Abstract

Predicting the spatial-temporal evolution of the pressure field and the migration of the CO_2 plume during and after its injection is complicated by several factors, including the complexity of the underlying flow and transport processes in porous rocks, the heterogeneity and uncertainty of the rock properties, and the computational demand of running high-fidelity simulation models. Traditional workflows, such as model calibration, optimization, uncertainty quantification, and risk assessment, require many numerical simulation runs, which can become computationally too demanding for field application. Although standard deep learning models provide powerful prediction tools for near real-time implementation, they suffer from several important limitations, including the lack of interpretability, extensive data needs, and physical inconsistency.

To overcome these limitations, we introduce a novel physics-encoded deep learning model, named Fluid Flow Deep Learning (FFDL) architecture, aimed at predicting the spatial-temporal evolution of pressure and saturation during geologic CO_2 storage. The proposed model leverages physical causality in the design of the architecture to approximately represent the subsurface fluid dynamics. The proposed model features a physics-informed encoder for generating physically interpretable latent variables, a customized recurrent neural network processor for evolving the latent state variables over time, and a decoder for mapping these variables to the desired outputs, such as pressure and saturation. The FFDL architecture uses physical operators that serve as non-linear activation functions and impose hard constraints to respect the structure of the fluid flow equations. A comprehensive investigation of FFDL based on several examples of CO_2 storage in saline aquifers is used to demonstrate the prediction performance of the approach. The performance of the model is validated by showing its superior prediction of the CO_2 plume migration compared to the results obtained by a modified recurrent U-Net architecture. The results show that FFDL possesses significantly improved generalizability and provides a more reliable and physically consistent prediction of the CO_2 plume migration.

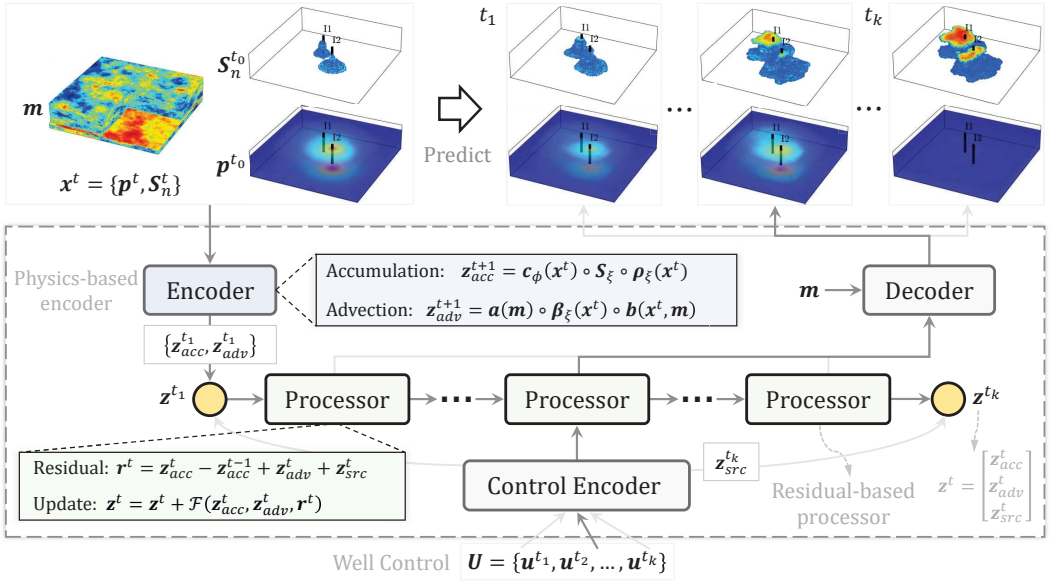

Figure 1 *Overview of the proposed Fluid Flow-based Deep Learning (FFDL) model for predicting spatial-temporal pressure and saturation in geologic* CO_2 *storage.*

Introduction

Carbon capture and storage (CCS) is an important component in reducing CO_2 emissions. The CCS technology works by capturing CO_2 from large point sources, such as power plants, and injecting it into deep geologic formations. Although geologic CO_2 storage (GCS) has significant potential as a climate change mitigation strategy, its commercial deployment requires sound economics and regulatory frameworks to ensure the safety of operation. Concerns about CO_2 migration or leakage into shallower aquifers and the atmosphere 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 predicting its migration path for monitoring and verification purposes remain challenging (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 assessing short-term and long-term risks (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 set of complex multi-component, multiphase flow and transport processes. Complex rock-fluids-fracture interactions involving miscibility, capillary pressure, relative permeability, and coupling between different physical processes lead to non-linear coupled systems of partial differential equations (PDEs) that are not trivial to solve (Bandilla et al., 2015). Furthermore, field-scale GCS projects span extensive spatial and temporal scales, including both injection and post-injection periods (Ajayi et al., 2019; Jiang, 2011). Therefore, numerical simulation of GCS at the field scale can become computationally prohibitive, particularly for complex optimization and uncertainty quantification tasks. 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 more efficient alternatives to traditional physics-based numerical simulation to predict the spatial-temporal evolution of fluid dynamics in the subsurface. Specifically, convolutional neural networks (CNNs) that have demonstrated a

strong capability of 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 approximately map input permeability fields to pressure and velocity maps for a 2-dimensional steady-state Darcy flow problem. Mo et al. (2019) extended the work in Zhu and Zabaras (2018) to predict the responses of a dynamic multiphase flow problem at different time steps. Wang and Lin (2020) designed a customized architecture for single- and two-phase flow systems, incorporating sparsely connected layers to account for the inherent sparse input-output interaction. 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 of injection and potentially much longer post-injection period, the predictive model needs to be versatile and provide forecasts over arbitrary time frames. However, existing DL models are limited to predict within fixed periods and face challenges in long-term predictions (which require extrapolation 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 by Mo 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 as they require predictions over long-term horizons. Another important feature is the flexibility of the model in terms of providing predictions in different development scenarios. For example, models should be able to capture the response of storage formations to time-varying controls, which is important in optimizing field performance by dynamically managing injection strategies based on in situ reservoir conditions and monitoring data (Pawar et al., 2015).

A domain-aware 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). In contrast to physics-informed approaches that serve as soft constraints, physics-encoded architectures impose hard constraints on the learning process. By capturing the underlying physical dependencies among variables, these architectures demonstrate their connections to Ordinary Differential Equations (ODEs) (E et al., 2017; Lu et al., 2018; Qin et al., 2024) and PDEs (Long et al., 2018; Rao et al., 2021). Long et al. (2018) proposed PDE-Net to learn PDEs from data. In their work, CNN is combined with Residual Network (ResNet) (He et al., 2016) to approximate the evolution of PDEs 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 to approximate the temporal evolution of the system, resulting in improved 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. However, 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 for geologic CO_2 storage. The proposed approach incorporates the general form of the flow equations, emulating the flux, accumulation, and source/sink terms, without hard-coding the detailed form of the equations. Instead, by providing learnable parameters, the model is trained to estimate these flow-related terms. 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 previous works (Long et al., 2018; Rao et al., 2021; Dulny et al., 2022), we employ convolutional layers to capture 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. The FFDL model is designed to handle time-varying well controls and to provide long-term predictions. We present a modified Recurrent R-U-Net to allow for time-varying control and arbitrary time steps, based on the work in Tang et al. (2022), as a baseline model due to its similar architecture and capability. The predictive performance of FFDL is investigated using a field-scale model of GCS in a saline aquifer. Our results show that by incorporating the general form of the flow equations FFDL is able to outperform the Recurrent R-U-Net on test sets featuring unseen permeability, well controls, and time frames.

Methodology

Problem Statement

This study addresses a spatial-temporal prediction task governed by a set of coupled PDEs within a 3D storage reservoir composed of grid blocks $\mathscr{D} \subset \mathbb{R}^3$. The prediction task can be formulated as $\mathbf{X} = \mathscr{F}(\boldsymbol{x}^t, \boldsymbol{m}, \mathbf{U})$, 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 $\mathscr{F}$ denotes the mapping from inputs to outputs. For each coordinate $\boldsymbol{\omega} \in \mathscr{D}$, 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 parameter $\boldsymbol{m}(\boldsymbol{\omega}) \in \mathbb{R}^{d_m}$ characterizes the model parameters, such as permeability and porosity 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. The notation $\boldsymbol{u}^t(\boldsymbol{\omega}) \in \mathbb{R}^{d_u}$ 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 of coordinate $\boldsymbol{\omega}$, respectively. The goal of this work is to approximate the operator $\mathscr{F}$ with the proposed deep learning model $\mathscr{F}_\theta$, where θ represents the trainable parameters.

In this approach, we convert the dynamic variables, pressure and saturation, into physically meaningful latent variables $\boldsymbol{z}^t = \{\boldsymbol{z}_{acc}^t, \boldsymbol{z}_{adv}^t, \boldsymbol{z}_{src}^t\}$ to represent 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, $\boldsymbol{z}_{acc}^{t+1}$ and $\boldsymbol{z}_{adv}^{t+1}$, using $\boldsymbol{x}^t$ and $\boldsymbol{m}$ as inputs. Concurrently, the control encoder produces the latent variables $\boldsymbol{z}_{src}^{t+1}$ given the control variables $\boldsymbol{u}^{t+1}$. The processor receives three latent variables as input and generates the updated latent variables $\boldsymbol{z}_{acc}^{t+1}$ and $\boldsymbol{z}_{adv}^{t+1}$ as outputs. These outputs are then sent to: 1) the decoder for the prediction of the dynamic variables $\boldsymbol{x}^{t+1}$; and 2) the processor itself for the estimation of the new latent variables $\boldsymbol{z}_{acc}^{t+2}$ and $\boldsymbol{z}_{adv}^{t+2}$. The decoder takes the updated latent variables $\boldsymbol{z}_{acc}^{t+1}$ and $\boldsymbol{z}_{adv}^{t+1}$, 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. The processor utilizes the previous latent variables as initial estimates and recurrently updates them for the next step by integrating $\boldsymbol{z}_{src}^{t+1}$ as external inputs. To articulate the model's function, the process $\mathbf{X} = \mathscr{F}(\boldsymbol{x}^t, \boldsymbol{m}, \mathbf{U})$ can be modularized as follows:

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

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

Physical Operators

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. Inspired by the Operator-Based Linearization (OBL) approach (Voskov, 2017), we adopt the concept of physical operators to map the input variables to physically

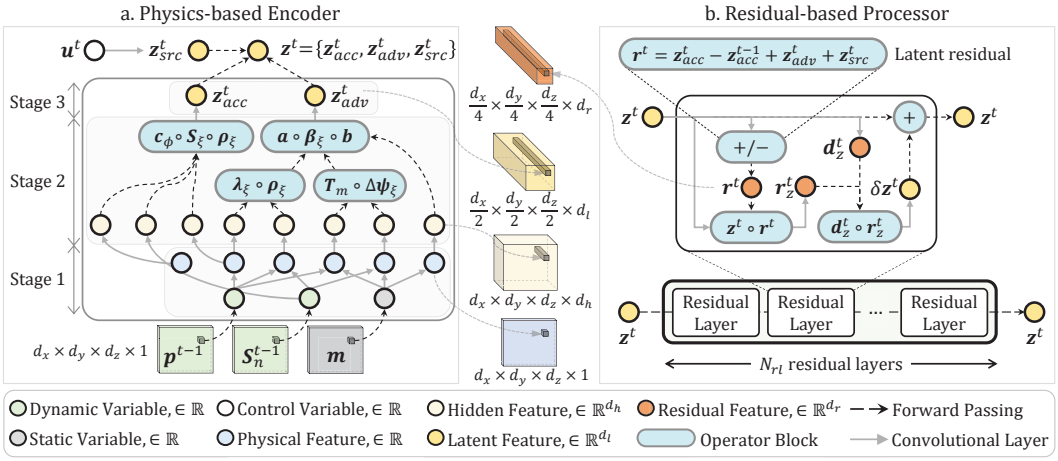

Figure 2 *Overview of the physics-based encoder (left) and the residual-based processor (right). Each circle represents a variable corresponding to a single voxel in the 3D representation of the reservoir.*

meaningful latent variables that represent accumulation and advection terms. 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 Eq. 2 as a combination of operators in an algebraic form for the whole reservoir in three-dimensional space $\mathscr{D}$:

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

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, which are defined as follows:

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

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

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

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

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

$$\boldsymbol{b}_l(\boldsymbol{x}, \boldsymbol{m}) = \boldsymbol{T}_m^l \circ \Delta \boldsymbol{\psi}_\xi^l = \boldsymbol{T}_m^l \circ \left(\boldsymbol{\psi}_\xi^v - \boldsymbol{\psi}_\xi^u\right), \tag{9}$$

where the symbol $\circ$ represents the Hadamard (or element-wise) product; $\boldsymbol{c}_\phi$ 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 $\{\boldsymbol{T}_m, \boldsymbol{I}_{PV}, \boldsymbol{D}\}$.

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:

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

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 two-layer fully-connected (FC) NNs with the hidden unit d_o, named state-related operator. Features $\Delta\boldsymbol{p}$ and $\Delta(\boldsymbol{\rho}_\xi \circ \boldsymbol{D})$ are the differential pressure and gravity terms used in the potential difference $\Delta\boldsymbol{\psi}$, calculated through a non-parametric spatial-related operator. 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 $\Delta(\boldsymbol{\rho}_\xi \circ \boldsymbol{D})$ are computed along the x-, y-, and z-directions. The differential operator is implemented by applying the discretization stencil $0.5 \times [-1, 0, 1]$ along each of the three directions.

In the second stage, convolutional layers are used to project dynamic variables $\boldsymbol{x}$, input parameters $\boldsymbol{m}$, and physical features $\boldsymbol{f}$ into high-dimensional hidden features $\boldsymbol{h}$. Each hidden feature has a dimensionality of d_h at the point $\boldsymbol{\omega}$. The hidden features are then passed to the operator blocks defined in Eqs. (4 - 9), which serves as hard constraints and returns the high-dimensional representations of the accumulation and advection terms. In the third stage, we further map the accumulation and advection terms into a high-dimensional latent space with dimension d_l ($d_l > d_h$) to output the latent variables $\boldsymbol{z}^t_{acc}$ and $\boldsymbol{z}^t_{adv}$. The latent variables are simultaneously downsampled through the convolutional layers, reducing the spatial dimension by a factor of 2. The downsampling is employed to increase the receptive field of the convolutional layers while reducing the GPU memory demand.

The latent variables will be further updated by the processor, taking into account the external effect of the control variable. The encoder $\mathscr{F}_{\theta_{\text{InputToLatent}}}$ can then be decomposed into three stages as follows:

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

where $\mathscr{F}_{\theta_{\text{InputToFeature}}}$, $\mathscr{F}_{\theta_{\text{FeatureToHidden}}}$, and $\mathscr{F}_{\theta_{\text{HiddenToLatent}}}$ refer to the three stages of the encoder, respectively.

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 processor unit in this study is called a residual-based processor, as it is composed of residual layers and draws inspiration from the concept of residuals in the governing equations. 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 $\mathscr{F}_{\theta_{\text{ControlToLatent}}}$. Each customized recurrent unit, named processor block, consists of N_{rl} residual layers to update the latent variables. To make the model more memory-efficient, we further reduce the spatial dimension of the latent variables $\boldsymbol{z}^t_{acc}$, $\boldsymbol{z}^t_{adv}$, and $\boldsymbol{z}^t_{src} \in \mathbb{R}^{\mathscr{D} \times d_l}$ by a factor of 2 and simultaneously project them onto the feature space with a higher dimension 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 $\mathscr{D} \times d_l$ and added to the latent variable $\boldsymbol{z}^t$. The residual layer updates the latent variables based on the concept of the residual of governing equations used in numerical solvers. For a numerical solver, the Newton-Raphson method is commonly applied, which is written as:

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

where $\boldsymbol{J}$ is the Jacobian matrix of the non-linear solver; $\boldsymbol{r}(\boldsymbol{x}^k)$ denotes the residual of the governing equations for iteration k. We update the latent variables and parameterize the update procedure using

neural networks. The dependency of input difference $\delta \boldsymbol{x}^k$ on terms $\boldsymbol{x}^k$ and $\boldsymbol{r}^k$ is parameterized by the operator $\mathscr{F}_{\theta_{\text{ResidualToDiff}}}$. We then introduce another operator $\mathscr{F}_{\theta_{\text{DiffToDiff}}}$ to convert the difference $\delta \boldsymbol{x}^k$ to the latent space $\delta \boldsymbol{z}^k = \boldsymbol{z}^{k+1} - \boldsymbol{z}^k$, where $\mathscr{F}_{\theta_{\text{DiffToDiff}}}$ approximates the mapping between the input difference $\delta \boldsymbol{x}^k$ and the latent difference $\delta \boldsymbol{z}^k$. Then, we can rewrite Eq. (12) as:

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

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

$$\boldsymbol{r}^t = z^t_{acc} - z^{t-1}_{acc} + z^t_{adv} + z^t_{src}, \tag{14}$$

$$\boldsymbol{r}^t_z = \mathscr{F}_{\theta_{\text{ResidualToDiff}}}\left(\begin{bmatrix} z^t_{acc} \\ z^t_{adv} \\ z^t_{src} \end{bmatrix} \circ \begin{bmatrix} \boldsymbol{r}^t \\ \boldsymbol{r}^t \\ \boldsymbol{r}^t \end{bmatrix} \right). \tag{15}$$

The Eq. (14) is similar to Eq. (3), with the distinction that each term in Eq. (3) corresponds to a specific phase. 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 $\delta \boldsymbol{z}^t$ and the updated latent variable z^t can be derived as follows:

$$\begin{aligned} \boldsymbol{d}^t_z &= \mathscr{F}_{\theta_{\text{LatentToDiff}}}(\boldsymbol{z}^t), \\ \delta \boldsymbol{z}^t &= \mathscr{F}_{\theta_{\text{DiffToDiff}}}(\boldsymbol{d}^t_z \circ \boldsymbol{r}^t_z), \\ \boldsymbol{z}^t &= \boldsymbol{z}^t + \delta \boldsymbol{z}^t. \end{aligned} \tag{16}$$

The operator $\mathscr{F}_{\theta_{\text{LatentToDiff}}}$ is to transform the latent variable into a new term to introduce non-linearity. The operator $\mathscr{F}_{\theta_{\text{DiffToDiff}}}$ takes the product $\boldsymbol{d}^t_z \circ \boldsymbol{r}^t_z$ as input and returns the latent residual term $\delta \boldsymbol{z}^t$.

Experimental Setup

In this study, we first investigate the model's performance in the presence of unseen pairs of control variables and permeability inputs. This assessment requires the model to generalize and handle complex and diverse scenarios. Then, 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. This extrapolation task poses a significant challenge due to the distinctive dynamics involved. We applied our model to a set of simulated datasets generated by a synthetic 3D simulation model of a deep saline aquifer, as shown in Figure (3). The simulation model consists of CO_2 injection over 15 years, followed by a post-injection period of 15 years. 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.

We proposed a modified version of Recurrent R-U-Net by modifying the original model proposed by 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. Specifically, we introduced control input to the Recurrent R-U-Net and redefined the input 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 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 as the loss function, which is defined as follows:

$$L(\{\boldsymbol{S}, \boldsymbol{p}\}_{t_1:t_2}, \{\hat{\boldsymbol{S}}, \hat{\boldsymbol{p}}\}_{t_1:t_2}) = \frac{\|\boldsymbol{S}_{t_1:t_2} - \hat{\boldsymbol{S}}_{t_1:t_2}\|_2}{\|\boldsymbol{S}_{t_1:t_2}\|_2} + \frac{\|\boldsymbol{p}_{t_1:t_2} - \hat{\boldsymbol{p}}_{t_1:t_2}\|_2}{\|\boldsymbol{p}_{t_1:t_2}\|_2}, \tag{17}$$

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.

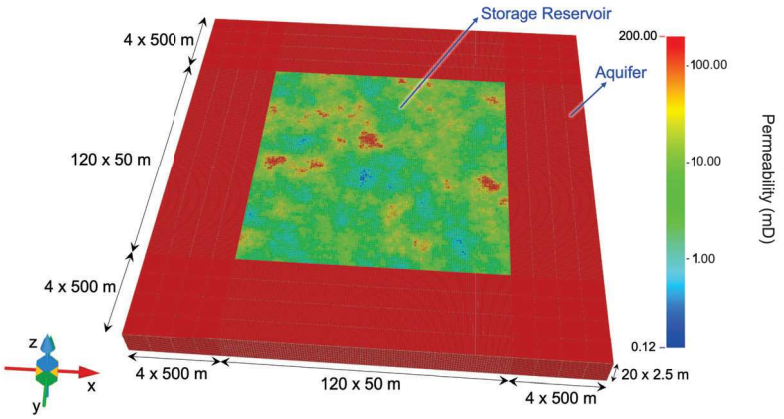

Figure 3 *Numerical simulation model of 3D deep saline aquifer reservoir.*

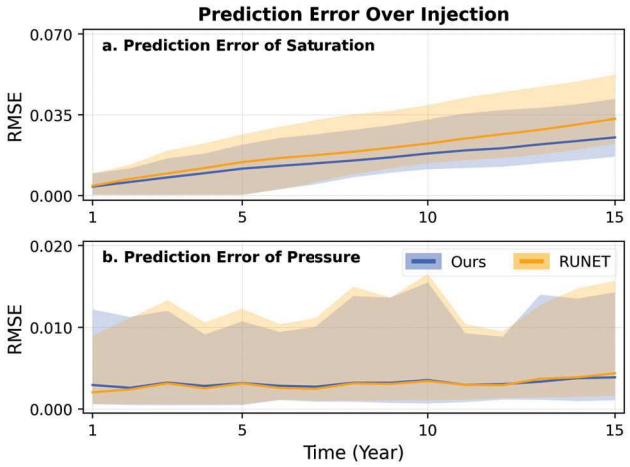

Figure 4 *Prediction errors of saturation and normalized pressure over the test set for two models: our proposed model (Ours) and modified Recurrent R-U-Net (RUNET). The error band and solid line represent a 95% confidence interval and the median of RMSE, respectively.*

Results and Discussion

Testing on Unseen Control and Permeability

In this experiment, we evaluate our model for the simulation of subsurface CO_2 storage. Specifically, the performances of two models (our model and the modified Recurrent R-U-Net) are evaluated on the test set where both control and permeability are unseen during training. The two models 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. For brevity, we will refer to the modified Recurrent R-U-Net as RUNET in the following experiments.

Figure 4 shows that our proposed model exhibits consistently lower mean and variance of RMSE compared to RUNET. The saturation error shows increasing trends over time due to the spreading of the saturation front. In contrast, the pressure error is more stationary, which is attributed to the presence of an aquifer region surrounding the storage reservoir, facilitating pressure dissipation and maintaining the pressure within a certain range. Figures 5 and 6 provide visualizations of saturation and pore pressure predictions for both models. The errors in both saturation and pressure tend to occur in areas where

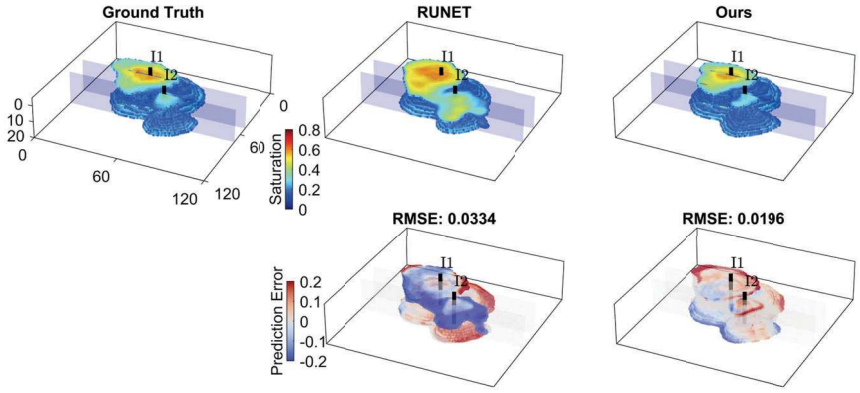

Figure 5 *Visualization of saturation prediction over unseen control and permeability.*

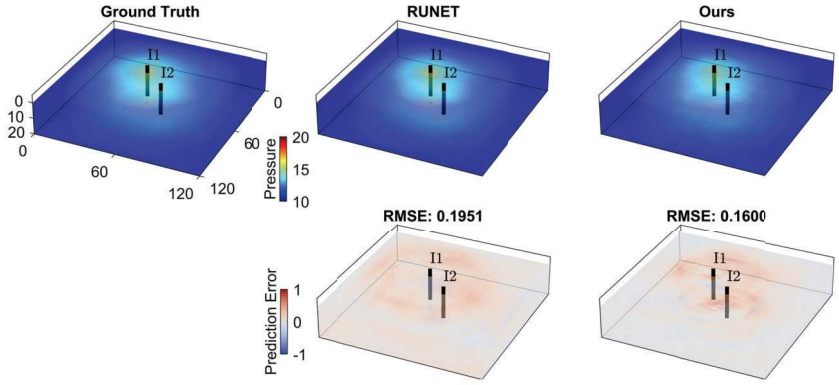

Figure 6 *Visualization of pressure prediction over unseen control and permeability. The pressure values are measured in MPa, and the RMSE is also reported in MPa.*

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. In Figure 5, 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.

Extrapolation over Post-Injection

In this experiment, we extend the application of these trained models to predict the storage reservoir response over 30 years, including 15 years of injection and 15 years of post-injection. Specifically, models trained in the previous experiment are directly applied to predict the last 15 years, from the 16th to the 30th year, with the initial state being the predicted saturation and pressure in the 15th year. Figure 7 shows that our model exhibits significantly lower prediction errors for saturation and remains 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 approaches the boundary pressure during post-injection. Therefore, the pressure prediction becomes less challenging for the two models, as both exhibit decreasing prediction errors after the 15th year when the post-injection period starts. Despite the comparable performance of pressure prediction for both models during the injection period (Figure 7 (b)), our model consistently outperforms RUNET in terms of lower mean and variance of prediction errors during post-injection. This indicates the robustness and accuracy of our model in handling the extrapolation task of predicting the distinct dynamics of the post-injection period.

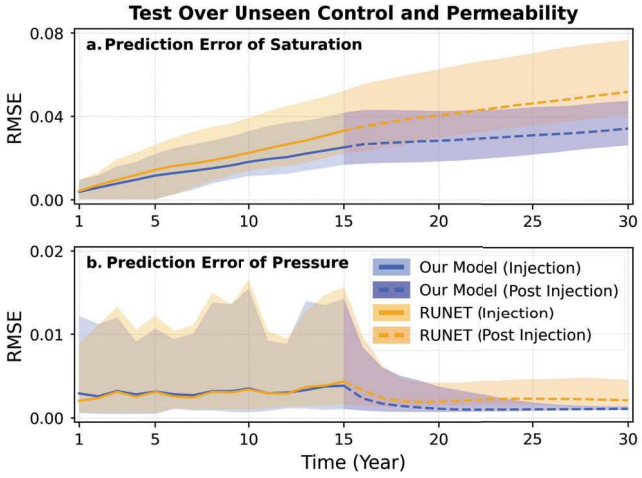

Figure 7 *Prediction errors of normalized saturation and normalized pressure over injection and post-injection for the examples of (left) unseen permeability and (right) unseen control and permeability. The first 15 years denote the injection, while the last 15 years denote the post-injection where injection rates are zeros for two wells.*

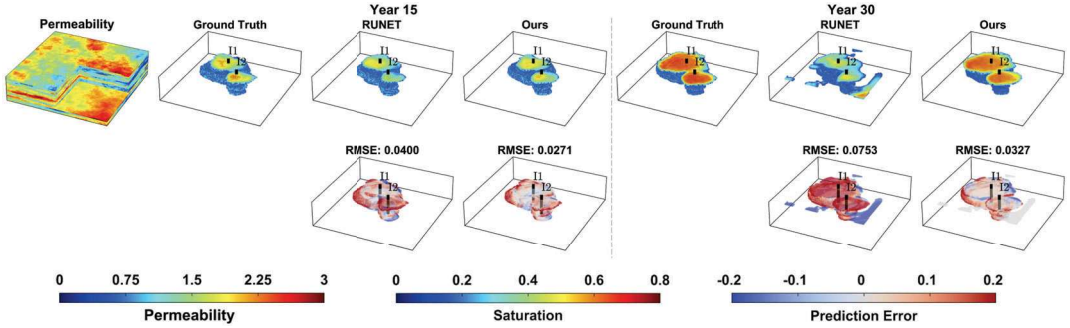

Figure 8 *Visualization of saturation prediction in the 15th and 30th years. The permeability map is in the unit of* $\log_{10}(mD)$.

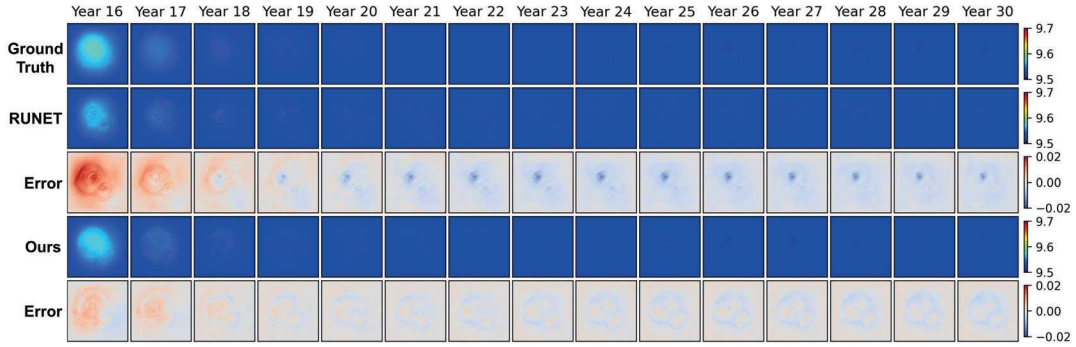

Figure 9 *Visualization of pressure prediction over the post-injection period.*

Figure 8 visualizes an example of the saturation predictions for two models in the 15th and 30th years. During the post-injection period, CO_2 is dominantly driven by the buoyant force and accumulates on the top layer, which is overburdened by cap rock. A common failure of RUNET is its physically inconsistent prediction during extrapolation. Specifically, the predicted CO_2 plume from RUNET in the 30th year shows disconnected patterns and deviates significantly from the reference case. On the other hand, our model can extrapolate the CO_2 plume migration beyond the injection period. Figure 9 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 the FFDL model in capturing long-term behavior and accurately predicting the saturation evolution beyond the injection period.

Conclusion

In this work, we propose a novel deep learning architecture, named FFDL, that incorporates the general structure of the physics of fluid flow in porous media. We use the model to predict the spatial-temporal distribution of pore pressure and phase saturation during geologic CO_2 storage. Specifically, we introduce a new approach that inherits the general structure of the underlying physics (by encoding it into the encoder and processor components of the architectures) and learns the specifics of the flow behavior from simulated data through a training process. We investigate the prediction performance of FFDL over different tasks and demonstrate its ability to learn the underlying physics-based behavior under varying permeability distributions and time-varying control trajectories. The model is also used to perform an extrapolation task by making predictions over both the injection and post-injection periods while being trained based on data from only the injection period. 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, including heterogeneous porosity, and different well locations. Another promising future work is the integration of physics-informed loss functions into our model to enable data-free training or transfer learning.

Acknowledgements

The authors acknowledge the support provided by Energi Simulation through an Industrial Research Chair on Subsurface Energy Data Science at the University of Southern California.

References

Ajayi, T., Gomes, J.S. and 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.

Bandilla, K.W., Celia, M.A., Birkholzer, J.T., Cihan, A. and Leister, E.C. [2015] Multiphase Modeling of Geologic Carbon Sequestration in Saline Aquifers. *Groundwater*, **53**, 362–377.

Bui, M., Adjiman, C.S., Bardow, A., Anthony, E.J., Boston, A., Brown, S., Fennell, P.S., Fuss, S., Galindo, A., Hackett, L.A., Hallett, J.P., Herzog, H.J., Jackson, G., Kemper, J., Krevor, S., Maitland, G.C., Matuszewski, M., Metcalfe, I.S., Petit, C., Puxty, G., Reimer, J., Reiner, D.M., Rubin, E.S., Scott, S.A., Shah, N., Smit, B., Trusler, J.P.M., Webley, P., Wilcox, J. and Mac Dowell, N. [2018] Carbon capture and storage (CCS): the way forward. *Energy & Environmental Science*, **11**, 1062–1176.

Celia, M.A., Bachu, S., Nordbotten, J.M. and 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.

Chen, R.T.Q., Rubanova, Y., Bettencourt, J. and Duvenaud, D.K. [2018] Neural Ordinary Differential Equations. In: *Advances in Neural Information Processing Systems*, 31. Curran Associates, Inc., 6571–6583.

Dulny, A., Hotho, A. and Krause, A. [2022] NeuralPDE: Modelling Dynamical Systems from Data. In: *KI 2022: Advances in Artificial Intelligence*, Lecture Notes in Computer Science. Springer International Publishing, 75–89.

E, W., Han, J. and 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.

Faroughi, S.A., Pawar, N., Fernandes, C., Raissi, M., Das, S., Kalantari, N.K. and Mahjour, S.K. [2023] Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks in Scientific Computing. ArXiv:2211.07377 [cs].

Gorecki, C.D., Ayash, S.C., Liu, G., Braunberger, J.R. and 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.

He, K., Zhang, X., Ren, S. and Sun, J. [2016] Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

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.

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

Lu, Y., Zhong, A., Li, Q. and 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*. PMLR, 3276–3285.

Mo, S., Zhu, Y., Zabaras, N., Shi, X. and 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.

Pawar, R.J., Bromhal, G.S., Carey, J.W., Foxall, W., Korre, A., Ringrose, P.S., Tucker, O., Watson, M.N. and 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.

Qin, Z., Jiang, A., Faulder, D., Cladouhos, T.T. and Jafarpour, B. [2024] Physics-Guided Deep Learning for Prediction of Energy Production from Geothermal Reservoirs. *Geothermics*, **116**, 102824.

Rao, C., Sun, H. and Liu, Y. [2021] Hard Encoding of Physics for Learning Spatiotemporal Dynamics.

Tang, M., Ju, X. and Durlofsky, L.J. [2022] Deep-learning-based coupled flow-geomechanics surrogate model for CO_2 sequestration. *International Journal of Greenhouse Gas Control*, **118**, 103692.

Tang, M., Liu, Y. and 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.

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.

Wang, Y. and Lin, G. [2020] Efficient deep learning techniques for multiphase flow simulation in heterogeneous porous media. *Journal of Computational Physics*, **401**, 108968.

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

Zheng, F., Jahandideh, A., Jha, B. and Jafarpour, B. [2021] Geologic CO_2 Storage Optimization under Geomechanical Risk Using Coupled-Physics Models. *International Journal of Greenhouse Gas Control*, **110**, 103385.

Zheng, F., Jha, B. and 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. and Zabaras, N. [2018] Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, **366**, 415–447.