Fourier-MIONet: Fourier-enhanced multiple-input neural operators for multiphase modeling of geological carbon sequestration

Zhongyi Jiang [a], Min Zhu [a], Lu Lu [a,b,*]

[a] *Department of Statistics and Data Science, Yale University, New Haven, CT 06511, USA*
[b] *Wu Tsai Institute, Yale University, New Haven, CT 06510, USA*

ARTICLE INFO

ABSTRACT

Geologic carbon sequestration (GCS) is a safety-critical technology that aims to reduce the amount of carbon dioxide in the atmosphere, which also places high demands on reliability. Multiphase flow in porous media is essential to understand CO_2 migration and pressure fields in the subsurface associated with GCS. However, numerical simulation for such problems in 4D is computationally challenging and expensive, due to the multiphysics and multiscale nature of the highly nonlinear governing partial differential equations (PDEs). It prevents us from considering multiple subsurface scenarios and conducting real-time optimization. Here, we develop a Fourier-enhanced multiple-input neural operator (Fourier-MIONet) to learn the solution operator of the problem of multiphase flow in porous media. Fourier-MIONet utilizes the recently developed framework of the multiple-input deep neural operators (MIONet) and incorporates the Fourier neural operator (FNO) in the network architecture. Once Fourier-MIONet is trained, it can predict the evolution of saturation and pressure of the multiphase flow under various reservoir conditions, such as permeability and porosity heterogeneity, anisotropy, injection configurations, and multiphase flow properties. Compared to the enhanced FNO (U-FNO), the proposed Fourier-MIONet has 90% fewer unknown parameters, and it can be trained in significantly less time (about 3.5 times faster) with much lower CPU memory (< 15%) and GPU memory (< 35%) requirements, to achieve similar prediction accuracy. In addition to the lower computational cost, Fourier-MIONet can be trained with only 6 snapshots of time to predict the PDE solutions for 30 years. Furthermore, we observed that Fourier-MIONet can maintain good accuracy when predicting out-of-distribution (OOD) data. The excellent generalizability of Fourier-MIONet is enabled by its adherence to the physical principle that the solution to a PDE is continuous over time. Moreover, the developed Fourier-MIONet makes it possible to solve the long-time evolution of geological carbon sequestration in a large-scale three-dimensional space accurately and efficiently.

1. Introduction

Multiphase flow in porous media is crucial in various industry and natural processes, such as geologic carbon sequestration (GCS) [1], enhanced oil recovery [2], hydrogen production [2,3], and nuclear waste geological repositories [4], which have high requirements for ensuring utmost reliability and safety. Numerical simulation is primarily used to solve the material and energy balances of the multiphase flow system [5,6], which involves highly nonlinear partial differential equations [7] and requires high-resolution grids [8,9] and multiscale and multiphysics modeling [10,11]. Large-scale geological carbon sequestration projects involve optimization and decision-making tasks under subsurface uncertainties [12–15], where a large number of forward simulations need to be conducted under various subsurface scenarios and realizations. For this purpose, conventional reservoir simulators are inefficient due to their high computational cost, whereas (physics-informed and/or data-driven) surrogate modeling has the potential to enable real-time predictions once trained on a reasonable amount of data from full-physics simulations [16–19].

Surrogate modeling based on deep neural networks that can learn solutions of PDEs is a promising alternative to traditional numerical simulation [20–25]. Deep learning techniques have recently been used to solve subsurface flow and transport problems in carbon storage [23,26,27]. The two most common deep learning techniques are data-driven learning [20–22,28–34] and physics-informed learning [35–39]. The former has achieved accurate predictions even for high-dimensional multiphase flow problems [23,40–42]. However, data-driven learning requires a large amount of data for training. Physics-informed neural networks (PINNs) [43] and their extensions [44–46]

---

* Corresponding author at: Department of Statistics and Data Science, Yale University, New Haven, CT 06511, USA.
  E-mail address: lu.lu@yale.edu (L. Lu).

have shown promising applications in computational science and engineering [47–52]. The physics-informed learning is usually implemented using artificial neural networks (ANNs) [53,54], thus requiring separate training if any new parameters or coefficients change [35]. Similar to traditional numerical simulations, physics-informed learning for solving geologic carbon storage problems can be computationally inefficient. In some cases, additional physical constraints or observations are needed to guarantee convergence [55–57].

In recent years, deep neural operators [58–63] have shown promising potential for many applications in computational science and engineering. Unlike traditional deep learning methods, deep neural operators are designed to learn operators between infinite-dimensional function spaces. Several numerical experiments have demonstrated the promise of deep neural operators, such as Fourier neural operator (FNO) [26], for real-time prediction of reservoir simulation. Very recently, an extension of FNO, U-FNO, has achieved the state-of-the-art performance in highly heterogeneous geological formations for carbon sequestration problems [26]. However, U-FNO requires a couple of days for training in one GPU with about 16 GiB GPU memory and more than 100 GiB CPU memory, and it also needs a large training dataset and performs poorly on out-of-distribution (OOD) data.

In this work, we overcome these challenges of high computational cost and data efficiency. We propose a novel deep neural operator, Fourier-MIONet, based on the multiple-input deep neural operators (MIONet) [64]. MIONet is an extension of the deep neural operator (DeepONet) [58,61] and it enables several input functions defined on different domains. We further incorporate U-FNO in MIONet. Fourier-MIONet requires much less GPU and CPU memory and significantly reduces the computational cost without compromising prediction accuracy. Moreover, to simulate geological carbon sequestration for 30 years, the required number of time snapshots can be reduced from 24 to only 6. Additionally, Fourier-MIONet achieves better generalization for OOD scenarios. This finding could have very important implications for reducing monitoring costs and safety-critical applications.

This study presents a significant contribution to the field of GCS through the development of the Fourier-MIONet. By integrating MIONet and U-FNO, Fourier-MIONet offers an efficient and accurate method for simulating multiphase flow in porous media. The model substantially reduces computational costs, using 85% less CPU memory and 64% less GPU memory, and accelerates training by 3.5 times compared to U-FNO. Additionally, its ability to generalize from limited data ensures accurate long-term predictions of CO_2 migration and pressure fields. This contribution benefits real-time optimization and decision-making in GCS, a safety-critical technology, improving the feasibility and effectiveness of large-scale carbon sequestration projects aimed at mitigating climate change.

The paper is organized as follows. In Section 2, we introduce the problem setup, including the governing equations of multiphase flow and the dataset generated by numerical simulation. In Section 3, after providing a brief overview of MIONet and U-FNO, we propose Fourier-MIONet. In Section 4, we demonstrate the accuracy, efficiency, and reliability of Fourier-MIONet on multiphase flow prediction for gas saturation and pressure. We conclude the paper in Section 5 and give a brief discussion of the 4D problem.

2. Problem setup

In this study, we consider the problem of multiphase flow in porous media. Specifically, we consider the application of underground carbon storage (also called geological carbon sequestration), which involves injecting supercritical CO_2 into underground geological formations for permanent storage. Geological carbon sequestration is one of the most important technologies for mitigating CO_2 emissions [65,66].

2.1. Multiphase flow

The problem we are considering is multiphase, multicomponent (CO_2, brine) flow in porous media. Each component α satisfies the mass conservation equation:

$$\frac{\partial M^\alpha}{\partial t} = -\nabla \cdot \left(\mathbf{F}^\alpha|_{adv} + \mathbf{F}^\alpha|_{dif} \right) + q^\alpha,$$

where $\mathbf{F}^\alpha|_{adv}$ is the advective mass flux, $\mathbf{F}^\alpha|_{dif}$ is the diffusive mass flux, q^α is the volumetric flow rate of the injection source, and M^α is the accumulation of mass given by

$$M^\alpha = \phi \sum_p S_p \rho_p X_p^\alpha.$$

Here, ϕ is the porosity, S_p is the saturation of phase p, X_p^α is the mass fraction of component α in phase p, and ρ_p is the density of phase p.

For simplicity, we do not consider molecular diffusion and hydrodynamic dispersion, and thus $\mathbf{F}^\alpha|_{dif} = 0$. The advective mass flux of component α is

$$\mathbf{F}^\alpha|_{adv} = \sum_p X_p^\alpha \rho_p \mathbf{u}_p.$$

Here, $\mathbf{u}_p$ is the Darcy velocity of phase p as follows:

$$\mathbf{u}_p = -k \left(\nabla P_p - \rho_p \mathbf{g} \right) k_{rp}/\mu_p,$$

where k is the absolute permeability tensor, P_p is the fluid pressure of phase p, $\mathbf{g}$ is the gravitational acceleration, k_{rp} is the relative permeability of phase p, and μ_p is the viscosity of phase p. The fluid pressure for wetting phase P_w or non-wetting phase P_n is

$$P_n = P_w + P_c,$$

where P_c is the capillary pressure.

2.2. Dataset

In this study, we use the open-source dataset from Wen et al. [26], generated for the purpose of testing surrogate modeling algorithms. To generate the dataset, we consider that CO_2 is injected at a constant rate into a radially symmetrical system. The thickness of the reservoir ranges from 12.5 to 200 m and the radius of the reservoir is 100,000 m. Closed boundary condition is applied at the top and bottom of the reservoir, meaning the flux normal to the boundary is zero. To mimic an open reservoir condition, a very large simulation domain was used with no flow boundary condition. The system is solved numerically with the finite difference method by the simulator ECLIPSE (e300) for 30 years.

In the dataset, several PDE parameters (i.e., network inputs) are randomly sampled to generate different PDE solutions (i.e., network outputs): gas saturation (SG) and pressure buildup (dP). These inputs can be categorized into two types (Table 1): space-dependent inputs (field inputs) and scalar inputs. Field inputs include the horizontal permeability map (k_x), vertical permeability map (k_y), and porosity map (ϕ). Permeability maps k_x are produced by the Stanford Geostatistical Modeling Software (SGeMS) [67] based on different parameters, including correlation lengths in the vertical and radial directions, medium appearances, permeability mean and standard deviation. The permeability value ranges widely from 0.001 mD to 10,000 mD. Values of k_x are binned into n_{aniso} tropic materials, where each bin is assigned a randomly sampled anisotropy ratio (k_x/k_y). Then the vertical permeability map k_y is computed by multiplication between k_x and the anisotropy map. Porosity (ϕ) is calculated by loosely correlated permeability map with a random Gaussian noise $\mathcal{N}(0, 0.001)$.

The scalar inputs include the initial reservoir pressure at the top of the reservoir (P_{init}), reservoir temperature (T), injection rate (Q), capillary pressure scaling factor (λ), irreducible water saturation (S_{wi}), and the top and bottom locations of the perforation ($perf_{top}$ and $perf_{bottom}$). We note that the reservoir thickness b is a variable randomly sampled in each case and determines the size of the field inputs, ranging from

Table 1

Summary of the branch network inputs.The size of the field inputs is (96, 200). $\mathcal{V}[a, b]$ means that the input value ranges from a to b.

	Parameter	Notation	Distribution	Unit
Field	Horizontal permeability field	k_x	$\mathcal{V}[0.001, 10000]$	mD
	Material anisotropy ratio	k_x/k_y	$\mathcal{V}[1, 150]$	–
	Porosity	ϕ	–	–
Scalar	Injection rate	Q	$\mathcal{V}[0.2, 2]$	MT/y
	Initial pressure	P_{init}	$\mathcal{V}[100, 300]$	bar
	Iso-thermal reservoir temperature	T	$\mathcal{V}[35, 170]$	°C
	Irreducible water saturation	S_{wi}	$\mathcal{V}[0.1, 0.3]$	–
	Van Genuchten scaling factor	λ	$\mathcal{V}[0.3, 0.7]$	–
	Perforation top location	$perf_{top}$	$\mathcal{V}[0, 200]$	m
	Perforation bottom location	$perf_{bottom}$	$\mathcal{V}[0, 200]$	m
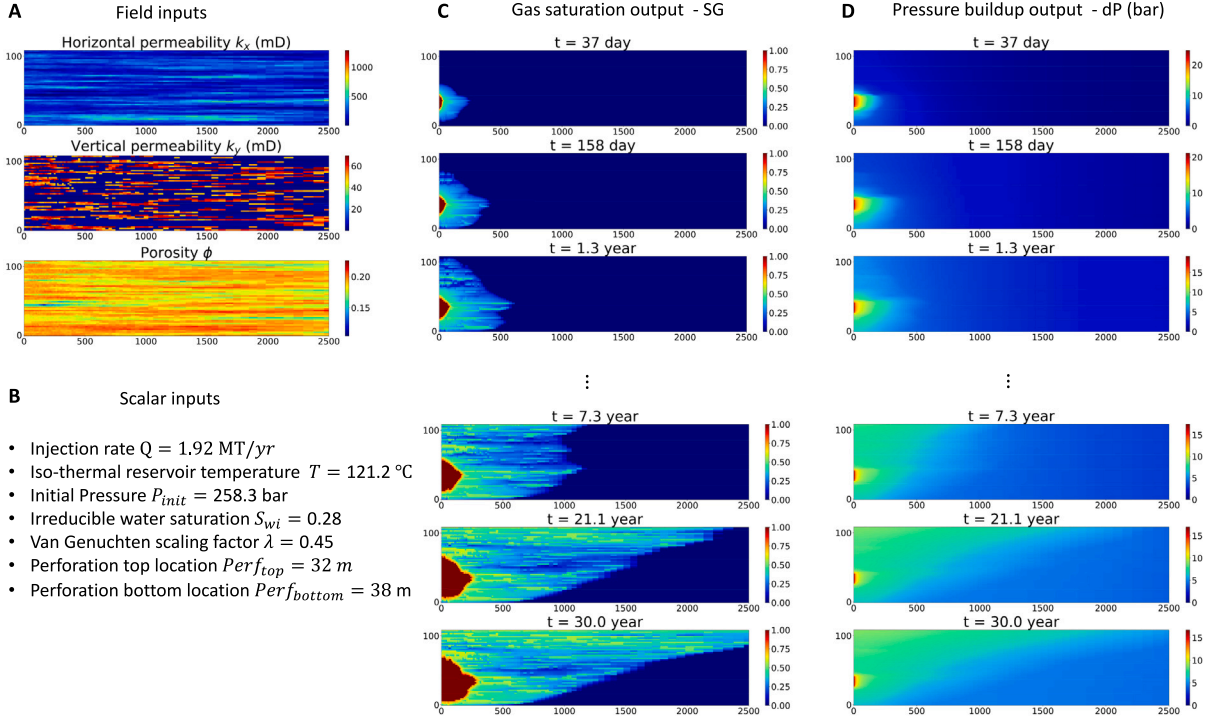
Fig. 1. An example of inputs and outputs. (A) Example of field inputs. (B) Example of scalar inputs. (C) Gas saturation evolution at 6 out of 24 time snapshots. (D) Pressure buildup evolution at 6 out of 24 time snapshots.

12.5 to 200 m. When b is thinner than 200 m, zero padding is used for the domain outside of the actual reservoir such that all inputs have the same shape. Then, $perf_{top}$ and $perf_{bottom}$ are randomly sampled within the range from 0 to b of each case.

All the input and output fields in the dataset are saved in a resolution of (96, 200). The outputs are saved at 24 time snapshots from 1 day to 30 years: {1 day, 2 days, 4 days, … ,14.8 years, 21.1 years, 30 years}. The training dataset includes 4500 cases, and the test dataset includes 500 cases. We show an example of inputs and outputs in Fig. 1.

3. Methods

U-FNO has been developed to build a surrogate model for such dataset, and has shown great prediction accuracy. However, there are still some challenges remaining: the computational cost is high, as we have discussed in the introduction. Moreover, these existing methods always use different channels to predict the solutions at different times, which do not guarantee the continuity of solution over time. The recently developed framework of the multiple-input deep neural operators (MIONet) overcomes the challenges of computational cost and data efficiency, but MIONet by itself cannot easily provide highly accurate predictions. In this section, we first briefly introduce MIONet and U-FNO. Next, we propose Fourier-enhanced multiple-input neural

operator (Fourier-MIONet) by combining the advantages of MIONet and U-FNO together for learning physical systems more effectively.

3.1. MIONet

MIONet was proposed by Jin et al. [64] for learning nonlinear operators mapping between function spaces. Based on the universal approximation theorem of Chen & Chen [68], the vanilla DeepONet is defined for input functions on a single Banach space. MIONet extends the capability of DeepONet [58] both theoretically and numerically from a single Banach space to multiple Banach spaces. MIONet can be understood as a multivariable regression method in the statistical regression area, but MIONet is different from other special regression models [69,70] which were designed to exploit the relationships among multiple variables.

We denote n input functions by v_i for $i = \{1, \ldots, n\}$ with each defined on the domain $D_i \subset \mathbb{R}^{d_i}$:

$$v_i : D_i \to \mathbb{R},$$

and the output function by u defined on the domain $D' \in \mathbb{R}^{d'}$:

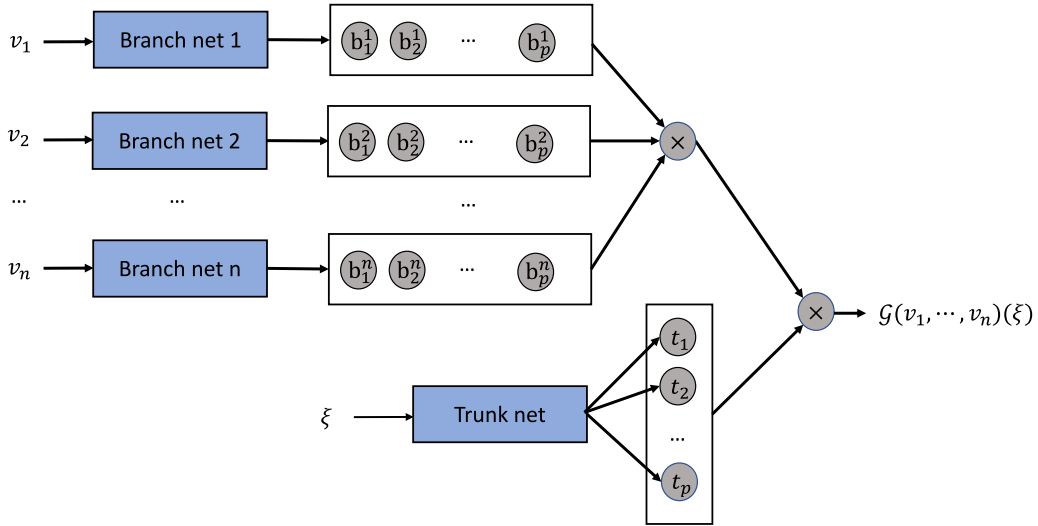$$u : D' \ni \xi \mapsto u(\xi) \in \mathbb{R}.$$

Fig. 2. Architecture of MIONet. All the branch nets and the trunk net have the same number of outputs, which are merged together via the element-wise product.

Then, the operator mapping from the input functions to the output function is

$$\mathcal{G} : (v_1, \dots, v_n) \mapsto u.$$

MIONet is developed to learn the operator $\mathcal{G}$ by using n independent branch nets and one trunk net (Fig. 2). The ith branch net encodes the input function v_i, and the trunk net encodes the coordinates input ξ. The output of MIONet is computed as

$$\mathcal{G}(v_1, v_2, \dots, v_n)(\xi) = \sum_{j=1}^{p} \underbrace{b_j^1(v_1)}_{\text{branch}_1} \times \underbrace{b_j^2(v_2)}_{\text{branch}_2} \cdots \times \underbrace{b_j^n(v_n)}_{\text{branch}_n} \times \underbrace{t_j(\xi)}_{\text{trunk}} + b_0,$$

where $b_0 \in \mathbb{R}$ is a bias, $\left\{ b_1^i, b_2^i, \dots, b_p^i \right\}$ are the p outputs of the branch net i, and $\{t_1, t_2, \dots, t_p\}$ are the p outputs of the trunk net. The choice of the branch nets and trunk net is problem dependent. For example, if the discretization of v_i is on an equispaced 2D grid, then a convolution neural network can be used as the branch net i; if the discretization of v_i is on an unstructured mesh, then a graph neural network can be used as the branch net i.

3.2. U-FNO

U-FNO [26] is an extension of Fourier neural operator (FNO) [59], which computes convolutions in the Fourier space rather than physical space. U-FNO uses additional U-Net blocks [71] in FNO. Different from DeepONet, U-FNO requires the input function $v(x)$ and output function $u(\xi)$ defined on the same equispaced mesh grid of the same domain. A schematic diagram of U-FNO is shown in Fig. 3.

First, the input function $v(x)$ is lifted to a higher dimensional representation $z_{l_0}(x)$ by

$$z_{l_0}(x) = P(v(x)),$$

where P is a linear transformation, which is often parameterized by a linear layer or a shallow fully-connected neural network. Then, L Fourier layers and M U-Fourier layers are applied iteratively to z_{l_0} and the final output is z_{m_M}. In the end, a local linear transformation Q is applied by employing a shallow neural network to convert the z_{m_M} to the output by

$$u(x) = Q\left(z_{m_M}(x) \right).$$

Each Fourier layer is defined by using the Fast Fourier Transform (FFT). For the output of the jth Fourier layer z_{l_j}, we compute the following transform by 3D FFT $\mathcal{F}$ and inverse 3D FFT $\mathcal{F}^{-1}$:

$$\mathcal{F}^{-1}\left(\mathcal{R}_{l_j} \cdot \mathcal{F}\left(z_{l_j} \right) \right),$$

where $\mathcal{R}$ is a weight matrix. Moreover, a residual connection with a weight matrix W_j is used to compute the output of the $(j+1)$th Fourier layer $z_{l_{j+1}}$ as

$$z_{l_{j+1}} = \sigma\left(\mathcal{F}^{-1}\left(\mathcal{R}_{l_j} \cdot \mathcal{F}\left(z_{l_j} \right) \right) + W_j \cdot z_{l_j} + b_{l_j} \right),$$

where σ is a nonlinear activation and b_{l_j} is a bias.

U-Fourier layer is defined based on the Fourier layer and 3D U-Net block:

$$z_{m_{k+1}} = \sigma\left(\mathcal{F}^{-1}\left(\mathcal{R}_{m_k} \cdot \mathcal{F}\left(z_{m_k} \right) \right) + U_{m_k}\left(z_{m_k} \right) + W_{m_k} \cdot z_{m_k} + b_{m_k} \right)$$

where U_{m_k} is an U-Net block.

3.3. Fourier-MIONet

Motivated by the discussion at the beginning of this section, here we develop a new version of MIONet, Fourier-MIONet, by leveraging MIONet and U-FNO. As we have discussed in Section 2, we split the inputs into two parts: for all the field parameters denoted by $\mathbf{v_1}$ and for all the scalar parameters denoted by $\mathbf{v_2}$ (Fig. 4). The coordinates of the output function u are x, y, and t, while x and y are space coordinates and t is time. If we apply MIONet, the trunk net input includes x, y, and t, and the output is $u(x, y, t) \in \mathbb{R}$, while for U-FNO, the network output is $u(\cdot, \cdot, \cdot) \in \mathbb{R}^{96 \times 200 \times 24}$. In our developed Fourier-MIONet, we treat spatial and temporal coordinates separately. Specifically, the trunk input is only t, and the network output is $u(\cdot, \cdot, t) \in \mathbb{R}^{96 \times 200}$.

In Fourier-MIONet, there are two branch nets to encode field inputs and scalar inputs, respectively (Fig. 4A). Let $\mathbf{b}_1$ and $\mathbf{b}_2$ be the outputs of the two branch nets, and $\mathbf{c}$ be the output of the trunk net. Two merger operations are used here: one to merge the two branch outputs (Branch merger operation) and one to merge the branch and trunk net outputs (Branch-Trunk merger operation). We choose the Branch merger operation as a point-wise summation:

$$\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2,$$

where $\mathbf{b}$ is the output of the branch merger operation. We choose the Branch-Trunk merger operation as a point-wise multiplication used in vanilla MIONet:

$$z = \mathbf{b} \odot \mathbf{c},$$

where z is the output of the merger operation. We note that there are other choices of the merger operations, which will be studied in the future.
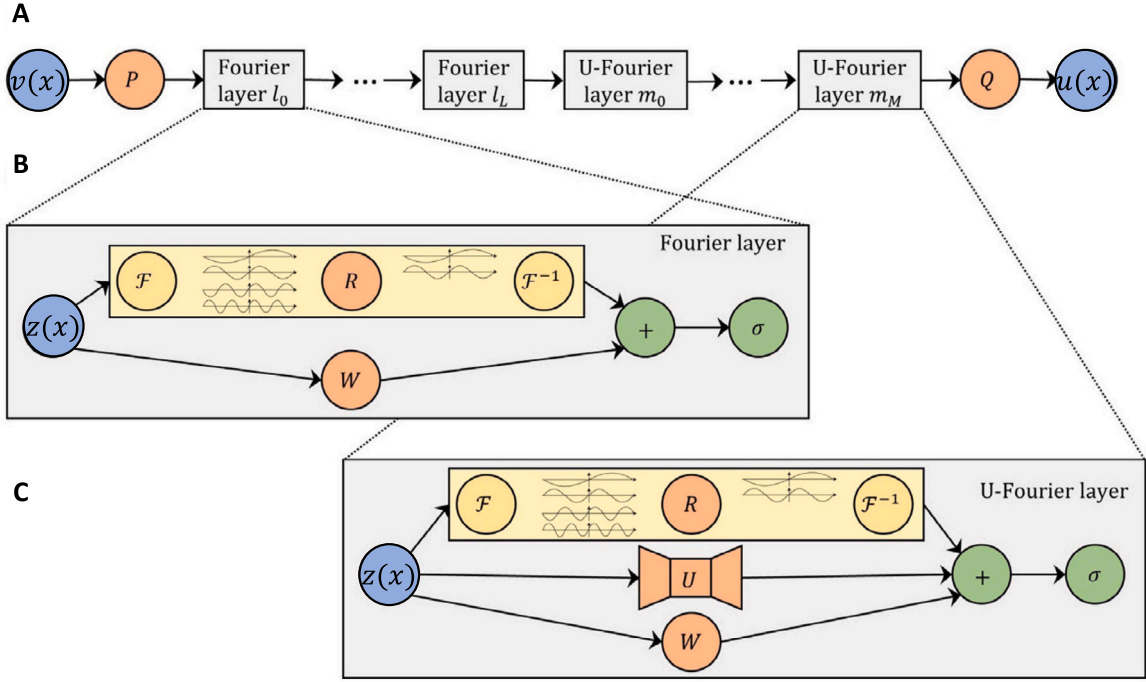
Fig. 3. Architecture of U-FNO. (A) $v(x)$ is the input function, P and Q are fully connected neural networks, and $u(x)$ is the output function. (B) Inside each Fourier layer, $\mathcal{F}$ denotes the Fourier transform, $\mathcal{R}$ is a weight matrix, $\mathcal{F}^{-1}$ is the inverse Fourier transform, W is an another weight matrix, and σ is the activation function. (C) Inside each U-Fourier layer, U denotes a U-Net block.
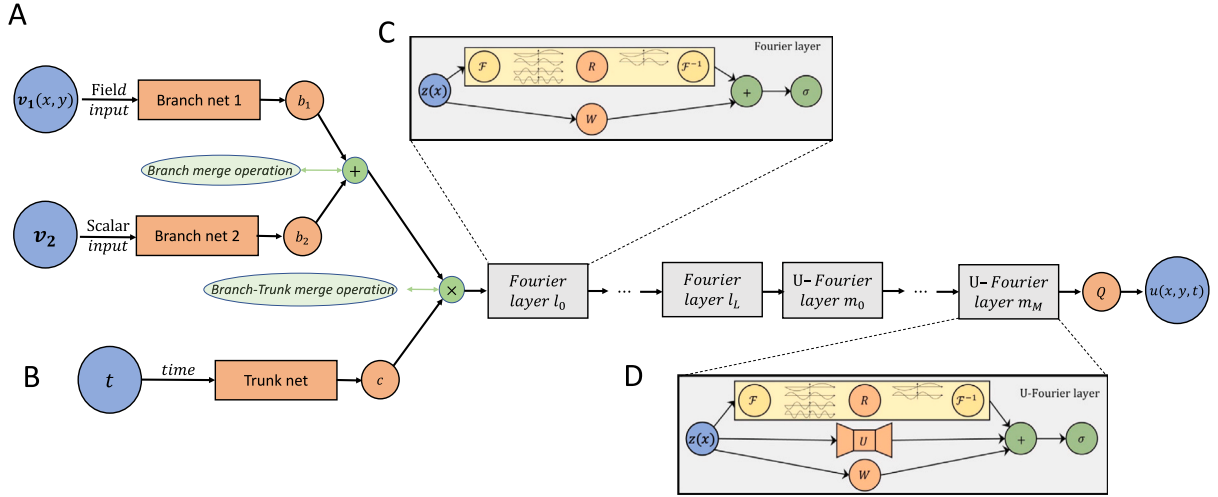*Source:* Figure is adapted from [26].

Fig. 4. Fourier-MIONet architecture. (A) $\mathbf{v}_1$ is field input and $\mathbf{v}_2$ is scalar input. (B) The input t is time. (C) Fourier layer. (D) U-Fourier layer.

Next, we use U-FNO as a decoder to map from the hidden vector z to the output u by applying Fourier layers and U-Fourier layers iteratively (Fig. 4C and D). We note that we use 2D FFT and 2D U-Net in this U-FNO, rather than the 3D FFT and 3D U-Net used in the original U-FNO. In the end, a fully connected neural network Q is applied to project the output of the last U-Fourier layer to the output.

3.4. Training and evaluation

As mentioned in Section 2, our dataset contains reservoirs with various thicknesses, and the domain outside of the reservoir is padded with zeros for both input and output. To accommodate for the variable reservoir thicknesses, we construct a mask for each data sample and only calculate the loss within the mask during training. In addition to

the input variables described in Table 1, we also supply the spatial grid information to the training as additional two field inputs.

Same as U-FNO [26], we use lp-loss function:

$$L(y, \hat{y}) = \frac{\|y - \hat{y}\|_p}{\|y\|_p} + \beta \frac{\|dy/dr - d\hat{y}/dr\|_p}{\| dy/dr\|_p},$$

where y is the true solution and $\hat{y}$ is the prediction. p is the order of norm, and β is a hyper-parameter. We choose $p = 2$ and $\beta = 0.5$. During training, the initial learning rate is specified to be 0.001, same as [26], the learning rate gradually decreases with a constant rate.

U-FNO directly outputs gas saturation and pressure fields at all the 24 time steps during training and testing, i.e., the output size is (96, 200, 24). In each training step, we choose different cases and the number of cases (i.e., batch size) is denoted by $batch_{case}$. For Fourier-MIONet, in addition to $batch_{case}$, we can also choose different number

Table 2

Fourier-MIONet architecture. In order to provide a fair comparison, we adopt a similar structure for U-FNO as in [26]. The "Padding" denotes a padding operation. "Linear" denotes a linear transformation. "Fourier2d" denotes the 2D Fourier transform. "Conv1d" denotes 1D convolution. "UNet2d" denotes a 2D U-Net. C represents $batch_{case}$. T represents $batch_{time}$. In this model, the number of total parameters is 3,685,325.

		Operation	Output shape
Branch net	Branch net 1	Padding/Linear	$(C, 104, 208, 36)$
	Branch net 2	FNN	$(C, 36)$
Branch merger operation		Point-wise summation	$(C, 104, 208, 36)$
Trunk net		FNN	$(T, 36)$
Branch-Trunk merger operation		Point-wise multiplication	$(C \times T, 104, 208, 36)$
Merger net	Fourier 1	Fourier2d/Conv1d/Add/ReLU	$(C \times T, 104, 208, 36)$
	Fourier 2	Fourier2d/Conv1d/Add/ReLU	$(C \times T, 104, 208, 36)$
	Fourier 3	Fourier2d/Conv1d/Add/ReLU	$(C \times T, 104, 208, 36)$
	U-Fourier 1	Fourier2d/Conv1d/UNet2d/Add/ReLU	$(C \times T, 104, 208, 36)$
	U-Fourier 2	Fourier2d/Conv1d/UNet2d/Add/ReLU	$(C \times T, 104, 208, 36)$
	U-Fourier 3	Fourier2d/Conv1d/UNet2d/Add/ReLU	$(C \times T, 104, 208, 36)$
	Projection 1	Linear/ReLU	$(C \times T, 104, 208, 128)$
	Projection 2	Linear	$(C \times T, 104, 208, 1)$
	De-padding	–	$(C \times T, 96, 200, 1)$
	Reshape	–	$(C, T, 96, 200)$

of time snapshots of the trunk net input for training, and we denote this batch size as $batch_{time}$. Hence, when training Fourier-MIONet model, the output size is (96, 200, $batch_{time}$).

To evaluate the performance of the trained network, we use R^2 and mean absolute error (MAE) as the metrics.

4. Results

We demonstrate the accuracy, reliability, training efficiency, and data efficiency of the proposed Fourier-MIONet in this section. The network architecture for the Fourier-MIONet model is shown in Table 2. All experiments are performed on a workstation with an AMD Thread-ripper Pro 5955WX CPU and an NVIDIA GeForce RTX 3090 GPU. We implement the experiments by using the library DeepXDE [37] and the code is available in GitHub at https://github.com/lu-group/fourier-mionet-gcs.

4.1. Gas saturation

We apply Fourier-MIONet to learn gas saturation. As we have discussed in Section 3.4, in Fourier-MIONet, we have the flexibility to select different values of the $batch_{case}$ and $batch_{time}$, while U-FNO only allows for $batch_{case}$ (i.e., $batch_{time}$ can only be chosen as 24). Then, we first select $batch_{time}$ as 24 (i.e., full batch for the time input). The comparison between Fourier-MIONet and U-FNO is shown in the first two rows of Table 3. First of all, the prediction accuracy between U-FNO and Fourier-MIONet is almost the same in terms of R^2 and MAE. Compared with U-FNO, Fourier-MIONet only has 1% lower in the R^2. Compared to U-FNO with 33 million parameters, Fourier-MIONet only has 10% trainable parameters (about 3 million). During training, U-FNO requires 103 GiB CPU memory and 15.9 GiB GPU memory, but Fourier-MIONet only needs 15 GiB CPU memory and 12.8 GiB GPU memory. Moreover, Fourier-MIONet is much faster to train, and the training time per epoch of Fourier-MIONet is 50% less than U-FNO.

As Fourier-MIONet has the flexibility of using different values of $batch_{time}$, here we investigate the effect of $batch_{time}$ and determine the optimal choice of $batch_{time}$. The range of $batch_{time}$ is from 24 to 1. First, the $batch_{time}$ has little effect on the final accuracy (Fig. 5A and B), unless the $batch_{time}$ is very small (e.g., smaller than 4). In terms of computational cost, the GPU memory requirement decreases approximately linearly with respect to $batch_{time}$ (Fig. 5C). However, training time per epoch of the Fourier-MIONet increases when $batch_{time}$ decreases. In general, fewer epochs are needed for training neural networks when smaller batch size is used, which is also observed in Fourier-MIONet.

Hence, for smaller batch size, less epochs are needed for the training to converge. In Fig. 5E, we show the minimum epoch needed for different $batch_{time}$. Since training time per epoch and minimum epoch needed have the opposite trend, the total training time has an approximate U-shaped curve (Fig. 5F). The minimum training time occurs at $batch_{time}$ equal to 8. Therefore, to achieve the fastest training, we use $batch_{time}$ as 8 in the following study.

The accuracy between Fourier-MIONet (with $batch_{time}$ as 8) and U-FNO is almost the same and 4 testing examples of Fourier-MIONet are shown in Fig. 6. GPU memory of Fourier-MIONet is 5.69 GiB (Table 3) which only accounts for 35% of the U-FNO. Total training time of Fourier-MIONet is 12.3 h (less than 30% of U-FNO). Hence, Fourier-MIONet is much more computationally efficient.

In addition to the training cost, we also summarize the computational cost during testing (Table 4). Compared with U-FNO, Fourier-MIONet needs much less CPU memory (33.3%) and GPU memory (50%) during the testing. For each case, Fourier-MIONet requires 0.041 s, while U-FNO requires 0.075 s, so Fourier-MIONet is about 1.8 times faster.

4.2. Inference for unseen time

We also test the generalizability of Fourier-MIONet at unseen training time, i.e., only partial time snapshots are used for training. We recall that in the dataset for each case, there are 24 time snapshots. Here, we choose to test three cases with different amounts of training dataset: 50%, 33%, and 25%.

In the first case, we use 50% of the time snapshots: $\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23\}$. As shown in Fig. 7A and B, Fourier-MIONet has good performance at both seen time index and the unseen time index, and there is nearly no accuracy loss for the unseen time. However, U-FNO only performs well at seen time index, and a significant accuracy drop can be seen at the unseen time index.

The same phenomenon has also been observed for the other two cases of 33% and 25% training data (Fig. 7C, D, E and F). We note that the behavior of U-FNO is typical for deep learning models, because it is usually difficult to generalize to unseen times by training with extremely sparse time snapshots. However, Fourier-MIONet shows good performance in both interpolation and extrapolation. For example in Fig. 7E and F, the training time snapshots are 1 day, 11 day, 53 day, 226 day, 2.6 year, and 10.4 year, and the testing time snapshots include both interpolation (from 2 day to 7.3 year) and extrapolation (14.8 year, 21.1 year, and 30.0 year). The R^2 and MAE at 30.0 years are 0.9616 and 0.0179. Here, Fourier-MIONet can generalize well
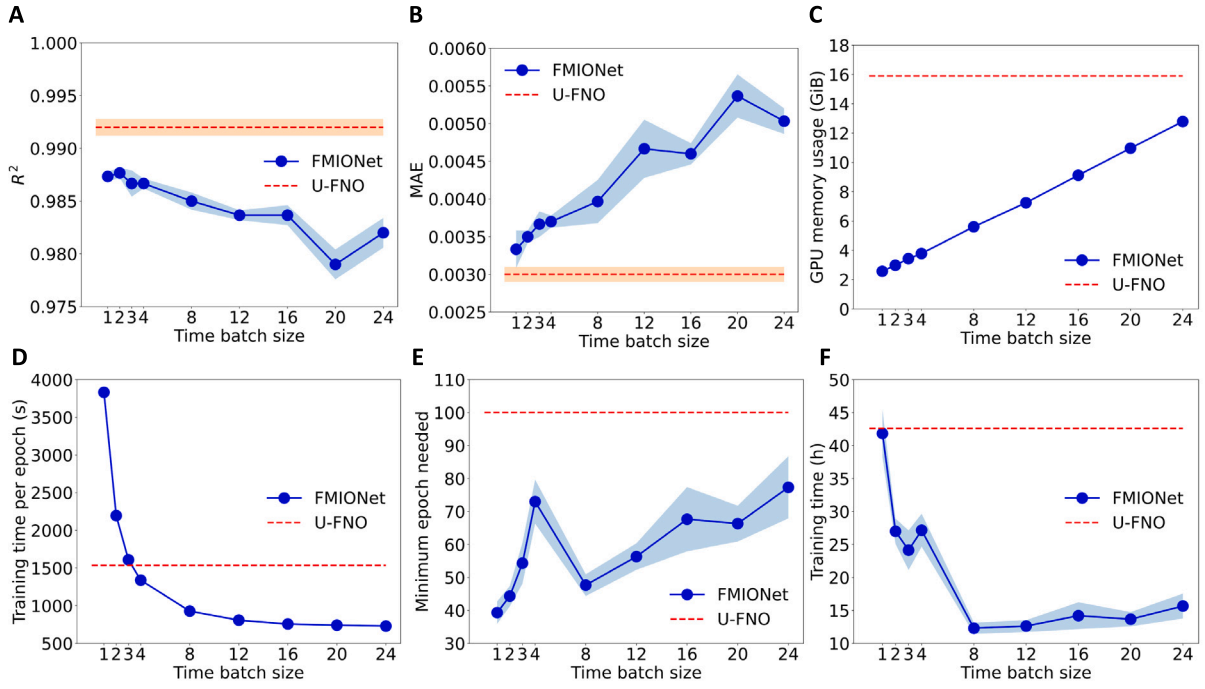
Fig. 5. Effect of $batch_{time}$ for Fourier-MIONet. (A) R^2. (B) MAE. (C) GPU memory usage. (D) Training time per epoch. (E) Minimum number of training epochs needed. (F) Total training time. The solid blue curves and shaded regions represent the mean and one standard deviation of 3 runs of Fourier-MIONet. The precise values in this figure can be found in Table 3. U-FNO does not have time batch size.

Table 3
Comparison between Fourier-MIONet and U-FNO. We choose $batch_{case}$ as 4 for all the experiments. U-FNO uses all 24 time snapshots during training.

	$batch_{time}$	No. of parameters	CPU memory (GiB)	GPU memory (GiB)	Training time per epoch (second)	Minimum epochs needed	Training time (hour)	R^2	MAE
U-FNO	–	33,097,829	103	15.9	1535	100	42.6	0.992 ± 0.001	0.0031 ± 0.0002
FMIONet	24 (Full)	3,685,325	15	12.8	730	77 ± 13	15.7 ± 2.7	0.982 ± 0.002	0.0050 ± 0.0002
	20	3,685,325	15	11.0	742	66 ± 8	13.7 ± 1.5	0.979 ± 0.002	0.0054 ± 0.0004
	16	3,685,325	15	9.1	757	68 ± 13	14.2 ± 2.8	0.984 ± 0.001	0.0046 ± 0.0002
	12	3,685,325	15	7.3	808	56 ± 6	12.6 ± 0.7	0.984 ± 0.001	0.0047 ± 0.0005
	8	3,685,325	15	5.6	928	48 ± 5	12.3 ± 1.2	0.985 ± 0.001	0.0040 ± 0.0004
	4	3,685,325	15	3.8	1339	73 ± 9	27.2 ± 3.4	0.987 ± 0.001	0.0036 ± 0.0002
	3	3,685,325	15	3.4	1609	54 ± 8	24.1 ± 4.0	0.987 ± 0.002	0.0037 ± 0.0002
	2	3,685,325	15	3.0	2195	44 ± 4	27.0 ± 2.7	0.988 ± 0.001	0.0035 ± 0.0001
	1	3,685,325	15	2.6	3832	39 ± 5	41.8 ± 4.2	0.987 ± 0.001	0.0033 ± 0.0003

Table 4
Computational cost of Fourier-MIONet and U-FNO during testing. The inference time is computed by taking the average of 500 test cases.

	CPU memory (GiB)	GPU memory (GiB)	Inference time (s)
U-FNO	15.3	7.1	0.075
FMIONet	5.1	3.5	0.041

Table 5
Six nonuniform sampling cases.

Case	Seen time snapshots	R^2
A	{1d, 7d, 53d, 323d, 5.2y, 30y}	0.9793 ± 0.0247
B	{1d, 4d, 53d, 323d, 5.2y, 30y}	0.9809 ± 0.0049
C	{1d, 4d, 37d, 226d, 5.2y, 30y}	0.9821 ± 0.0063
D	{1d, 4d, 37d, 226d, 3.6y, 30y}	0.9807 ± 0.0092
E	{1d, 4d, 37d, 323d, 5.2y, 30y}	0.9793 ± 0.0040
F	{1d, 4d, 37d, 323d, 7.3y, 30y}	0.9773 ± 0.0048

because we use time as the input of the trunk net, which guarantees that the prediction of gas saturation is continuous with respect to time, i.e., Fourier-MIONet obeys the physics law that gas saturation is continuous over, showcasing a crucial advance in ensuring the reliability of our model under varied conditions. In contrast, U-FNO uses different channels to predict gas saturation at different time, which does not guarantee the continuity of solution over time, weakening generalization ability for out-of-distribution (OOD) scenarios.

4.3. Nonuniform sampling of training data

To further improve the accuracy of using 25% training data (i.e., six time snapshots) in the previous section, we propose to train the network using nonuniform time snapshots. The 24 time snapshots in the datasets

corresponds to 1 day, 2 day, 4 day, 7 day, 11 day, 17 day, 25 day, 37 day, 53 day, 77 day, 111 day, 158 day, 226 day, 323 day, 1.3 year, 1.8 year, 2.6 year, 3.6 year, 5.2 year, 7.3 year, 10.4 year, 14.8 year, 21.1 year, and 30.0 year. Specifically, we design 6 different nonuniform sampling cases as listed in Table 5. The R^2 for each case can be found in Table 5.

For all these 6 cases, the R^2 of all time snapshots is shown in Fig. 8. Since the dynamics of gas saturation changes fast at the beginning, more data need to be chosen at the very first few time snapshots, otherwise, predictions will become worse. For example, if we choose the second time snapshots at 7 day (Fig. 8A), the accuracy at the time 2 day will drop significantly. For later time, we need to choose
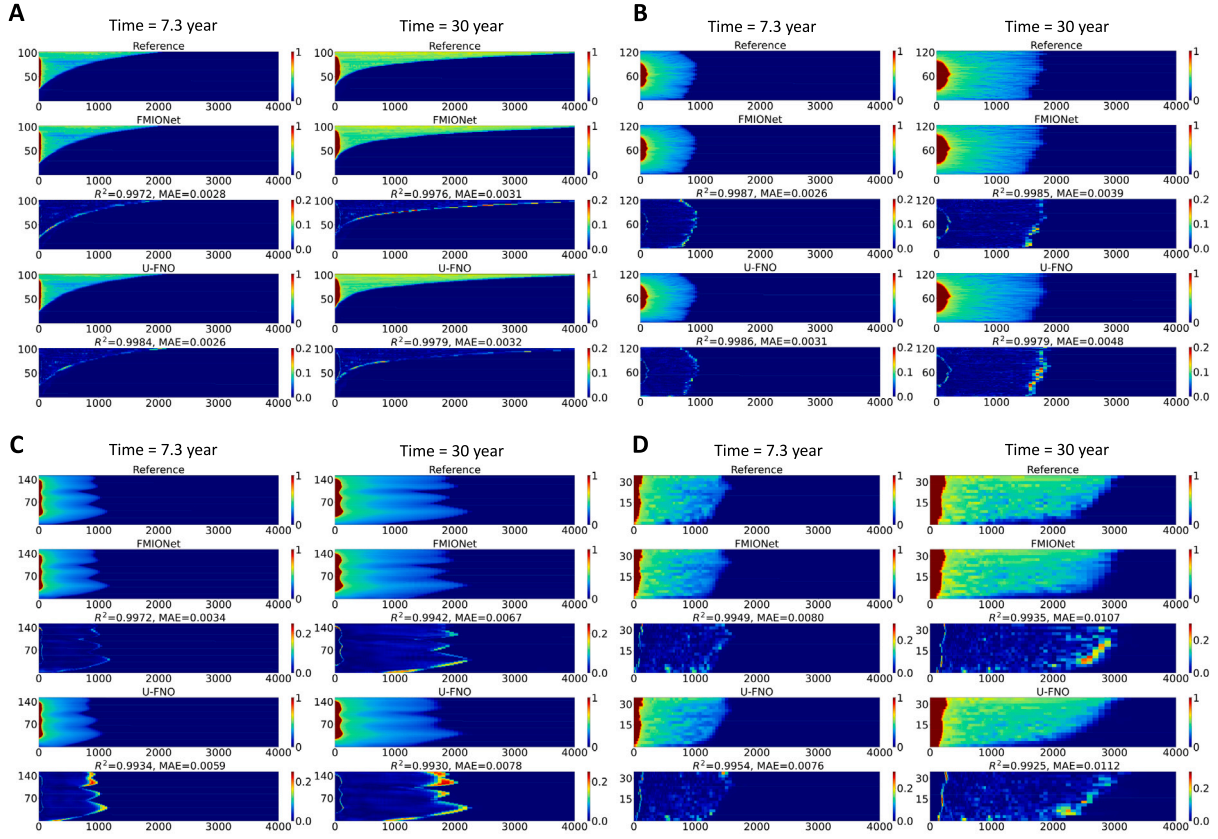
Fig. 6. Four testing examples of gas saturation. For each example, we show the reference solution and the predictions and errors of Fourier-MIONet and U-FNO at 7.3 year and 30 year. The R^2 and MAE error are also computed for each prediction and shown in the figure.

time snapshots relatively uniformly. As shown in Fig. 8D, there are 5 unseen time snapshots between the last two points, which makes the prediction of the unseen time with much lower accuracy. Hence, by choosing dense time snapshots at the beginning and relatively uniform snapshots between later times, Fourier-MIONet can achieve good prediction accuracy at both seen time and unseen time with only 6 time snapshots for training (Fig. 8B, C, E and F).

4.4. Pressure buildup

When it comes to the prediction of pressure buildup, Fourier-MIONet also achieves good accuracy. We follow the same setup as U-FNO and use a smaller network for pressure buildup. Specifically, we only apply one Fourier layer and one U-Fourier layer in the merge net, and other parts of the network architecture are the same as the one in the gas saturation problem. Based on Section 4.1, we choose $batch_{time}$ as 8. The testing R^2 of Fourier-MIONet is 0.986. We show 4 examples of the prediction and absolute error with the corresponding R^2 and MAE in Fig. 9.

4.5. Discussions about baseline network architectures

In this study, we have demonstrated good performance of Fourier-MIONet, and in this section, we show the necessity of using Fourier and U-Fourier layers as the merger net by studying two baseline networks. One baseline is the vanilla MIONet, which uses an inner product to construct the output (Section 3.1). Another baseline is MIONet-FNN, where the dot product in the vanilla MIONet is replaced by a fully-connected neural network (FNN) as the merger net [72,73]. The network architectures of the vanilla MIONet and MIONet-FNN are in Tables 8 and 9, respectively.

Table 6
Accuracy of vanilla MIONet, MIONet-FNN and Fourier-MIONet.

		R^2
Gas saturation	Vanilla MIONet	0.948
	MIONet-FNN	0.971
	Fourier-MIONet	0.985
Pressure buildup	Vanilla MIONet	0.961
	MIONet-FNN	0.979
	Fourier-MIONet	0.986

The testing R^2 of vanilla MIONet for both gas saturation and pressure buildup is the worst among these three models (Table 6) due to the simple dot product in merger operation. By using FNN as the merger net, MIONet-FNN performs better than vanilla MIONet. Fourier-MIONet achieves the best accuracy by employing Fourier and U-Fourier layers as a decoder. Fourier layers with truncated modes can effectively capture low-frequency components in the data, while U-Nets are designed to capture both local and global features.

We also show 4 examples of the gas saturation prediction and absolute error with the corresponding R^2 and MAE in Fig. 10. Vanilla MIONet and MIONet-FNN have good predictions in the case with smooth solutions (Fig. 10A). However, when dealing with cases with non-smooth solutions, these two methods result in blurred outputs of interface structures (Fig. 10B, C, and D). In contrast, Fourier-MIONet generates accurate predictions no matter whether the solution is smooth or not (Fig. 6).

5. Conclusions

In this paper, we propose a novel deep neural operator, a Fourier-enhanced multiple-input neural operator (Fourier-MIONet), which combines the advantages of data and computational efficiency from MIONet
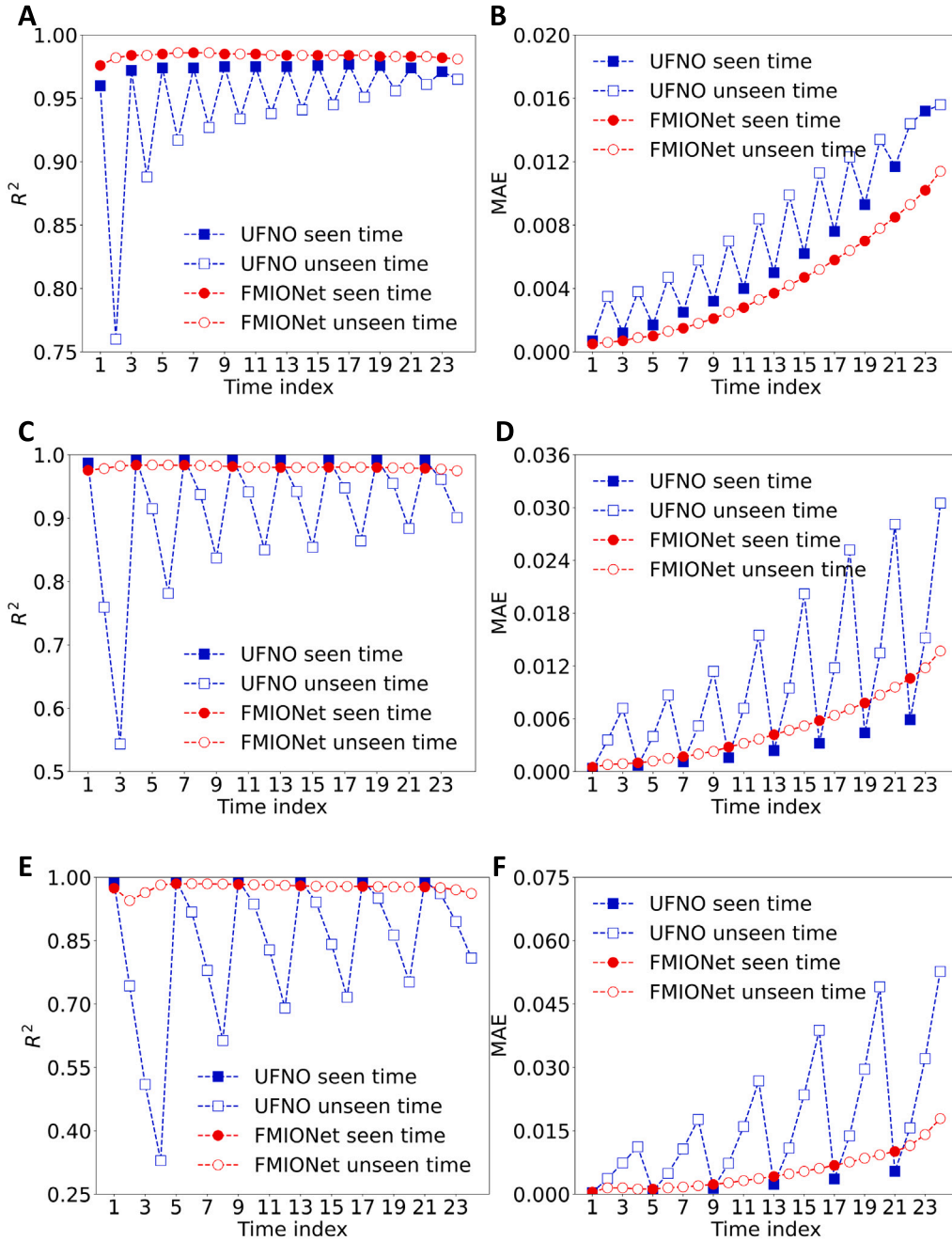
Fig. 7. Three different test cases for unseen time. (A and B) Training at time index $\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23\}$. (C and D) Training at time index $\{1, 4, 7, 10, 13, 16, 19, 22\}$. (E and F) Training at time index $\{1, 5, 9, 13, 17, 21\}$.

and good accuracy from U-FNO. We show that Fourier-MIONet is both accurate and efficient for solving the problem of multiphase flow in porous media while ensuring reliability, for safety-critical applications including geological carbon sequestration. Compared with U-FNO, Fourier-MIONet saves 85% CPU memory and 64% GPU memory, and can be trained 3.5 times faster.

Fourier-MIONet also provides much more accurate predictions at unseen time snapshots with only six time snapshots in each case for training. This good generalizability is due to that Fourier-MIONet obeys the physics that the PDE solution is continuous over time, which provides an interpretable basis for maintaining reliability when handling diverse and complex OOD data. We also consider the nonuniform sampling method to further improve the generalizability.

Here, we give a brief discussion of the time-dependent 3D problem (i.e., 4D problem), since the heterogeneity in a geologic formation is

beyond the radially symmetric models. As we do not have a 4D dataset available, we generate a synthetic dataset with the mesh size (100, 100, 18, 24), which means the x-axis resolution is 100, y-axis resolution is 100, z-axis resolution is 18, and there are in total 24 time snapshots. We find that U-FNO is too big to be trained in a GPU with 24 GiB memory, even if the $batch_{case}$ is chosen as 1, and thus we use FNO for the comparison. Fourier-MIONet also uses FNO as the decoder merge net. We have two choices for Fourier-MIONet. First, we only time as the trunk net input and then the merge net should be FNO with 3D FFT. Compared with FNO, Fourier-MIONet (3D FFT) with $batch_{time}$ as 8 only has 10% the number of parameters and needs much less GPU memory (<30%) (Table 7). Second, we can further use z-coordinates combined with time as the trunk net input, and then we only need 2D FFT in the merge net. In this case, the number of parameters and GPU memory can be further decreased. The Fourier-MIONet with 2D FFT
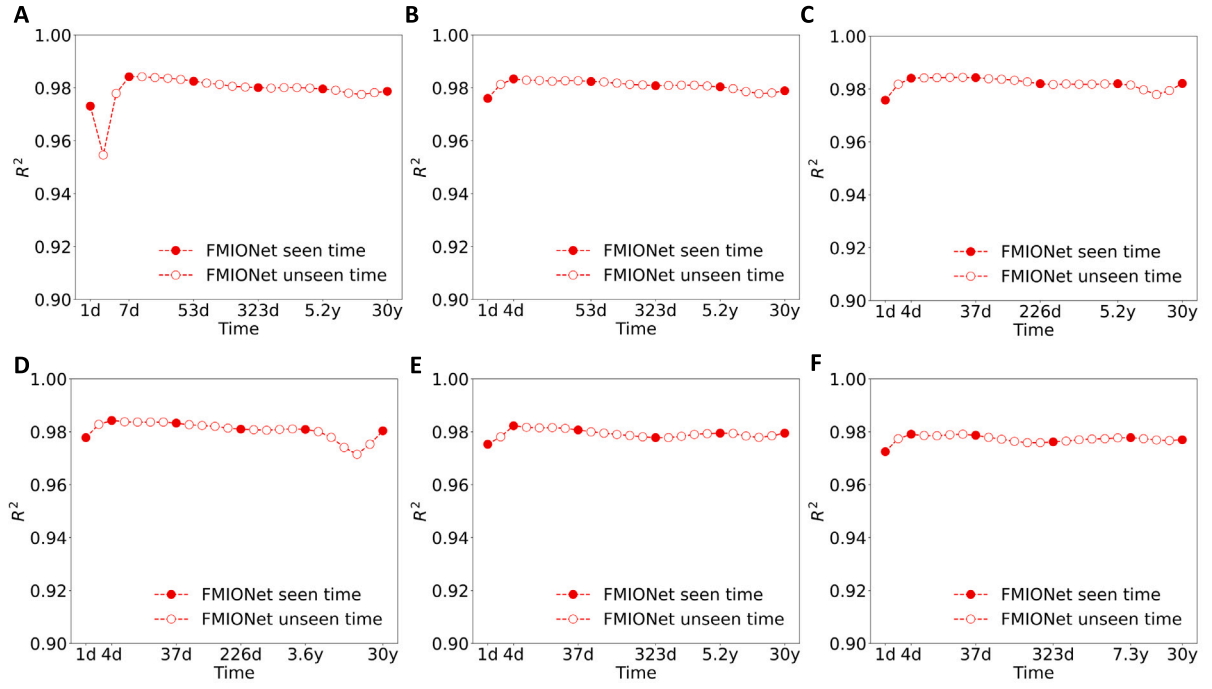
Fig. 8. Six nonuniform sampling of time. (A) Training at time snapshots: {1d, 7d, 53d, 323d, 5.2y, 30y}. (B) Training at time snapshots: {1d, 4d, 53d, 323d, 5.2y, 30y}. (C) Training at time snapshots: {1d, 4d, 37d, 226d, 5.2y, 30y}. (D) Training at time snapshots: {1d, 4d, 37d, 226d, 3.6y, 30y}. (E) Training at time snapshots: {1d, 4d, 37d, 323d, 5.2y, 30y}. (F) Training at time snapshots: {1d, 4d, 37d, 323d, 7.3y, 30y}.
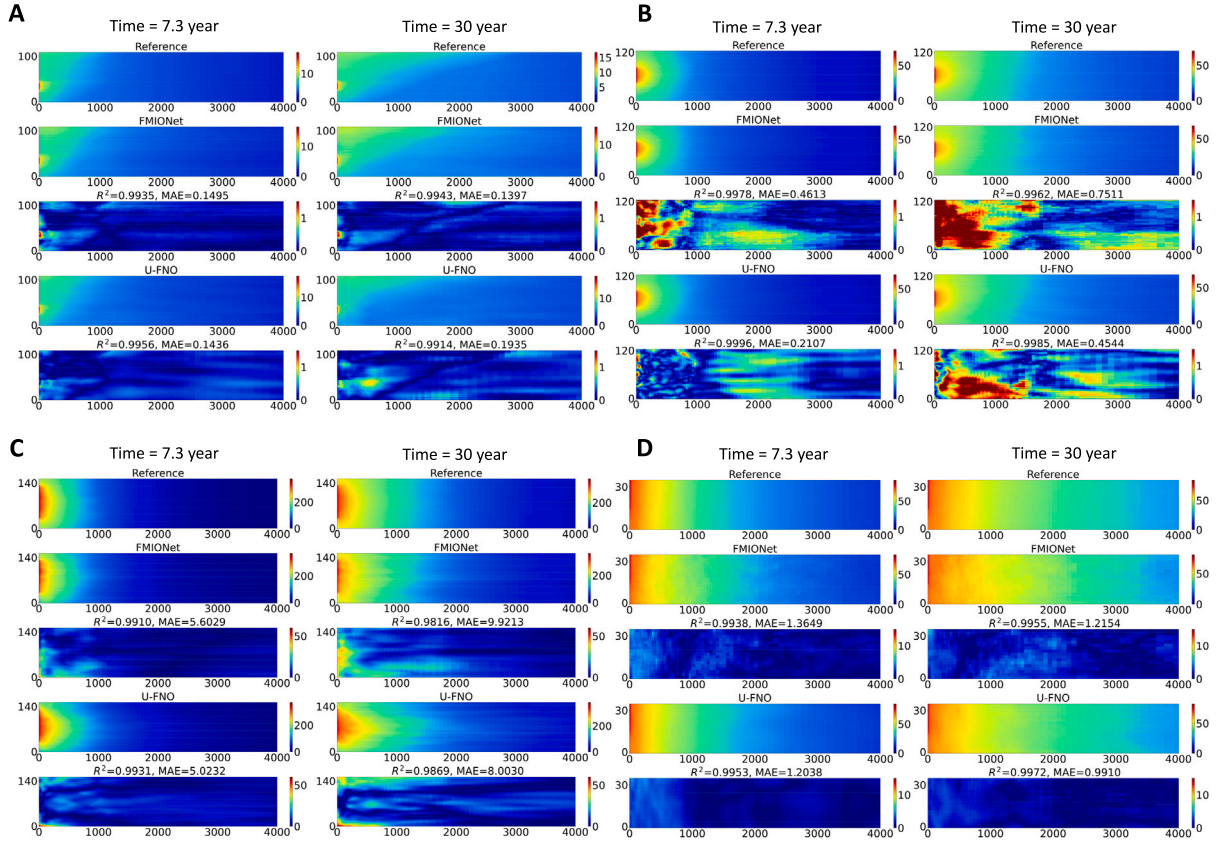

Fig. 9. Four testing examples of pressure buildup. The reference solution and the predictions and errors of Fourier-MIONet and U-FNO at 7.3 year and 30 year. The R^2 and MAE error are also computed for each prediction and shown in the figure.

has 3.3% number of trainable parameters and 14.7% GPU memory. The advantages of our architecture show more potential possibilities for 4D multiphase problems, though more work is required to evaluate the performance with 4D datasets in the future.
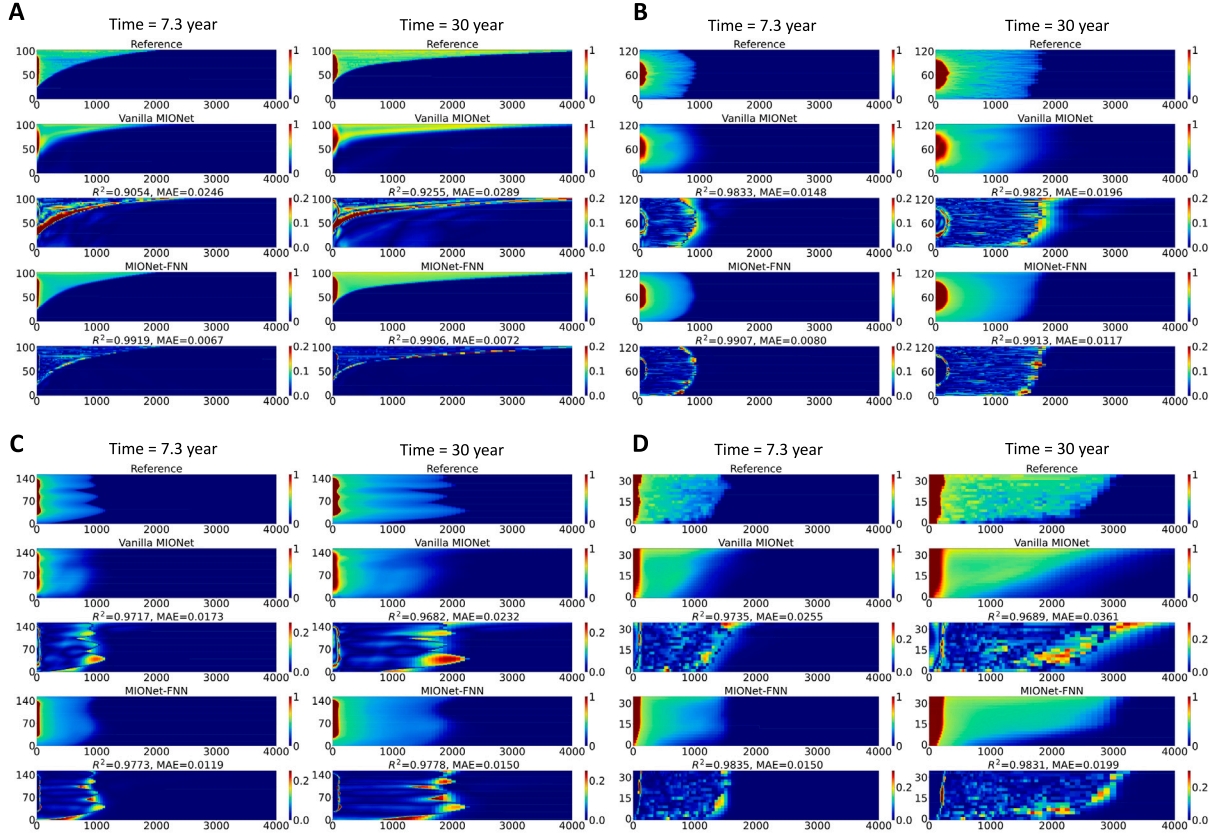
Fig. 10. Four testing examples of vanilla MIONet and MIONet-FNN for gas saturation. For each example, we show the reference solution and the predictions and errors of vanilla MIONet and MIONet-FNN at 7.3 and 30 years. The R^2 and MAE error are also computed for each prediction.

Table 7
Comparison between Fourier-MIONet and FNO on time-dependent 3D problems. We use $batch_{case}$ as 1 for all the experiments.

	$batch_{time}$	$batch_z$	No. of parameters	GPU memory (GiB)
FNO (4D FFT)	–	–	46,665,329	23.2
FMIONet (3D FFT)	24 (Full)	–	4,679,033	16.3
	8	–	4,679,033	6.7
FMIONet (2D FFT)	8	6	1,568,669	3.4

One limitation of the proposed Fourier-MIONet is that it cannot be directly applied to irregular geometry because Fourier and U-Fourier layers require the input to be a regular geometry. There are two possible solutions to this issue. One solution is mapping the irregular geometry to a regular geometry such as gFNO+ [60]. Another solution is using other networks as a merger net, such as MIONet-FNN. Another limitation of U-FNO and the proposed Fourier-MIONet is that they cannot provide accurate predictions of the CO_2 saturation plume front (Fig. 6). One possible solution is adding more Fourier and U-Fourier layers in the decoder merge net, but it will also cost more computing resources. How to improve the prediction of CO_2 plume front will be investigated in future studies.

CRediT authorship contribution statement

Zhongyi Jiang: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Min Zhu:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Lu Lu:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Appendix. Network architectures

The network architectures of the vanilla MIONet and MIONet-FNN in Section 4.5 are shown in Tables 8 and 9, respectively, while the "CNN" denotes a convolutional neural network in Table 10.

Table 8

Vanilla MIONet architecture.

		Operation	Output shape
Branch net	Branch net 1	CNN	$(C, 512)$
	Branch net 2	FNN	$(C, 512)$
Branch merger operation		Point-wise multiplication	$(C, 512)$
Trunk net		FNN	$(T \times 96 \times 200, 512)$
Branch-Trunk merger operation		Dot product Reshape	$(C, T \times 96 \times 200)$ $(C, T, 96, 200)$

Table 9

MIONet-FNN architecture.

		Operation	Output shape
Branch net	Branch net 1	CNN	$(C, 512)$
	Branch net 2	FNN	$(C, 512)$
Branch merger operation		Point-wise multiplication	$(C, 512)$
Trunk net		FNN	$(T \times 96 \times 200, 512)$
Branch-Trunk merger operation		Point-wise multiplication	$(C \times T \times 96 \times 200, 512)$
Merger net		FNN/Reshape	$(C, T, 96, 200)$

Table 10

CNN architecture. The "Conv3d" denotes 3D convolution. "LeakyReLU" denotes a leaky rectified linear unit with the negative slope as 0.2.

Layer	Operation	Output shape
1	Conv3D/LeakyReLU	$(C, 32, 96, 100)$
2	Conv3D/LeakyReLU	$(C, 64, 48, 50)$
3	Conv3D/LeakyReLU	$(C, 64, 48, 50)$
4	Conv3D/LeakyReLU	$(C, 64, 24, 25)$
5	Conv3D/LeakyReLU	$(C, 64, 24, 25)$
6	Conv3D/LeakyReLU	$(C, 128, 12, 13)$
7	Conv3D/LeakyReLU	$(C, 128, 12, 13)$
8	Conv3D/LeakyReLU	$(C, 256, 6, 7)$
9	Conv3D/LeakyReLU	$(C, 256, 6, 7)$
10	Conv3D/LeakyReLU/Flatten	$(C, 512)$
11	Linear/LeakyReLU	$(C, 2048)$
12	Linear	$(C, 512)$

References

[1] Pachauri RK, Meyer LA. Climate change 2014: Synthesis report. Contribution of working groups I, II and III to the fifth assessment report of the intergovernmental panel on climate change, 2014.

[2] Aziz K. Petroleum reservoir simulation. 476, Applied Science Publishers; 1979.

[3] Hashemi L, Blunt M, Hajibeygi H. Pore-scale modelling and sensitivity analyses of hydrogen-brine multiphase flow in geological porous media. Sci Rep 2021;11(1):1–13.

[4] Amaziane B, El Ossmani M, Jurak M. Numerical simulation of gas migration through engineered and geological barriers for a deep repository for radioactive waste. Comput Vis Sci 2012;15(1):3–20.

[5] Prosperetti A, Tryggvason G. Computational methods for multiphase flow. Cambridge University Press; 2009.

[6] Balachandar S, Eaton JK. Turbulent dispersed multiphase flow. Annu Rev Fluid Mech 2010;42:111–33.

[7] Orr FM, et al. Theory of gas injection processes, vol. 5, Tie-Line Publications Copenhagen; 2007.

[8] Doughty C. Investigation of CO_2 plume behavior for a large-scale pilot test of geologic carbon storage in a saline formation. Transp Porous Media 2010;82(1):49–76.

[9] Wen G, Benson SM. CO_2 plume migration and dissolution in layered reservoirs. Int J Greenh Gas Control 2019;87:66–79.

[10] Khebzegga O, Iranshahr A, Tchelepi H. Continuous relative permeability model for compositional simulation. Transp Porous Media 2020;134(1):139–72.

[11] Meng X, Guo Z. Localized lattice Boltzmann equation model for simulating miscible viscous displacement in porous media. Int J Heat Mass Transfer 2016;100:767–78.

[12] Strandli CW, Mehnert E, Benson SM. CO_2 plume tracking and history matching using multilevel pressure monitoring at the illinois basin – decatur project. Energy Procedia 2014;63:4473–84, 12th International Conference on Greenhouse Gas Control Technologies, GHGT-12.

[13] Kitanidis PK. Persistent questions of heterogeneity, uncertainty, and scale in subsurface flow and transport. Water Resour Res 2015;51(8):5888–904.

[14] Kröker I, Oladyshkin S. Arbitrary multi-resolution multi-wavelet-based polynomial chaos expansion for data-driven uncertainty quantification. Reliab Eng Syst Saf 2022;222:108376.

[15] Rehme MF, Franzelin F, Pflüger D. B-splines on sparse grids for surrogates in uncertainty quantification. Reliab Eng Syst Saf 2021;209:107430.

[16] Tahmasebi P, Kamrava S, Bai T, Sahimi M. Machine learning in geo- and environmental sciences: From small to large scale. Adv Water Resour 2020;142:103619.

[17] Cardoso MA, Durlofsky LJ, Sarma P. Development and application of reduced-order modeling procedures for subsurface flow simulation. Internat J Numer Methods Engrg 2009;77(9):1322–50.

[18] Razavi S, Tolson BA, Burn DH. Review of surrogate modeling in water resources. Water Resour Res 2012;48(7).

[19] Bazargan H, Christie M, Elsheikh AH, Ahmadi M. Surrogate accelerated sampling of reservoir models with complex structures using sparse polynomial chaos expansion. Adv Water Resour 2015;86:385–99.

[20] Zhu Y, Zabaras N. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. J Comput Phys 2018;366:415–47.

[21] Mo S, Zhu Y, Zabaras N, Shi X, Wu J. Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. Water Resour Res 2019;55(1):703–28.

[22] Tang M, Liu Y, Durlofsky LJ. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. J Comput Phys 2020;413:109456.

[23] Wen G, Hay C, Benson SM. CCSNet: A deep learning modeling suite for CO_2 storage. Adv Water Resour 2021;155:104009.

[24] Liu Y, Wang D, Sun X, Dinh N, Hu R. Uncertainty quantification for Multiphase-CFD simulations of bubbly flows: a machine learning-based Bayesian approach supported by high-resolution experiments. Reliab Eng Syst Saf 2021;212:107636.

[25] Wu H, Xu Y, Liu Z, Li Y, Wang P. Adaptive machine learning with physics-based simulations for mean time to failure prediction of engineering systems. Reliab Eng Syst Saf 2023;240:109553.

[26] Wen G, Li Z, Azizzadenesheli K, Anandkumar A, Benson SM. U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow. Adv Water Resour 2022;163:104180.

[27] Wen G, Li Z, Long Q, Azizzadenesheli K, Anandkumar A, Benson SM. Accelerating carbon capture and storage modeling using Fourier neural operators. 2022, arXiv preprint arXiv:2210.17051.

[28] Zhong Z, Sun AY, Jeong H. Predicting CO_2 plume migration in heterogeneous formations using conditional deep convolutional generative Adversarial Network. Water Resour Res 2019;55(7):5830–51.

[29] Wen G, Tang M, Benson SM. Towards a predictor for CO_2 plume migration using deep neural networks. International Journal of Greenhouse Gas Control 2021;105:103223.

[30] Zhu M, Feng S, Lin Y, Lu L. Fourier-DeepONet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness. Comput Methods Appl Mech Engrg 2023;416:116300.

[31] Jiao A, He H, Ranade R, Pathak J, Lu L. One-shot learning for solution operators of partial differential equations. 2021, arXiv preprint arXiv:2104.05512.

[32] Liu X-Y, Sun H, Zhu M, Lu L, Wang J-X. Predicting parametric spatiotemporal dynamics by multi-resolution PDE structure-preserved deep learning. 2022, arXiv preprint arXiv:2205.03990.

[33] Fan S, Yang Z. Accident data-driven human fatigue analysis in maritime transport using machine learning. Reliab Eng Syst Saf 2024;241:109675.

[34] Kandel R, Baroud H. A data-driven risk assessment of Arctic maritime incidents: Using machine learning to predict incident types and identify risk factors. Reliab Eng Syst Saf 2024;243:109779.

[35] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.

[36] Zhu Y, Zabaras N, Koutsourelakis P-S, Perdikaris P. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. J Comput Phys 2019;394:56–81.

[37] Lu L, Meng X, Mao Z, Karniadakis GE. DeepXDE: A deep learning library for solving differential equations. SIAM Rev 2021;63(1):208–28.

[38] Cai S, Mao Z, Wang Z, Yin M, Karniadakis GE. Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mech Sin 2021;37(12):1727–38.

[39] Fan B, Qiao E, Jiao A, Gu Z, Li W, Lu L. Deep learning for solving and estimating dynamic macro-finance models. 2023, arXiv preprint arXiv:2305.09783.

[40] Jiang Z, Tahmasebi P, Mao Z. Deep residual U-net convolution neural networks with autoregressive strategy for fluid flow predictions in large-scale geosystems. Adv Water Resour 2021;150:103878.

[41] Tang M, Liu Y, Durlofsky LJ. Deep-learning-based surrogate flow modeling and geological parameterization for data assimilation in 3D subsurface flow. Comput Methods Appl Mech Engrg 2021;376:113636.

[42] Wu H, Qiao R. Physics-constrained deep learning for data assimilation of subsurface transport. Energy AI 2021;3:100044.

[43] Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nat Rev Phys 2021;3(6):422–40.

[44] Pang G, Lu L, Karniadakis GE. fPINNs: Fractional physics-informed neural networks. SIAM J Sci Comput 2019;41(4):A2603–26.

[45] Yu J, Lu L, Meng X, Karniadakis GE. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. Comput Methods Appl Mech Engrg 2022;393:114823.

[46] Wu C, Zhu M, Tan Q, Kartha Y, Lu L. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. Comput Methods Appl Mech Engrg 2023;403:115671.

[47] Chen Y, Lu L, Karniadakis GE, Dal Negro L. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. Opt Express 2020;28(8):11618–33.

[48] Lu L, Pestourie R, Yao W, Wang Z, Verdugo F, Johnson SG. Physics-informed neural networks with hard constraints for inverse design. SIAM J Sci Comput 2021;43(6):B1105–32.

[49] Yazdani A, Lu L, Raissi M, Karniadakis GE. Systems biology informed deep learning for inferring parameters and hidden dynamics. PLoS Comput Biol 2020;16(11):e1007575.

[50] Daneker M, Zhang Z, Karniadakis GE, Lu L. Systems Biology: Identifiability analysis and parameter identification via systems-biology informed neural networks. 2022, arXiv preprint arXiv:2202.01723.

[51] Song C, Xiao R, Zhang C, Zhao X, Sun B. Simulation-free reliability analysis with importance sampling-based adaptive training physics-informed neural networks: Method and application to chloride penetration. Reliab Eng Syst Saf 2024;246:110083.

[52] Das S, Tesfamariam S. Reliability assessment of stochastic dynamical systems using physics informed neural network based PDEM. Reliab Eng Syst Saf 2024;243:109849.

[53] Kamrava S, Sahimi M, Tahmasebi P. Simulating fluid flow in complex porous materials by integrating the governing equations with deep-layered machines. npj Comput Mater 2021;7(1).

[54] Wang K, Chen Y, Mehana M, Lubbers N, Bennett KC, Kang Q, Viswanathan HS, Germann TC. A physics-informed and hierarchically regularized data-driven model for predicting fluid flow through porous media. J Comput Phys 2021;443:110526.

[55] Fuks O, Tchelepi H. Physics based deep learning for nonlinear two-phase flow in porous media. In: ECMOR XVII. 2020.

[56] Almajid MM, Abu-Al-Saud MO. Prediction of porous media fluid flow using physics informed neural networks. J Pet Sci Eng 2022;208:109205.

[57] Fraces CG, Tchelepi H. Physics informed deep learning for flow and transport in porous media. In: SPE reservoir simulation conference. OnePetro; 2021.

[58] Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat Mach Intell 2021;3(3):218–29.

[59] Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A. Fourier neural operator for parametric partial differential equations. 2020, arXiv preprint arXiv:2010.08895.

[60] Lu L, Meng X, Cai S, Mao Z, Goswami S, Zhang Z, Karniadakis GE. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. Comput Methods Appl Mech Engrg 2022;393:114778.

[61] Zhu M, Zhang H, Jiao A, Karniadakis GE, Lu L. Reliable extrapolation of deep neural operators informed by physics or sparse observations. Comput Methods Appl Mech Engrg 2023;412:116064.

[62] Yin M, Zhang E, Yu Y, Karniadakis GE. Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems. Comput Methods Appl Mech Engrg 2022;402:115027.

[63] Goswami S, Yin M, Yu Y, Karniadakis GE. A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. Comput Methods Appl Mech Engrg 2022;391:114587.

[64] Jin P, Meng S, Lu L. MIONet: Learning multiple-input operators via tensor product. 2022, arXiv preprint arXiv:2202.06137.

[65] Benson SM, Cole DR. CO_2 sequestration in deep sedimentary formations. Elements 2008;4(5):325–31.

[66] Zhang D, Song J. Mechanisms for geological carbon sequestration. Procedia IUTAm 2014;10:319–27.

[67] Remy N, Boucher A, Wu J. Applied geostatistics with SGeMS: A user's guide. Cambridge University Press; 2009.

[68] Chen T, Chen H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans Neural Netw 1995;6(4):911–7.

[69] Chen Y, Rao M, Feng K, Niu G. Modified varying index coefficient autoregression model for representation of the nonstationary vibration from a planetary gearbox. IEEE Trans Instrum Meas 2023;72:1–12.

[70] Ma S, Song PX-K. Varying index coefficient models. J Amer Statist Assoc 2015;110(509):341–56.

[71] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention. Springer; 2015, p. 234–41.

[72] Mao S, Dong R, Lu L, Yi KM, Wang S, Perdikaris P. PPDONet: Deep operator networks for fast prediction of steady-state solutions in disk–planet systems. Astrophys J Lett 2023;950(2):L12.

[73] Seidman J, Kissas G, Perdikaris P, Pappas GJ. NOMAD: Nonlinear manifold decoders for operator learning. Adv Neural Inf Process Syst 2022;35:5601–13.