right figure shows the QQ-plot of cumulative injection rates for the two wells, illustrating the differences in the distribution between the two datasets.

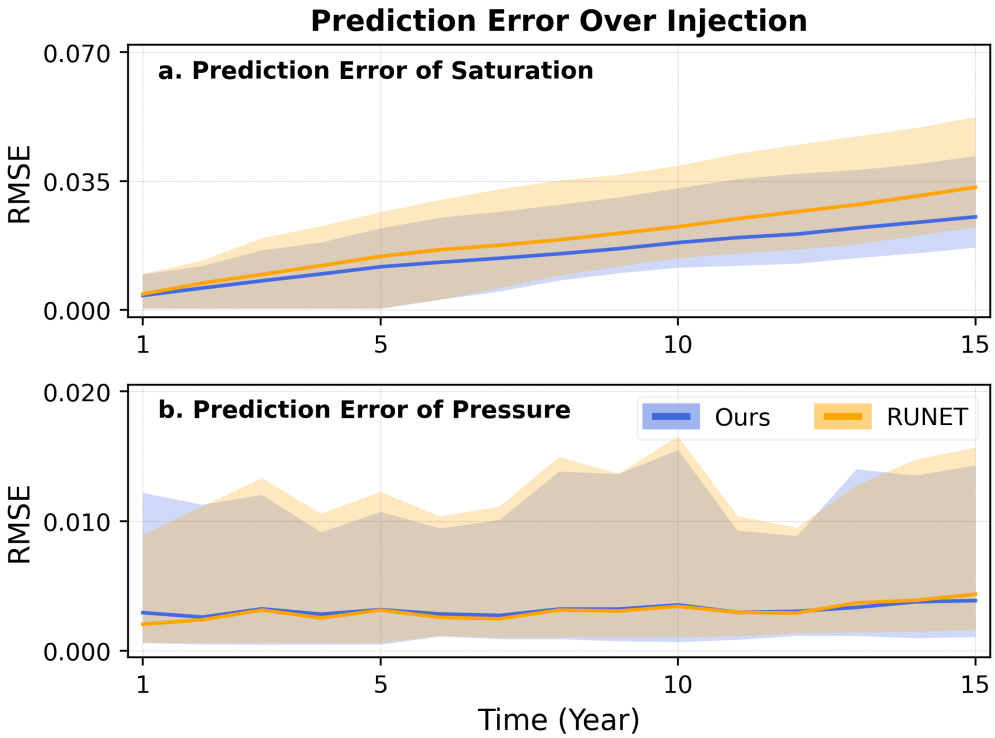

Figure 10: Prediction errors of normalized saturation and pressure over the test set for two models: our proposed model (Ours) and modified Recurrent R-U-Net (RUNET). The error band and solid line represent a 95% confidence interval and the median of RMSE, respectively.

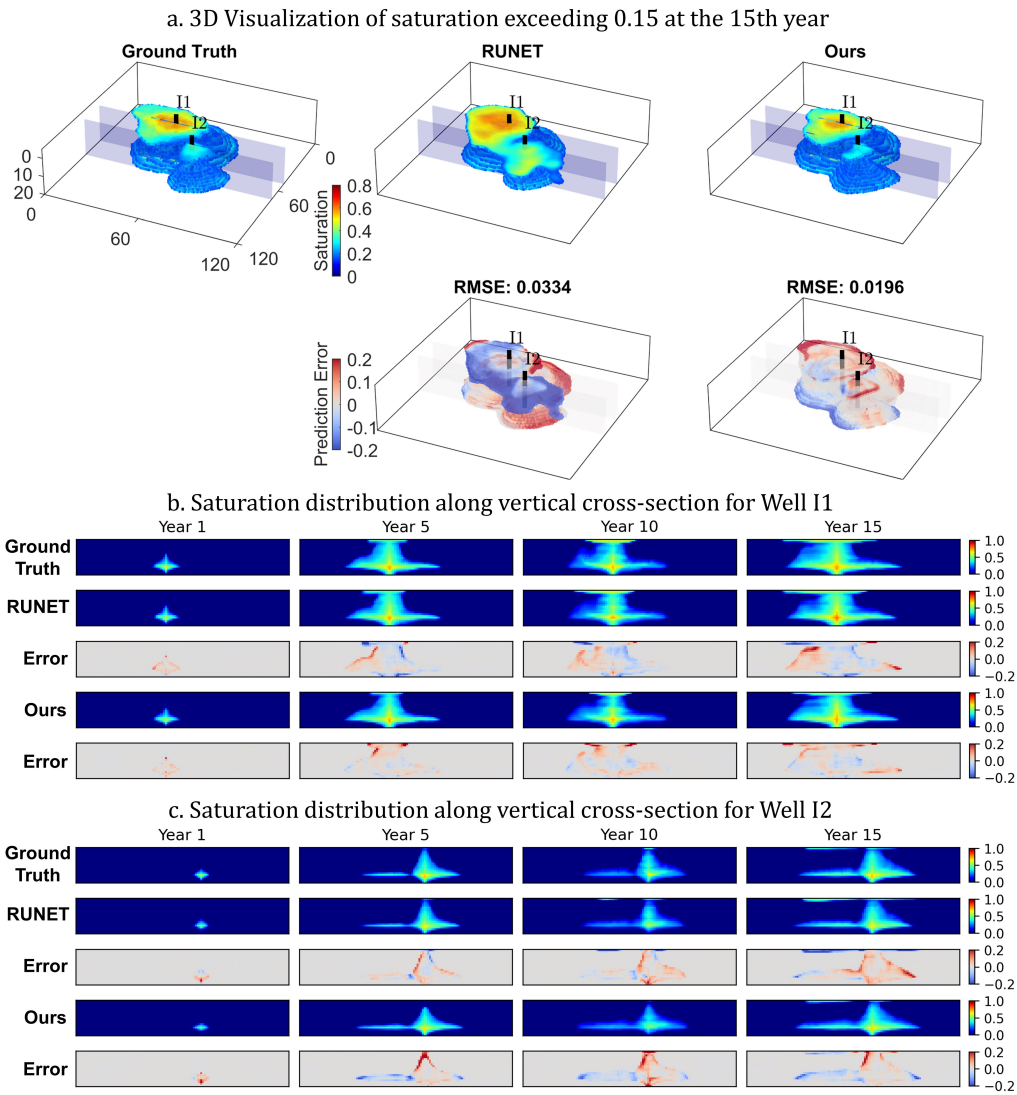

Figure 11: Visualization of saturation prediction over unseen control and permeability. The prediction from RUNET exhibits physical inconsistency and shows a significant mismatch in the shape of saturation.

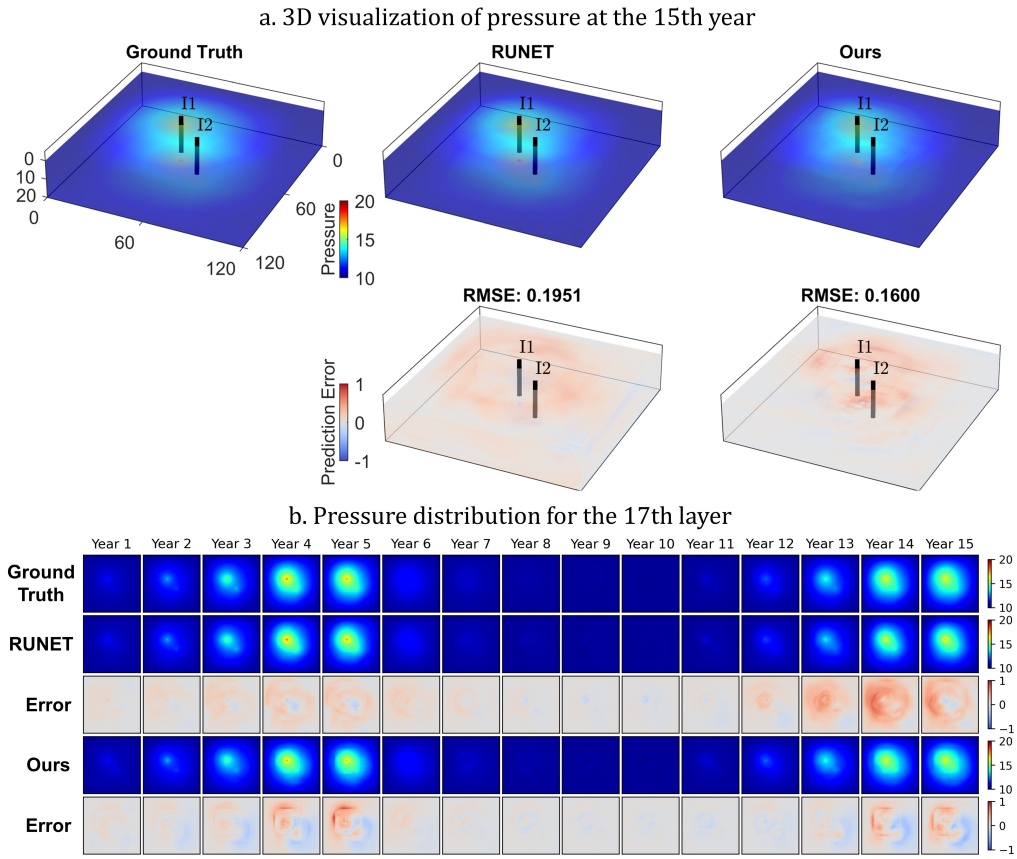

Figure 12: Visualization of pressure prediction over unseen control and permeability. The pressure values are measured in MPa, and the RMSE is also reported in MPa.

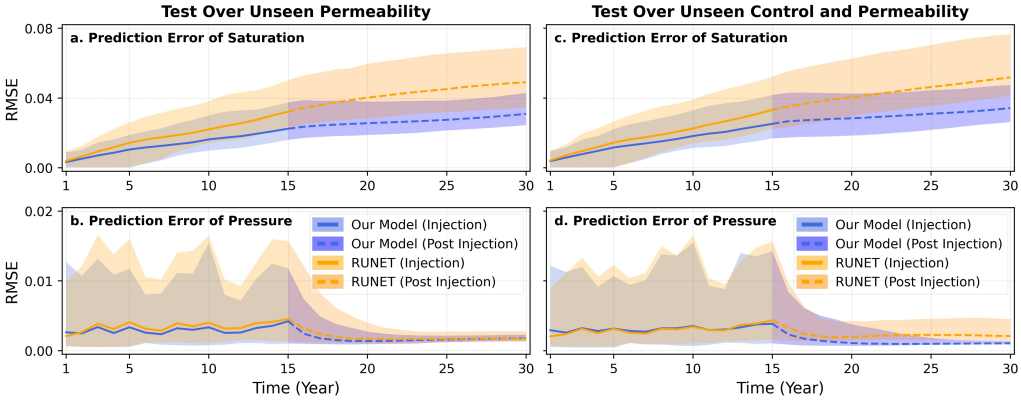

Figure 13: Prediction errors of normalized saturation and normalized pressure over injection and post-injection for the examples of (left) unseen permeability and (right) unseen control and permeability. The first 15 years denote the injection, while the last 15 years denote the post-injection where injection rates are zeros for two wells.

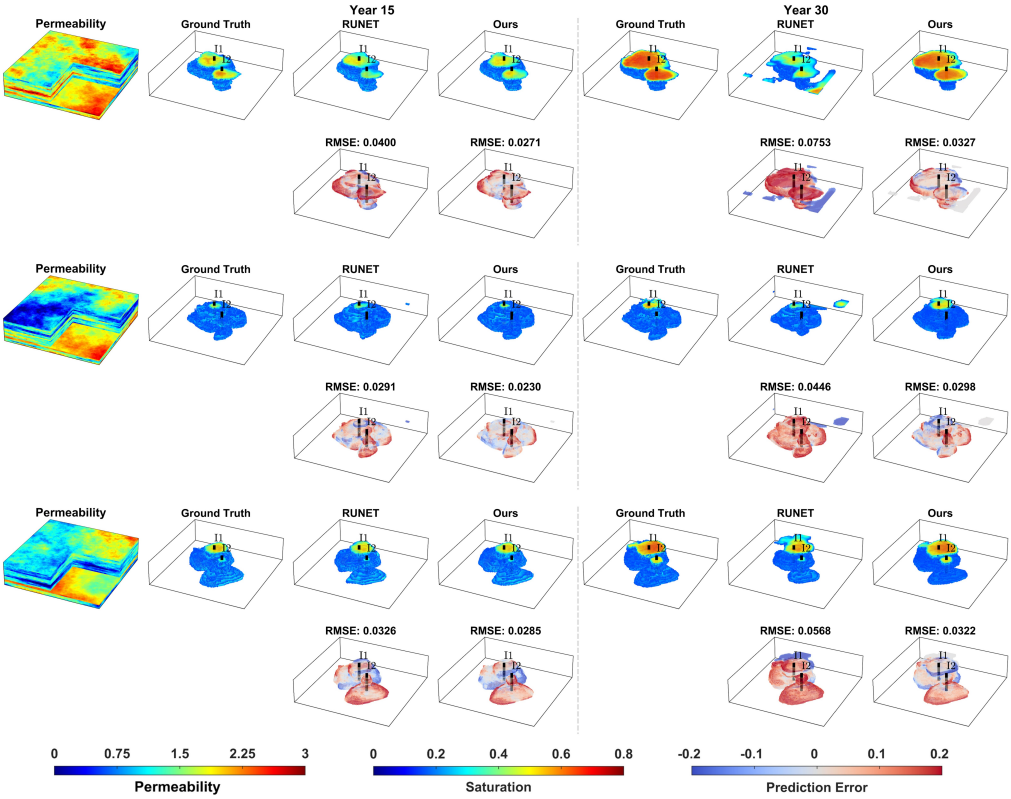

Figure 14: Visualization of saturation prediction of three different cases in the 15th and 30th years. Each case has different permeability and injection control as inputs. The permeability map is in the unit of $\log_{10}(mD)$.

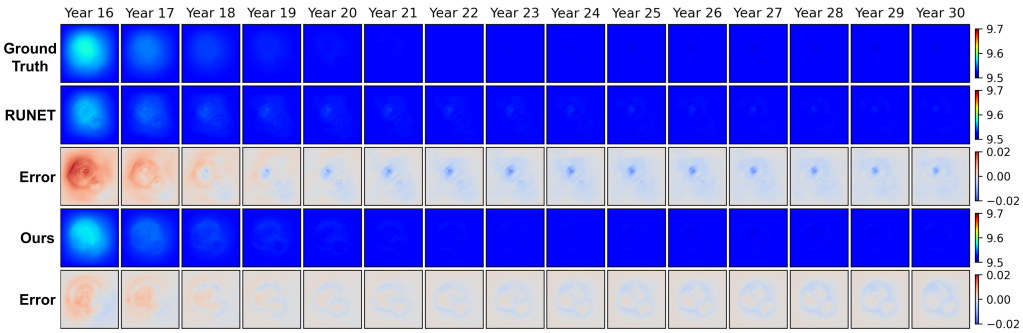

Figure 15: Visualization of pressure prediction over the post-injection period.

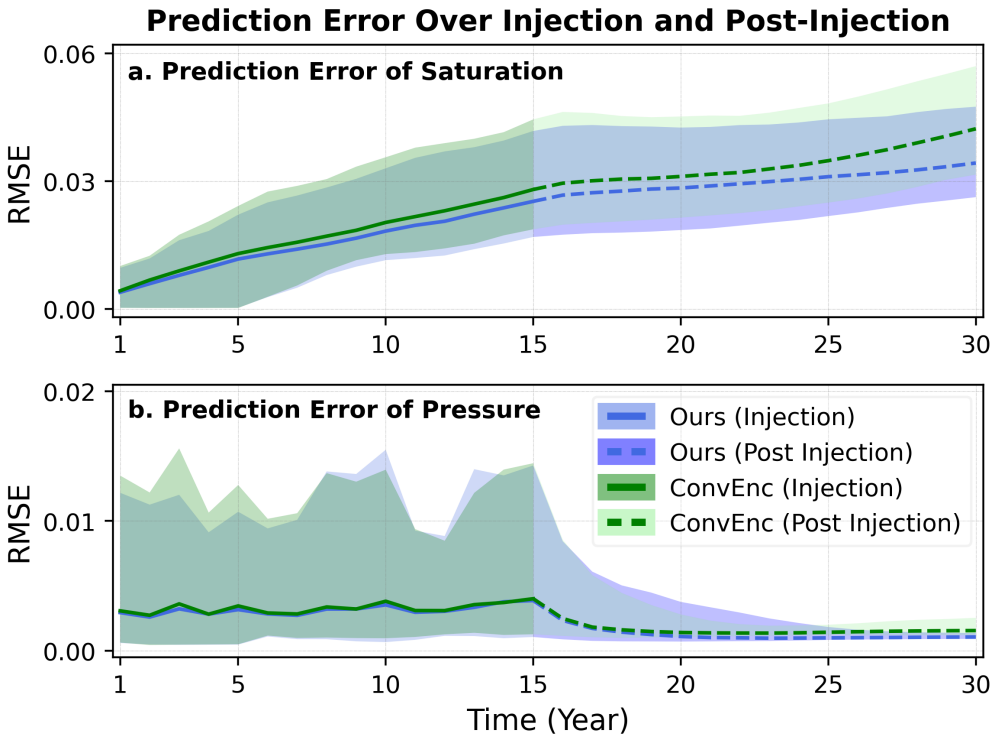

Figure 16: Prediction errors of normalized saturation and normalized pressure over injection and post-injection for two predictive models with different encoders: convolutional and physics-based.

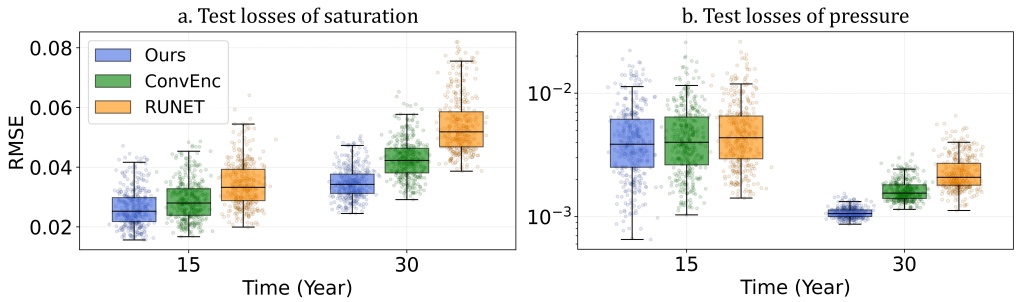

Figure 17: Distributions of test losses of (a) saturation and (b) pressure for different models.

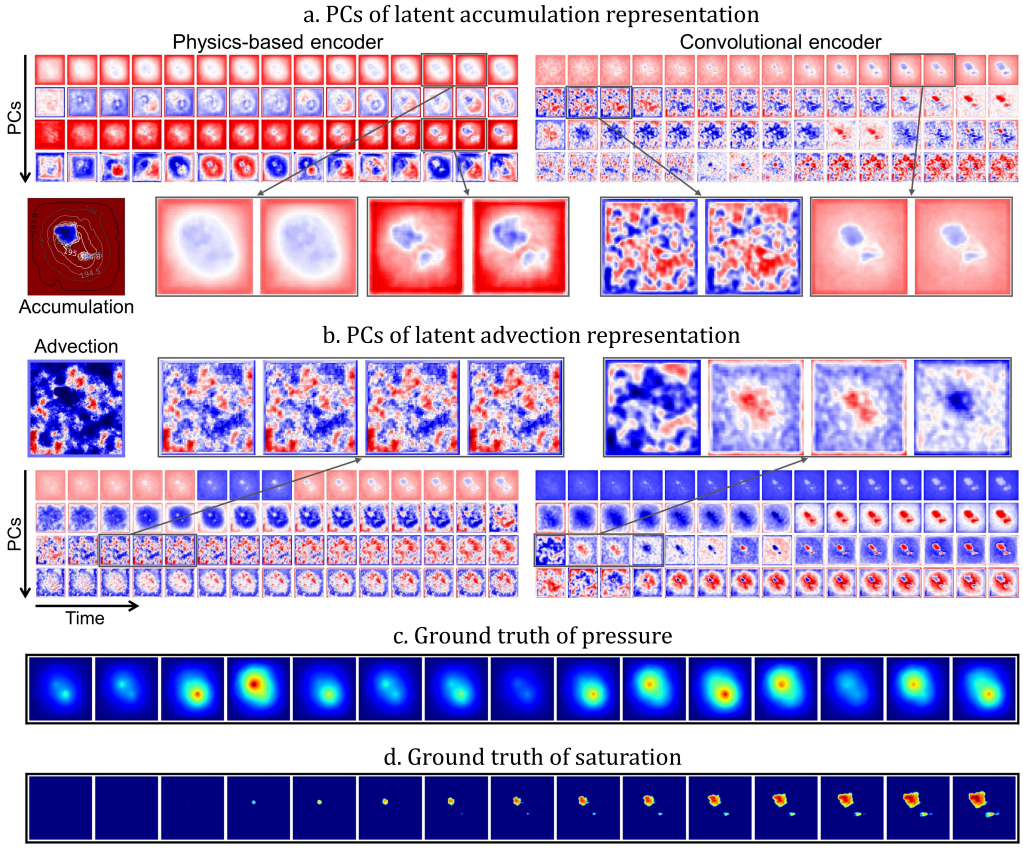

Figure 18: Visualization of the first four principal components of latent representations of (a) accumulation and (b) advection terms from two deep learning models with physics-based (left column) and convolutional encoders (right column), respectively. Each row in (a) and (b) represents the evolution of a principal component over the first 15 years. (c) and (d) visualize the ground truths of pressure and saturation of the first layer over the first 15 years. The true accumulation and z-direction advection of the wetting phase are provided in (a) and (b) for reference.

Tables

Acknowledgement

The authors acknowledge the support provided by Energi Simulation through an Industrial Research Chair on Subsurface Energy Data Science at USC. Upon publication of this study, a link to the data, codes, and examples used in this study will be posted to the FAIR-compliant Zenodo online repository as well as our research website at http://sees.usc.edu.

Author Contribution

Zhen Qin: Conceptualization, Methodology, Software, Data curation, Investigation, Validation, Visualization, Writing - Original draft preparation. **Yingxiang Liu**: Conceptualization, Data curation, Investigation, Validation. **Fangning Zheng**: Software, Data curation, Visualization, Writing- Original draft preparation. **Behnam Jafarpour**: Conceptualization, Resources, Supervision, Funding acquisition, Writing - Review & Editing.

Appendix A Governing Equations

We follow the work in (Voskov, 2017) to derive governing equations and implement the finite-volume discretization for the mass balance equation. For a system with n_c components and n_p phases, the transport equations can be written as follows:

$$\frac{\partial}{\partial t}\left(\phi\sum_{\xi=1}^{n_p} x_{i,\xi}\rho_\xi S_\xi\right) + \nabla\cdot\sum_{\xi=1}^{n_p}(x_{i,\xi}\rho_\xi \mathbf{v}_\xi) = \sum_{\xi=1}^{n_p} x_{i,\xi}\rho_\xi\tilde{q}_\xi, \quad i = 1, ..., n_c, \tag{A1}$$

where ϕ is porosity; $x_{i,\xi}$ is the mole fraction of component i in phase ξ; ρ_ξ is the density of phase ξ; S_ξ is the saturation of phase ξ; $\mathbf{v}_\xi$ is the volumetric flux vector (or Darcy flux) for phase ξ; $\tilde{q}_\xi$ is the external sources or sinks of volumetric rate per unit volume in phase ξ.

For a geologic CO_2 storage site where CO_2 is injected into the deep saline aquifer, the phases $\xi \in \{w, n\}$ could be w for the wetting phase (brine) and n for the non-wetting phase (supercritical CO_2). The multiphase extension of Darcy's equation is applied to describe the flow of each phase and is written as:

$$\mathbf{v}_\xi = -\mathbf{K}\frac{k_{r,\xi}}{\mu_\xi}(\nabla p_\xi - \rho_\xi g\nabla d) = -\lambda_\xi\mathbf{K}(\nabla p_\xi - \rho_\xi g\nabla d), \quad \xi \in \{w, n\}, \tag{A2}$$

where $\mathbf{K}$ is the permeability tensor, $k_{r,\xi}$ is the relative permeability, μ_ξ is the phase viscosity, $\lambda_\xi = k_{r,\xi}/\mu_\xi$ is the phase mobility, p_ξ is the phase pressure, g is the gravitational acceleration, and d represents the depth.

The capillary pressure is defined as the difference between the pressure of two phases and is a function of saturation:

$$p_c(s_w) = p_n - p_w, \tag{A3}$$

where p_n and p_w are the pressures of non-wetting (n) and wetting phases (w).

By applying the finite-volume discretization and backward Euler approximation in time, the mass balance equation can be discretized as:

$$\begin{aligned} &V\left((\phi\sum_{\xi=1}^{n_p} x_{i,\xi}\rho_\xi S_\xi)^{(n+1)} - (\phi\sum_{\xi=1}^{n_p} x_{i,\xi}\rho_\xi S_\xi)^{(n)}\right) \\ &-\Delta t\sum_{l\in L}\left(\sum_{\xi=1}^{n_p} x^l_{i,\xi}\rho^l_\xi T^l_\xi \Delta\psi^l_\xi\right) + \Delta t\sum_{\xi=1}^{n_p} x_{i,\xi}\rho_\xi q_\xi = 0, \end{aligned} \tag{A4}$$

where V is the volume of the grid cell; T^l_ξ is the phase transmissibility between two grid cells connected by the interface l; $\Delta\psi^l_\xi$ is the phase potential difference across the interface l considering pressure, capillary pressure, and gravity; q_ξ is the volumetric rate of phase ξ.

In this work, we simplify the CO_2-brine system to an immiscible two-fluid-phase system with no internal component gradient. Then the mass balance equation reduces to the phase-base balance equation, written as:

$$\frac{\partial}{\partial t}(\phi S_\xi\rho_\xi) + \nabla\cdot(\rho_\xi\mathbf{v}_\xi) = \rho_\xi\tilde{q}_\xi, \quad \xi \in \{w, n\}. \tag{A5}$$

The discretized mass balance equation in Eq. A4 is then transformed into:

$$V\left((\phi\rho_\xi S_\xi) - (\phi\rho_\xi S_\xi)^t\right) - \Delta t\sum_{l\in L}\rho^l_\xi T^l_\xi\Delta\psi^l_\xi = \Delta t\rho_\xi q_\xi. \tag{A6}$$

For the two grid cells u and v with the interface of l, T_ξ^l is defined as follows:

$$T_\xi^l = \frac{\bar{k}_{u,v} A_l}{L_{u,v}} \lambda_\xi = T_m \lambda_\xi, \tag{A7}$$

where $\bar{k}_{u,v}$ is the harmonic average of permeability between the grid cells u and v; A_l is the area of interface l; $L_{u,v}$ is the distance between the grid cells u and v.

The porosity ϕ in can be represented as $\phi_0(1 + c_r(p - p_{ref}))$. By denoting $1 + c_r(p - p_{ref})$ as c_ϕ, Eq. A6 can be written as:

$$\big((c_\phi \rho_\xi S_\xi) - (c_\phi \rho_\xi S_\xi)^t\big) - \frac{\Delta t}{V\phi_0} \sum_{l \in L} \rho_\xi^l \lambda_\xi^l T_m^l \Delta\psi_\xi^l = \frac{\Delta t}{V\phi_0} \rho_\xi q_\xi. \tag{A8}$$

By presenting the Eq. (A8) in algebraic form and the definition in Eqs. (4 - 9), Eq. (A8) can be rewritten as follows:

$$\begin{aligned} &\boldsymbol{r}_\xi(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{u}^t) = \\ &\big(\boldsymbol{z}_{acc,\xi}(\boldsymbol{x}) - \boldsymbol{z}_{acc,\xi}(\boldsymbol{x}^t)\big) - \boldsymbol{z}_{adv,\xi}(\boldsymbol{x}, \boldsymbol{m}) + \boldsymbol{z}_{src,\xi}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{0}. \end{aligned} \tag{A9}$$

Appendix B Implementation Details

B1 Description of Decoder and Control Encoder

At each time step t, the control encoder takes the control variable $\boldsymbol{u}^t$ as input and generates the latent variable $\boldsymbol{z}^t_{src}$ as output. The control encoder consists of two 3D convolutional (Conv3D) layers with linear activation functions and a kernel size of 3 on x-, y-, and z-directions. The output from Conv3D layers is downsampled by reducing the spatial dimension by a factor of 2 and projected to the latent space with the feature dimension of d_l.

At each time step t, the decoder takes the latent variables $\boldsymbol{z}^t_{acc}$ and $\boldsymbol{z}^t_{adv}$ as well as the static variable $\boldsymbol{m}$ as inputs and generates the dynamic states $\boldsymbol{S}^t_n$ and $\boldsymbol{p}^t$ as outputs (Figure B1). First, the latent variables from the processor layer are projected back to the original dimension D through the upsampling layers. Second, the inputs to the decoder are projected to the space with the dimension of d_{d_1} for each grid block. The projected features are then fused together to be the variable $\boldsymbol{z}^t$ through addition and concatenation. The role of the input $\boldsymbol{m}$ here is to remove the effect of static variables, $\boldsymbol{T}_m$ and $\boldsymbol{I}_{PV}$, that are coupled in the latent variable $\boldsymbol{z}^t_{adv}$. Then, the dynamic variable $\boldsymbol{x}^t = \{\boldsymbol{p}^t, \boldsymbol{S}^t_n\}$ is learned from the latent variable $\boldsymbol{z}^t$ through two Conv3D layers with the dimension of output channels to be d_{d_2} and d_{d_3}, respectively.

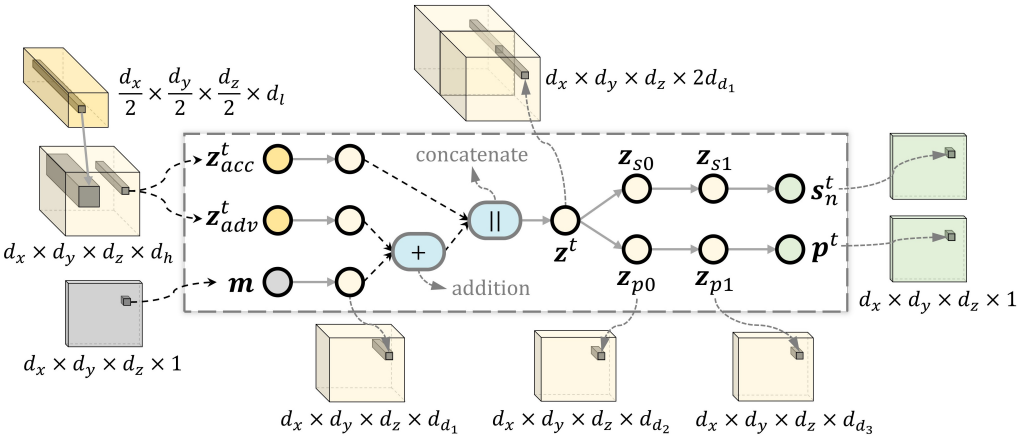

Figure B1: Illustration of the architecture of the decoder.

B2 Hyperparameters of Deep Learning Models

The hyperparameters of deep learning models, including our proposed model and the modified Recurrent R-U-Net are summarized in Tables B1 and B2. Both two models consist of approximately 3 million trainable parameters.

Table B1: Hyperparameters of each module of the proposed model: Encoder, Processor, and Decoder. The third column denotes the selected values of hyperparameters. The last column denotes the feasible range of the hyperparameters.

Module	Hyperparameters	Value	Range
Encoder	Physical Operator Dimension, d_o	40	{10, 20, 30, 40}
	Hidden Dimension, d_h	16	{8, 12, 16, 20}
	Latent Dimension, d_l	32	{24, 32, 48}
Processor	Dimension of Residual Feature, d_r	48	{32, 48, 64}
	Number of Residual Layers, N_{rl}	3	{2, 3, 4}
Decoder	Dimension of the 1st Layer, d_{d_1}	16	{8, 16, 32}
	Dimension of the 2nd Layer, d_{d_2}	8	{4, 8, 12, 16}
	Dimension of the 3rd Layer, d_{d_3}	4	{4, 6, 8}

Table B2: Hyperparameters of modified Recurrent R-U-Net.

Hyperparameter	**Selected Size**
Channels	[16, 32, 64]
# of Groups for GroupNorm Layer	4

B3 Configuration of Training

The training in this study employs the Adam optimizer as the chosen optimization algorithm, with the decay of the learning rate implemented gradually. Specifically, the learning rate decreases progressively over every step size of training samples by being multiplied by the decay factor. Further details about the optimization hyperparameters can be found in Table B3. The number of samples used for training, validation, and test sets for all models and experiments are tabulated in Table B4. All experiments are implemented using PyTorch (Paszke et al., 2019) and conducted on a single NVIDIA A100 40GB GPU. Summaries of model efficiency comparison and the input configuration are presented in Tables B5 and B6, respectively. As shown in the input configuration, the first dimension of three variables denotes the batch size of 1. The second and third dimensions of the control variable are the number of time steps of the prediction horizon during training and the number of components (water and CO_2), respectively. The second dimension of the dynamic variable denotes the number of dynamic variables including pressure and saturation. The second dimension of static variable represents the number of static variables, including the inverse of grid volume, depth, and transmissibility of six faces over x-, y-, and z-directions for a structured grid system. The dimension $128 \times 128 \times 20$ represents the grid sizes of the reservoir over three directions.

Table B3: Hyperparameters for the training of our model and RUNET in all experiments.

Hyperparameter	Value
Batch Size	2
Learning Rate	1e-4
Step Size	4500
Decay Factor	0.9
Gradient Clipping	40

Table B4: The numbers of samples used for training, validating, and testing all models in three experiments.

Experiment	Training	Validate	Test
Exp 1 (Public Dataset)	200, 500, 1000, 1500, 2000, 2500	50	373
Exp 1 (Our Dataset)	700	100	200
Exp 2	700	100	200

Table B5: Comparison of model efficiency. ConvEnc refers to the model sharing the same architecture as the proposed DL model except the encoder is the convolutional encoder.

Model	Parameter (Million)	Runtime (s/iter)
Ours	3.3951	0.2238
RUNET	3.0371	0.4201
ConvEnc	3.5225	0.4012

Table B6: Input size used for the comparison of model efficiency.

Input Variable	Input Size
Control Variable	$1 \times 8 \times 2 \times 128 \times 128 \times 20$
Dynamic Variable	$1 \times 2 \times 128 \times 128 \times 20$
Static Variable	$1 \times 8 \times 128 \times 128 \times 20$

Appendix C Numerical Simulation Model and Data Generation

In this section, we describe the numerical simulation model used to generate the training and test data for the described physics-encoded DL model for predicting the spatial-temporal evolution of the pressure and saturation for subsurface CO_2 storage. Figure C1 shows a synthetic 3D two-phase flow simulation model constructed using CMG-GEM (Computer Modeling Group (CMG), 2019). The storage reservoir region is encompassed by an aquifer region used to allow pressure dissipation and minimize the effects of no-flow boundaries. The zero-flux boundary conditions are imposed on the reservoir boundaries. The storage reservoir is at a depth of 1000 meters with a vertical grid dimension of 20 grid blocks $\times$ 2.5 meters and a horizontal grid dimension of 120 grid blocks $\times$ 50 meters, resulting in a total grid number of 288,000. The aquifer region is situated at the same depth and exhibits identical vertical and horizontal resolutions for the middle section as the storage reservoir. The two sides of the horizontal resolutions of the aquifer region are coarsened to 4 grid blocks $\times$ 500 meters to reduce computational cost. The total grid number of the entire simulation model is 327,680. The reservoir region has heterogeneous permeability values ranging from 0.1 to 2000 mD. Table C1 summarizes the simulation model settings. The injection spans 15 years, followed by a monitoring period of 85 years without CO_2 injection.

In this paper, we assume an isothermal environment for the CO_2 injection and storage processes. Various trapping mechanisms of CO_2 storage, including geologic, residual, solubility, and mineral trapping (Blunt, 2010), are considered in the simulation model. We use the Brooks-Corey relative permeability model (Brooks, 1965) with $n = 2$ to calculate the CO_2-water relative permeability. The maximum residual gas saturation (S_{grmax}) is set to be 0.4. The capillary pressure for the sand-CO_2-brine system is referred to Plug and Bruining (2007) of the drainage capillary pressure curve for supercritical CO_2 at 40 °C. The gas density is calculated with the Peng-Robinson Equation of State (Peng & Robinson, 1976). The gas viscosity is calculated from the Jossi, Stiel and Thodos correlation (Poling et al., 2001). The aqueous phase density and viscosity are calculated from the Rowe and Chou correlation (Rowe Jr & Chou, 1970) and the Kestin correlation (Kestin et al., 1981), respectively.

Uncertainties in the geologic properties of storage reservoirs, particularly in CO_2 sequestration sites like saline aquifers, arise due to limited understanding and measurement of geologic representation at various scales (Ma, 2011). These uncertainties stem from factors such as reservoir heterogeneity, anisotropy, and lateral variation resulting from uncertain rock composition, texture, pore structure, and rock type, which can influence the stress state and strength differences within the rocks (Middleton et al., 2012). Consequently, geomechanical properties become uncertain, impacting the capacity and costs of CO_2 storage in carbon capture and storage systems during pre-injection, injection, and post-injection periods (Anderson, 2017). To enhance the robustness of the developed deep learning model and effectively address geologic uncertainty, we employ sampling techniques and ensemble representations. These techniques allow us to incorporate varying levels of heterogeneity range and generate multiple realizations of the permeability map. We utilize sequential Gaussian simulation with the Stanford Geostatistical Modeling Software (Remy et al., 2009) for this purpose. The geometric anisotropy is characterized by a spherical variogram model where the maximum (x-direction) and medium (y-direction) ellipsoid ranges are set isotropically with three distinct scenarios of 20, 60, and 100 units. The minimum ellipsoid range is represented by two distinct scenarios, namely 2 and 5 units. Consequently, a comprehensive set of six scenarios, each featuring different ellipsoid ranges, is utilized to generate a total of 100 realizations per scenario, thereby resulting in a total of 600 realizations of the 3D permeability maps. Figure C2 shows the permeability maps of the first four realizations from each of the six distinct scenarios.

Given each realization of the permeability map, the simulation data is generated based on various cases of well injection schedules. The well location is fixed at 64,64,17 for the one-well scenario and the well indexes of the two-well scenario are fixed at {55,55,17} and {75,75,17}. The total CO_2 injection volume, $V^{inj}_{T_{CO_2}}$, over 15 years of injection is fixed to be 3.5e+09 m^3 ($\sim$ 10 $mtCO_2$) for both one-well and two-well scenarios. To ensure comprehensive coverage of the realistic response space, we divide the injection duration into three intervals, each spanning 5 years. We assign each interval with an injection volume, $V_{interval_x}$, with one of the V_{high}, V_{med}, or V_{low}, where $V_{high} = 0.5V^{inj}_{T_{CO_2}}$, $V_{med} = 0.35V^{inj}_{T_{CO_2}}$, and $V_{low} = 0.15V^{inj}_{T_{CO_2}}$. We then create ten different combinations of those injection volumes as shown in Table C2 and Figure C3 (a). The injection rate, u (m^3/day), changes every year, and $U = \{u^{t_1}, u^{t_2}, \ldots, u^{t_k}\}$ where $k = 15$. We randomly assign a portion of the injection volume to each year based on its corresponding interval. Let

$$p = rand(1, 5) = [p_1, p_2, p_3, p_4, p_5], \tag{C1}$$

and injection rate u at time t_i is calculated by

$$u^{t_i} = \frac{p_i}{sum(p)} \frac{V_{interval_x}}{365}, \tag{C2}$$

where p_i corresponds to the same injection interval x where u^{t_i} belongs to.

Figure C3 (b) shows an example of random injection rate allocation based on total injection allocation in Figure C3 (a). In this way, a total of 6000 simulation runs are performed and the spatial and temporal results of pressure and gas saturation are generated. The described data generation approach maximizes our ability to explore a wide range of injection rate combinations.

For the two-well scenario, the injection rate is distributed into two wells with specific portions:

$$[u^{t_i}_{w_1}, u^{t_i}_{w_2}] = u^{t_i}[q_i, 1 - q_i], \tag{C3}$$

where q_i is the percentage of injection amount assigned to well 1, $(1-q_i)$ is the percentage of injection amount assigned to well 2, and $q = [0.1 : 0.1 : 1]$ for a total of 10 injection schedules for each permeability realization.

Table C1: Numerical simulation model settings.

	Reservoir Region	Aquifer Region
Grid numbers $\times$ Grid size (meter) in x and y	30×100	$4 \times 500 + 30 \times 100 + 4 \times 500$
Grid numbers $\times$ Grid size (meter) in z	20×2.5	20×2.5
Permeability (mD)	$0.1 - 2000$	200
Porosity	0.18	0.18
Pore pressure gradient (kPa/m)	9.8	9.8
k_v/k_h	0.1	0.1
Temperature ($^\circ C$) (isothermal condition)	40	40

Table C2: Ten combinations of the percentage injection volume for each interval. V_{high}, V_{med}, V_{low}, and V_{avg} correspond to 50%, 35%, 15%, and 33.33% of the total injection volume, respectively.

	$V_{interval_1}$	$V_{interval_2}$	$V_{interval_3}$
Combination 1	V_{high}	V_{med}	V_{low}
Combination 2	V_{high}	V_{low}	V_{med}
Combination 3	V_{med}	V_{high}	V_{low}
Combination 4	V_{med}	V_{low}	V_{high}
Combination 5	V_{low}	V_{high}	V_{med}
Combination 6	V_{low}	V_{med}	V_{high}
Combination 7	V_{avg}	V_{avg}	V_{avg}
Combination 8	V_{high}	0	V_{high}
Combination 9	V_{high}	V_{high}	0
Combination 10	0	V_{high}	V_{high}

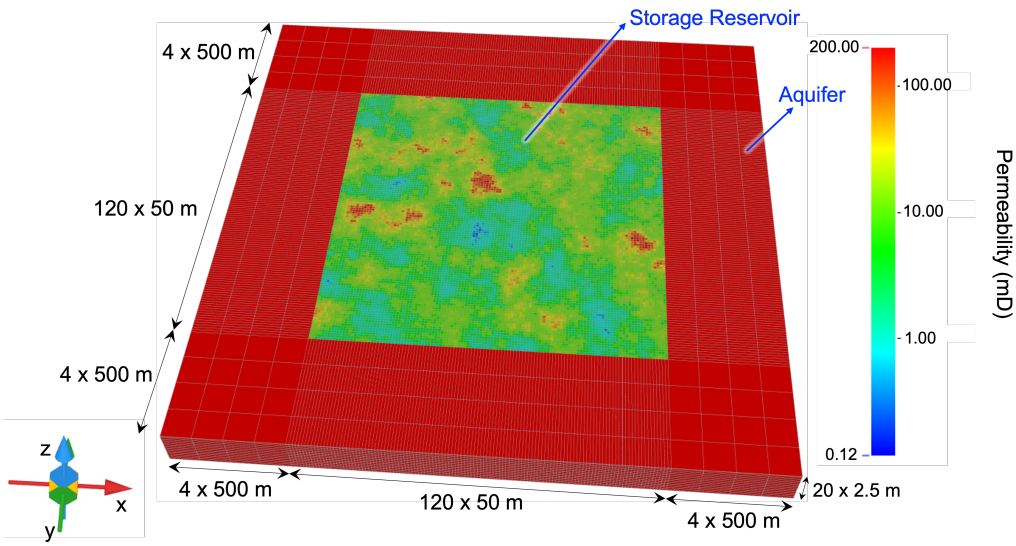

Figure C1: Numerical simulation model of 3D deep saline aquifer reservoir.

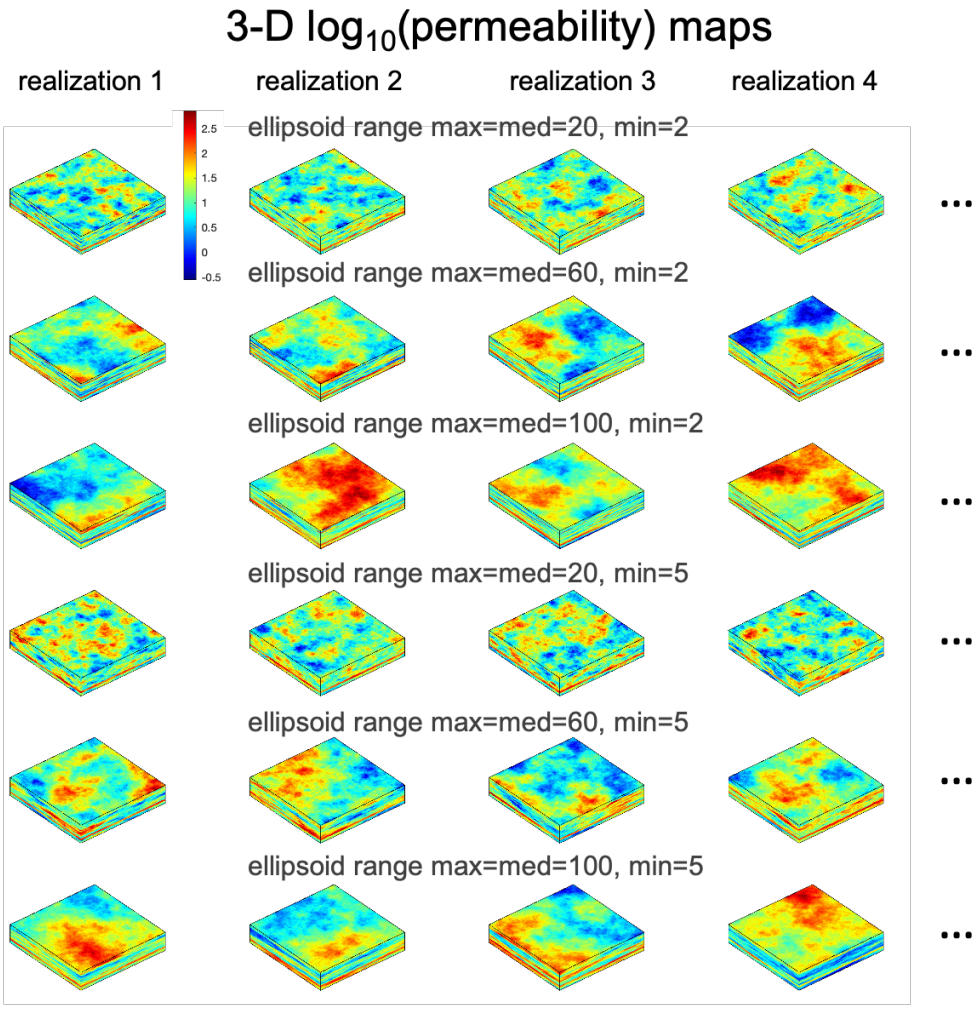

Figure C2: The 3D Permeability maps of the first four realizations from each of the six scenarios with distinct ellipsoid ranges.

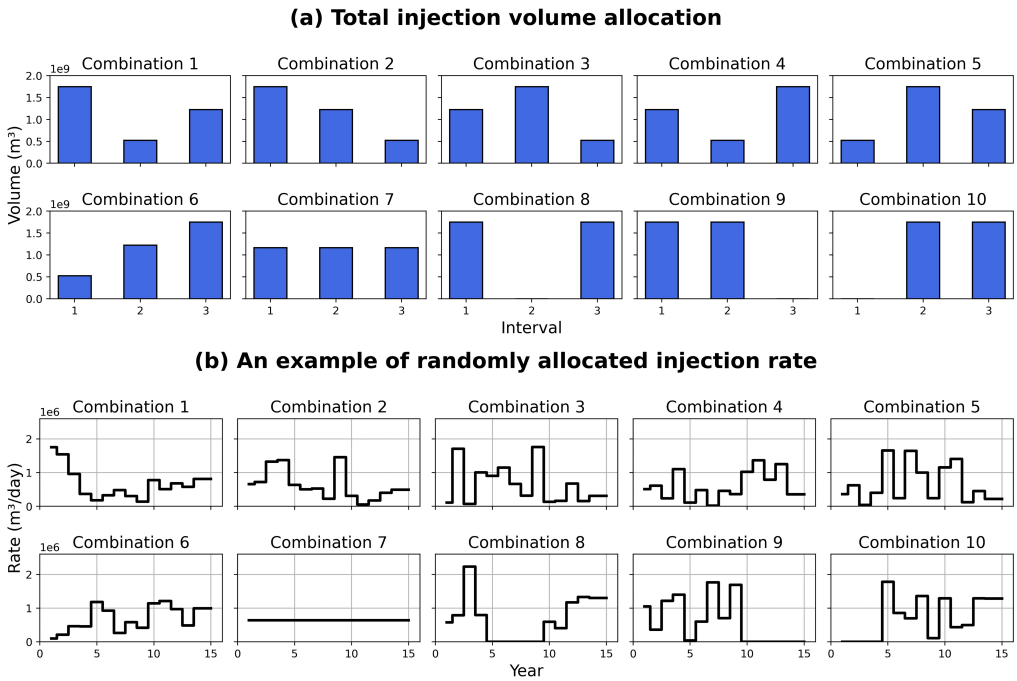

Figure C3: Ten combinations of the percentage injection volume for each interval. (a) total injection volume allocation for each interval. (b) an example of random injection rate allocation based on total injection allocation on (a).

References

Ajayi, T., Gomes, J. S., & Bera, A. (2019). A review of CO_2 storage in geological formations emphasizing modeling, monitoring and capacity estimation approaches. *Petroleum Science*, *16*(5), 1028–1063. doi: 10.1007/s12182-019-0340-8

Anderson, S. T. (2017). Cost implications of uncertainty in CO_2 storage resource estimates: A review. *Natural Resources Research*, *26*, 137–159.

Bandilla, K. W., Celia, M. A., Birkholzer, J. T., Cihan, A., & Leister, E. C. (2015). Multiphase modeling of geologic carbon sequestration in saline aquifers. *Groundwater*, *53*, 362–377. doi: 10.1111/gwat.12315

Blunt, M. (2010). Carbon dioxide storage. *Grantham Institute Briefing Paper*, *4*.

Brooks, R. H. (1965). *Hydraulic properties of porous media.* Colorado State University.

Bui, M., Adjiman, C. S., Bardow, A., Anthony, E. J., Boston, A., Brown, S., ... Mac Dowell, N. (2018). Carbon capture and storage (CCS): the way forward. *Energy & Environmental Science*, *11*, 1062–1176. doi: 10.1039/C7EE02342A

Celia, M. A., Bachu, S., Nordbotten, J. M., & Bandilla, K. W. (2015). Status of CO_2 storage in deep saline aquifers with emphasis on modeling approaches and practical simulations. *Water Resources Research*, *51*, 6846–6892. doi: 10.1002/2015WR017609

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems* (Vol. 31, pp. 6571–6583). Curran Associates, Inc.

Computer Modeling Group (CMG). (2019). *Gem user manual.*

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, *92*(3), 88.

Dulny, A., Hotho, A., & Krause, A. (2022). NeuralPDE: Modelling Dynamical Systems from Data. In *KI 2022: Advances in Artificial Intelligence* (pp. 75–89). Springer International Publishing.

E, W. (2017). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, *1*(5), 1–11. doi: 10.1007/s40304-017-0103-z

E, W., Han, J., & Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, *5*(4), 349–380. doi: 10.1007/s40304-017-0117-6

Faroughi, S. A., Pawar, N., Fernandes, C., Raissi, M., Das, S., Kalantari, N. K., & Mahjour, S. K. (2023). *Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks in Scientific Computing.* arXiv. (arXiv:2211.07377 [cs]) doi: 10.48550/arXiv.2211.07377

Fuks, O., & Tchelepi, H. A. (2020). Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, *1*(1), 19–37. doi: 10.1615/JMachLearnModelComput.2020033905

Gorecki, C. D., Ayash, S. C., Liu, G., Braunberger, J. R., & Dotzenrod, N. W. (2015). A comparison of volumetric and dynamic CO_2 storage resource and efficiency in deep saline formations. *International Journal of Greenhouse Gas Control*, *42*, 213–225. doi: 10.1016/j.ijggc.2015.07.018

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

Hendrycks, D., & Gimpel, K. (2023). *Gaussian error linear units (gelus).*

Ioffe, S., & Szegedy, C. (2015, 07–09 Jul). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the*

32nd international conference on machine learning (Vol. 37, pp. 448–456). Lille, France: PMLR.

J. Nagoor Kani, & Elsheikh, A. H. (2019). Reduced-order modeling of subsurface multi-phase flow models using deep residual recurrent neural networks. *Transport in Porous Media*, *126*(3), 713–741. doi: 10.1007/s11242-018-1170-7

Jiang, X. (2011). A review of physical modelling and numerical simulation of long-term geological storage of CO_2. *Applied Energy*, *88*(11), 3557–3566. doi: 10.1016/j.apenergy.2011.05.004

Jiang, Z., Tahmasebi, P., & Mao, Z. (2021). Deep residual u-net convolution neural networks with autoregressive strategy for fluid flow predictions in large-scale geosystems. *Advances in Water Resources*, *150*, 103878. doi: 10.1016/j.advwatres.2021.103878

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. doi: 10.1038/s42254-021-00314-5

Kestin, J., Khalifa, H. E., & Correia, R. J. (1981). Tables of the dynamic and kinematic viscosity of aqueous nacl solutions in the temperature range 20–150 c and the pressure range 0.1–35 mpa. *Journal of physical and chemical reference data*, *10*(1), 71–88.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., & Houlsby, N. (2020). Big transfer (BiT): General visual representation learning. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer Vision – ECCV 2020* (pp. 491–507). Cham: Springer International Publishing.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., & Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. In *Advances in neural information processing systems* (Vol. 34, pp. 26548–26560). Curran Associates, Inc.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. M., & Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. *CoRR*, *abs/2010.08895*. Retrieved from `https://arxiv.org/abs/2010.08895`

Long, Z., Lu, Y., & Dong, B. (2019). PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, *399*, 108925. doi: 10.1016/j.jcp.2019.108925

Long, Z., Lu, Y., Ma, X., & Dong, B. (2018). PDE-Net: Learning PDEs from Data. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 3208–3216). PMLR.

Lu, Y., Zhong, A., Li, Q., & Dong, B. (2018). Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 3276–3285). PMLR.

Ma, Y. Z. (2011). Uncertainty analysis in reservoir characterization and management: How much should we know about what we don't know? In *Uncertainty Analysis and Reservoir Modeling* (pp. 1–15). American Association of Petroleum Geologists. doi: 10.1306/13301404M963458

Middleton, R. S., Keating, G. N., Viswanathan, H. S., Stauffer, P. H., & Pawar, R. J. (2012). Effects of geologic reservoir uncertainty on CO_2 transport and storage infrastructure. *International Journal of Greenhouse Gas Control*, *8*, 132–142.

Mo, S., Zabaras, N., Shi, X., & Wu, J. (2019). Deep autoregressive neural networks for high-dimensional inverse problems in groundwater contaminant source identification. *Water Resources Research*, *55*(5), 3856–3881. doi: 10.1029/2018WR024638

Mo, S., Zhu, Y., Zabaras, N., Shi, X., & Wu, J. (2019). Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow

in heterogeneous media. *Water Resources Research*, *55*(1), 703–728. doi: 10.1029/2018WR023528

Muther, T., Dahaghi, A. K., Syed, F. I., & Van Pham, V. (2023). Physical laws meet machine intelligence: current developments and future directions. *Artificial Intelligence Review*, *56*(7), 6947–7013.

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 *Advances in neural information processing systems* (Vol. 32, pp. 8026–8037). Curran Associates, Inc.

Pawar, R. J., Bromhal, G. S., Carey, J. W., Foxall, W., Korre, A., Ringrose, P. S., ... White, J. A. (2015). Recent advances in risk assessment and risk management of geologic CO_2 storage. *International Journal of Greenhouse Gas Control*, *40*, 292–311. doi: 10.1016/j.ijggc.2015.06.014

Peng, D. Y., & Robinson, D. B. (1976). A new two-constant equation of state. *Industrial & Engineering Chemistry Fundamentals*, *15*(1), 59–64.

Plug, W. J., & Bruining, J. (2007). Capillary pressure for the sand–CO_2–water system under various pressure conditions. application to CO_2 sequestration. *Advances in Water Resources*, *30*(11), 2339–2353.

Poling, B. E., Prausnitz, J. M., & O'connell, J. P. (2001). *Properties of gases and liquids*. McGraw-Hill Education.

Qin, Z., Jiang, A., Faulder, D., Cladouhos, T. T., & Jafarpour, B. (2023). Efficient optimization of energy recovery from geothermal reservoirs with recurrent neural network predictive models. *Water Resources Research*, *59*(3), e2022WR032653. doi: https://doi.org/10.1029/2022WR032653

Qin, Z., Jiang, A., Faulder, D., Cladouhos, T. T., & Jafarpour, B. (2024). Physics-Guided Deep Learning for Prediction of Energy Production from Geothermal Reservoirs. *Geothermics*, *116*, 102824. doi: https://doi.org/10.1016/j.geothermics.2023.102824

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., ... Courville, A. (2019, 09–15 Jun). On the spectral bias of neural networks. In *Proceedings of the 36th international conference on machine learning* (Vol. 97, pp. 5301–5310). PMLR.

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

Rao, C., Sun, H., & Liu, Y. (2021). *Hard encoding of physics for learning spatiotemporal dynamics.*

Remy, N., Boucher, A., & Wu, J. (2009). *Applied geostatistics with sgems: A user's guide*. Cambridge University Press. Retrieved from `https://books.google.com/books?id=AFleglOOLacC`

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Medical image computing and computer-assisted intervention – miccai 2015* (pp. 234–241). Springer International Publishing.

Rowe Jr, A. M., & Chou, J. C. (1970). Pressure-volume-temperature-concentration relation of aqueous sodium chloride solutions. *Journal of Chemical and Engineering Data*, *15*(1), 61–66.

Ruthotto, L., & Haber, E. (2020). Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, *62*(3), 352–364. doi: 10.1007/s10851-019-00903-1

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., & Woo, W.-c. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 28, p. 802—

810). Curran Associates, Inc.

Shokouhi, P., Kumar, V., Prathipati, S., Hosseini, S. A., Giles, C. L., & Kifer, D. (2021). Physics-informed deep learning for prediction of CO_2 storage site response. *Journal of Contaminant Hydrology*, *241*, 103835. doi: 10.1016/j.jconhyd.2021.103835

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., ... Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in neural information processing systems* (Vol. 33, pp. 7537–7547). Curran Associates, Inc.

Tang, H., Fu, P., Sherman, C. S., Zhang, J., Ju, X., Hamon, F., ... Morris, J. P. (2021). A deep learning-accelerated data assimilation and forecasting workflow for commercial-scale geologic carbon storage. *International Journal of Greenhouse Gas Control*, *112*, 103488. doi: 10.1016/j.ijggc.2021.103488

Tang, M., Ju, X., & Durlofsky, L. J. (2022). Deep-learning-based coupled flow-geomechanics surrogate model for CO_2 sequestration. *International Journal of Greenhouse Gas Control*, *118*, 103692. doi: 10.1016/j.ijggc.2022.103692

Tang, M., Liu, Y., & Durlofsky, L. J. (2020). A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *Journal of Computational Physics*, *413*, 109456. doi: 10.1016/j.jcp.2020.109456

Tang, M., Liu, Y., & Durlofsky, L. J. (2021). Deep-learning-based surrogate flow modeling and geological parameterization for data assimilation in 3d subsurface flow. *Computer Methods in Applied Mechanics and Engineering*, *376*, 113636. doi: 10.1016/j.cma.2020.113636

Voskov, D. V. (2017). Operator-based linearization approach for modeling of multiphase multi-component flow in porous media. *Journal of Computational Physics*, *337*, 275–288. doi: 10.1016/j.jcp.2017.02.041

Wang, N., Zhang, D., Chang, H., & Li, H. (2020). Deep learning of subsurface flow via theory-guided neural network. *Journal of Hydrology*, *584*, 124700. doi: 10.1016/j.jhydrol.2020.124700

Wang, Y., & Lin, G. (2020). Efficient deep learning techniques for multiphase flow simulation in heterogeneous porous media. *Journal of Computational Physics*, *401*, 108968. doi: 10.1016/j.jcp.2019.108968

Wen, G., Hay, C., & Benson, S. M. (2021). CCSNet: A deep learning modeling suite for CO_2 storage. *Advances in Water Resources*, *155*, 104009. doi: 10.1016/j.advwatres.2021.104009

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., & Benson, S. M. (2022). U-FNO—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, *163*, 104180. doi: 10.1016/j.advwatres.2022.104180

Wen, G., Li, Z., Long, Q., Azizzadenesheli, K., Anandkumar, A., & Benson, S. M. (2023). Real-time high-resolution CO_2 geological storage prediction using nested fourier neural operators. *Energy & Environmental Science*, *16*(4), 1732–1741. doi: 10.1039/D2EE04204E

Wen, G., Tang, M., & Benson, S. M. (2021). Towards a predictor for CO_2 plume migration using deep neural networks. *International Journal of Greenhouse Gas Control*, *105*, 103223. doi: 10.1016/j.ijggc.2020.103223

Willard, J., Jia, X., Xu, S., Steinbach, M., & Kumar, V. (2022, November). Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems. *ACM Computing Surveys*, *55*(4), 66:1–66:37. doi: 10.1145/3514228

Wu, H., Hu, T., Luo, H., Wang, J., & Long, M. (2023). *Solving high-dimensional pdes with latent spectral models.*

Wu, Y., & He, K. (2018, September). Group normalization. In *Proceedings of the european conference on computer vision (eccv)* (pp. 3–19).

Xiong, W., Huang, X., Zhang, Z., Deng, R., Sun, P., & Tian, Y. (2023). *Koop-*

man neural operator as a mesh-free solver of non-linear partial differential equations.

Yan, B., Harp, D. R., Chen, B., Hoteit, H., & Pawar, R. J. (2022). A gradient-based deep neural network model for simulating multiphase flow in porous media. *Journal of Computational Physics*, *463*, 111277. doi: 10.1016/j.jcp.2022.111277

Yan, B., Harp, D. R., Chen, B., & Pawar, R. (2022). A physics-constrained deep learning model for simulating multiphase flow in 3d heterogeneous porous media. *Fuel*, *313*, 122693. doi: 10.1016/j.fuel.2021.122693

Zheng, F., Jahandideh, A., Jha, B., & Jafarpour, B. (2021). Geologic CO_2 storage optimization under geomechanical risk using coupled-physics models. *International Journal of Greenhouse Gas Control*, *110*, 103385. doi: 10.1016/j.ijggc.2021.103385

Zheng, F., Jha, B., & Jafarpour, B. (2022). Optimization of CO_2 storage and leakage through caprock fracturing using coupled flow-geomechanics-fracturing simulation. *ECMOR 2022*, *2022*(1), 1–16.

Zhu, Y., & Zabaras, N. (2018). Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, *366*, 415–447. doi: 10.1016/j.jcp.2018.04.018