Learning CO_2 Plume Migration in Storage Reservoir using Neural Operators

Isaac Ju
Stanford University
Energy Science and Engineering Department
ju1@stanford.edu

Alaa Alahmed
Stanford University
Energy Science and Engineering Department
alahmeda@stanford.edu

Abstract

Deep-learning-based surrogate models are important for quantifying uncertainties in carbon storage reservoirs. However, existing surrogate models use convolutional neural networks or their variants as a backbone to learn mappings between point-wise evaluations of input and output functions, which is often data-hungry and not mesh-invariant. In this course project, we apply three neural-operator-based surrogate models (UFNO, FNO, and CNO) to learn output functions directly from input functions, training them on a challenging synthetic dataset representing faulted storage reservoirs. The models are employed in an autoregressive manner to predict CO_2 plume sequences with high fidelity, with some achieving mesh invariance. We systematically compare model performances in terms of testing errors and computational efficiency, finding that FNO, without enhancement, achieves the best testing results and is more sample-efficient than UFNO and CNO. CNO maintains the best mesh-invariance but produces blurred predictions that fail to capture sharp edges. FNO outperforms other operators computationally, achieving a 900-fold speedup over traditional simulators. These findings demonstrate the effectiveness of FNO-based models for learning subsurface flow physics compared to CNO, guiding surrogate model choice when data is scarce, a common scenario in geoscience applications. The code for our project is available at `https://github.com/IsaacJu-debug/operator_co2_flow`.

1. Introduction

Multiphase flow in porous media is important for various geoscience applications, such as contaminant transport [2], geothermal energy [12], carbon capture and storage (CCS) [17], hydrogen storage [18], and nuclear waste storage [1]. Limited by expensive field experiments, numerical simulations are often the only tool for accessing these systems, where mass and energy conservation equations are solved monotonically. Simulationt tasks in these applications are very challenging due to the coupled-physics, highly non-linear, and multi-scale nature of the processes involved [16]. However, as fine spatial and temporal discretization to accurately capture flow processes are often required, these simulations are time-consuming and computationally intensive, or even intractable in many scenarios, such as inverse modeling.

2. Related Work

2.1. Deep-Learning-Based Surrogates

Data-driven deep learning (DL) models utilizes data to understand the underlying physics by constructing statistical models from simulation data generated by high-fidelity simulators. These models strive to minimize the data loss between predicted fields and label data, efficiently achieving converged solutions with satisfactory accuracy [4]. Previous DL-based models have exhibited excellent accuracy in predicting flow dynamics [28] and superior computational efficiency compared to High-Fidelity (HF) reservoir simulators [23, 21]. Mo et al. developed a DL surrogate model combining an autoregressive model with a CNN-based encoder-decoder network to predict CO_2 plume migration in random 2D permeability fields [15]. Tang et al.[23, 21] integrated a residual U-Net (R-U-Net) with convLSTM networks to forecast the temporal evolution of saturation and pressure fields in 2D and 3D oil production simulations, later applying their recurrent R-U-Net model to CO_2 storage with coupled flow and geomechanics[22]. Wen et al. [27] developed an R-U-Net-based surrogate model for CO_2 plume migration, encoding injection durations, rates, and locations as input image channels.

Note that these surrogate models learn mappings between pointwise evaluations of input and output functions. Generally, they are not mesh-invariant, which is an important feature for building surrogates for physical systems where data is expensive. Mesh invariance allows the surrogate model to be applied to different discretizations of the same physical domain without retraining, making it more robust and efficient in handling data-scarce scenarios.

2.2. Neural Operators

In contrast to learning mappings between point-wise evaluations, neural operators are design to learn maps between function spaces, and they can be used to approximate the solution operator of a given partial differential equation (PDE). The input and output functions in neural operators can be at any resolution or on any mesh, and the output function can be evaluated at any point in the domain.

Operator networks [5] and DeepONets [14], along with its variants [14], have been developed to learn nonlinear operators from data. PCA-net [3] utilizes principal component analysis to reduce the dimensionality of the input space before learning the operator. Neural operators [7], such as graph neural operator [10], Multipole neural operator [9], and the widely adopted Fourier Neural Operator [8] and its variants [11, 25], have shown promising results in learning operators for various physical systems.

Particularly relevant to our project are the works of U-FNO, an enhanced FNO and CNO, CNN inspired operator networks. Wen et al. [25, 26] combined U-Net and Fourier neural operator (U-FNO) by incorporating convolutional information in the Fourier layer, significantly improving the cost-accuracy trade-off. They demonstrated that U-FNO requires only a third of the training data compared to the CNN baseline model to achieve equivalent accuracy. Raonic et al. proposed the Convolution-based Neural Operator (CNO) [19], which combines the computational efficiency of CNN models with the ability to learn operator mappings between input and output functions. As U-FNO is developed for the subsurface flow system, we consider it as a strong baseline to verify its efficiency on our own dataset and later compare it to CNO.

3. Problem Statement

In this project, we aim to address the aforementioned computational bottleneck by learning the mappings between the input functions and output functions using neural operators [8, 9, 7]. Neural operators learn mappings between infinite-dimensional function spaces and can approximate the solution operator of a given partial differential equation (PDE) or a family of PDE systems.

To achieve this goal, we design our project as two part: (1) first implement a baseline model, U-Net Fourier Neural Operator (U-FNO) [25], to learn the spatio-temporal evolution of the CO_2 saturation; and (2) then compare its performances with FNO and CNO on the same dataset. Particularly, the first component involves applying U-FNO as a recurrent model to learn the sequence of CO_2 predictions, see Figure 1. Given the initial state $\mathbf{Y}^0 = [y_1^0, \ldots, y_{n_C}^0]^T$ and the static model features $\mathbf{M} = [(\mathbf{m}_1)^T, \ldots, (\mathbf{m}_{n_C})^T]^T$ defined at all pixels, we compute the sequence of dynamic variables $(\hat{\mathbf{Y}}^1, \ldots, \hat{\mathbf{Y}}^{n_T})$ in an autoregressive way as follows:

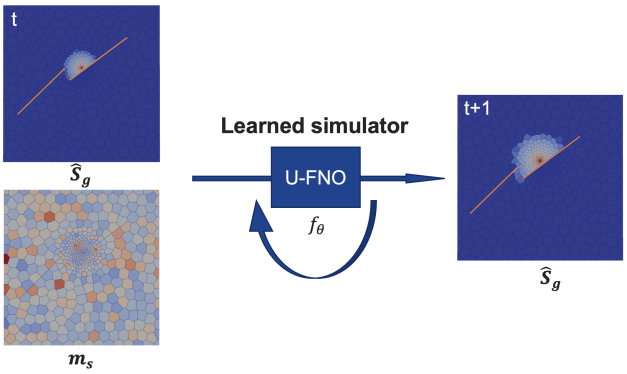

Figure 1: Workflow schematic for the recurrent UFNO model in this project.

$$\hat{\mathbf{Y}}^0 = \mathbf{Y}^0, \tag{1}$$

$$\hat{\mathbf{Y}}^{n+1} = f_{\text{U-FNO},\theta}(\hat{\mathbf{Y}}^n, \mathbf{M}, \mathbf{F}), \qquad n \in \{1, \ldots, n_T\}, \tag{2}$$

where $f_{\text{U-FNO},\theta}$ is the neural model parameterized by the model weights θ. The number of cells (pixels) in the grid is denoted by n_C and the number of temporal snapshots by n_T. Learning the simulator involves finding the parameters that minimizes the data loss between predicted values and ground truth at all timesteps. Following the recommendation in [25], we use a relative l_2 loss to train the deep neural operators, defined as:

$$\mathcal{L}_{\text{U-FNO}} = \frac{1}{n_T} \sum_{n=1}^{n_T} \frac{\|y^n - \hat{y}^n\|_2}{\|y^n\|_2} \tag{3}$$

where n_T denotes the number of rollout steps during the training, y^n denotes the true output in the data set, $\hat{y}^n$ is the output predicted by U-FNO, as formalized in Eq. (2). This form of relative loss acts as a regularizer and proves especially beneficial in cases where the data display significant variance in norms, such as our case. We empirically found that compared with mean squared error (MSE) or root mean square error (RMSE) loss, a l_2 loss significantly improves the performance for predicting gas saturation sequences.

4. Dataset and Evaluation Metrics

4.1. Dataset

We train and evaluate all operator models using a synthetic dataset generated from GEOS, a high-performance computing simulation environment for geoscience applications [6]. The input and output feature maps are given in Figure 2. This dataset mimics a multiphase flow problem with discontinuous structures imposed internal fault boundary. The mathematical formulations and numerical treatments of the simulated physics are given in appendix 8.1.

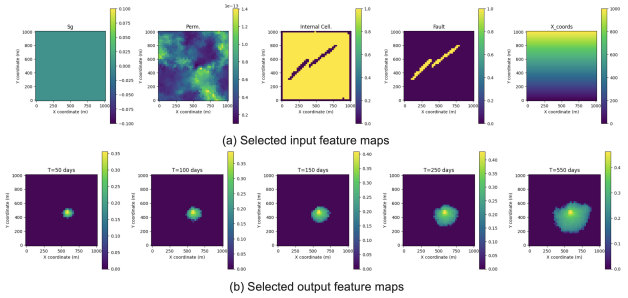

Figure 2: Selected feature maps of (a) input and (b) output. The input features are explained from left to right: s_g represents gas saturation at the initial state; perm. denotes permeability, which affects the transport of the gas plume; type of internal cell (high indicates that the cell is internal); type of faulted cells identifies whether the current cell is on a fault or not; xcoords represents the coordinates of each cell center in x direction. The output maps are a sequence of gas saturations at varying time.

We generate a total of 500 realizations of the synthetic geological models of size 1 km x 1 km x 1 m. The domain shape as well as the position of the two impermeable faults are fixed across all realizations.

This dataset is numerically chanllenging because of having multiple boundary conditions and complex, discontious structures. The latter feature, representing the faults, is known to be challenging for FNO based models, as the FNO tends to predict smooth behaviors. We hypothesize that adding a translation equivarient block, U-Net [20], helps with learning the transport behaviors manifested by our physical systems.

Figure 2 presents the selected input and output feature maps of the dataset. The input feature maps consist of initial gas saturation ($s_{g,0}$), permeability, cell types, and cell coordinates. The output feature is a sequence of gas saturation (s_g) maps that simulate the migration of the CO_2 plume over time.

As mentioned in the last section, the s_g output map will be auto-regressively updated during the training, while the other static features will continue attending the rest of time steps. The output feature maps are essentially a sequence of gas saturation at a fixed time intervals. Observing the output demonstrates that local evolution of gas plume and strong discontinuous behaviors along the fault lines and the boundary of gas plume. In the context of CCS, accurately capturing the plume size and the interactions between CO_2 plume and fault play significant roles in accessing the feasibility of CCS project in a certain reservoir.

The following experiments will be conducted with 400 input simulation results, in which each case has an 11-step rollout of simulation data, representing 550 days of CO_2 injection. To preprocess the dataset, we use 'detrending' scaling for all fields of input and output feature maps. Specifically, the preprocessing can be expressed as:

$$\tilde{\mathbf{x}}_i^n = \frac{\mathbf{x}_i^n - \text{mean}([\mathbf{x}_1^n, \ldots, \mathbf{x}_{n_S}^n])}{\text{std}([\mathbf{x}_1^n, \ldots, \mathbf{x}_{n_S}^n])}, \quad i = 1, \ldots, n_S, \quad n = 1, \ldots, n_T, \tag{4}$$

where n_S represents the total number of training samples. This process involves centering the node features, $\mathbf{x}^n i$, by removing the average field across all nS samples and subsequently scaling by the standard deviation of these features at each time step.

4.2. Evaluation Metrics

To assess the accuracy of gas saturation predictions, we employ the plume saturation error, δ^{s_g}, as presented in [26]. This error metric is defined by the following equation:

$$\delta^{s_g} = \frac{1}{\sum_{i,n} I_i^n} \sum_{n=1}^{n_T} \sum_{i=1}^{n_C} I_i^n \left| s_{g,i}^n - \hat{s}_{g,i}^n \right|, \quad I_i^n = 1 \quad \text{if} \quad \left(s_{g,i}^n > 0.01\right) \cup \left(\left|\hat{s}_{g,i}^n\right| > 0.01\right), \tag{5}$$

where $I_i^n = 1$ signifies the presence of non-zero gas saturation in either the actual data or the prediction for a particular mesh cell, s_g^n represents the actual gas saturation values obtained from HF simulations, $\hat{s}_g^n$ denotes the predicted gas saturation from surrogatr models, n_T refers to the total number of temporal snapshots, and n_C indicates the total number of cells in the mesh.

5. Methods

We consider the state-of-the-art U-FNO model [25] as a baseline and later compare it with recently proposed operator models, such as convolutional neural operator (CNO) [13, 19].

5.1. U-FNO Baseline

U-FNO enhances the original FNO with a mini-UNET to better capture the underlying physics of CO_2 plume migration. Its architecture schematic is depicted in Figure 3 and each block is briefly discussed for completeness:

- **The U-Net Component:** The U-Net architecture is a symmetrically structured encoder-decoder with skip connections. It is optimized to capture and then reconstruct spatial hierarchies within data, and this can be mathematically expressed as:
  Encoder: sequence of convolutional and pooling operations applied to input x resulting in a compressed feature space; expressed formally as:

$$z = E(x; \theta_E)$$

where θ_E denotes encoder parameters.

Decoder: A sequence of up-convolutions that expans the encoded features back to the original input space, assisted by the skip connections, to maintain the spatial information, and modelled as:

$$\hat{x} = D(z; \theta_D),$$

where θ_D is the parameters of the decoder.

- **Fourier Neural Operator (FNO) Block:** The FNO forms the basis for learning mappings of infinite-dimensional function spaces, a property at the very heart of solving PDEs in an effective manner. It is made up of spatial features mapped into the Fourier domain, parametrized spectral transformations, and mapping them back out and is formally written as:

$$\hat{f}(\xi) = \mathcal{F}(f(x)), \quad f(x) = \mathcal{F}^{-1}(R(\xi) \cdot \hat{f}(\xi)),$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier and inverse Fourier transforms, respectively, and $R(\xi)$ is some learnt transformation matrix in the Fourier domain.

This kind of architecture seamlessly integrates the localized processing power of U-Net with the global approximation power of FNO, which makes the learning process not only efficient but also allows accurate modeling of complex systems governed by nonlinear PDEs.

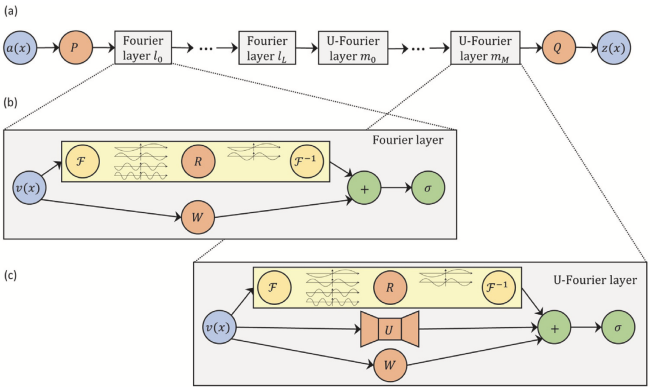

Figure 3: U-FNO model architecture [25].

As U-FNO uses U-Net block to enhance its capability for processing local features, which however sacrifices the mesh-invariant property of neural operator. Using U-FNO to solve PDEs often leads to results that depend heavily on the underlying grid resolution, as we will see in section 6.3. Moreover, U-FNO could suffer from aliasing errors that could occur, especially when applying nonlinear operations performed at the discrete level. This phenomenon, where distinct continuous signals become indistinguishable upon sampling, limits the expressiveness of U-FNO [19].

5.2. CNO Model

Insipred by the U-Net structure, a Convolution-based Neural Operator (CNO) is developed to preserve mesh invariance by employing nonlinear operations that honors continuous-discrete equivalence (CDE) [19]. Respecting CDE property allows CNO to avoid aliasing errors, often suffered from other popular neural operators such as FNO. CNO is a promising neural operator in that it enjoys the computational efficiency of CNN models and also learning operator mappings between input and output functions.

Mathematically, CNO can be defined as a compositional mapping between functions, as follow:

$$G : u \mapsto P(u) = v_0 \mapsto v_1 \mapsto \cdots v_L \mapsto Q\left(v_L\right) = \bar{u} \quad (6)$$

where each layer v_{l+1} is defined as $v_{l+1} = V_l \circ \Sigma_l \circ K_l\left(v_l\right), \quad 0 \leq l \leq L-1$; at layer l, V_l can be either upsampling/downsampling operator, K_l denotes the convolution operator, Σ_l is the activation operator. All elementary operators in CNO are designed to maintain control over the function spectra while preserving the CDE property. To achieve this, nonlinear activation functions are replaced with CDE-preserving counterparts. These counterparts upsample the input signal by a fixed factor, apply regular activation functions (e.g., ReLU), and then downsample the signal by the same factor (see Figure 4). The upsampling factor is chosen in accordance with the Nyquist–Shannon sampling theorem [24]. Mathematical proofs demonstrating why these

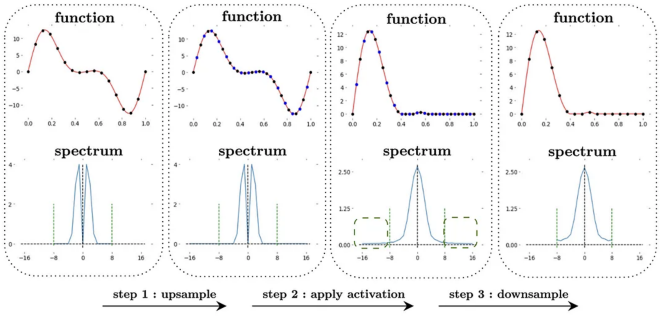

Figure 4: Schematics of the three-step procedure for applying a continuous-discrete equivalent activation function [19].

specially-designed activation functions preserve CDE can be found in [19]. Intuitively, using these nonlinear activation functions allows for the management of high-frequency feature introduction while maintaining the integrity of the underlying function's representation. The architectural details of each elementary operator will be discussed below.

CNO's architecture resemble that of a U-Net, where the input and output are discretized function evaluations and each layer respects CDE property, as shown in Figure 5.

The input function is passed through a set of encoders, where it is downsampled in space but expanded in channel width. The encoded features are then processed through a set of decoders, where the channel width is reduced, and the spatial resolution is increased. To further enhance the network's capacity, "invariant blocks" are added, in which the spatial resolution remains unchanged.

A key feature of CNO's architecture involves including skip connections between encoder and decoder layers at the same spatial resolution or bandlimit. These skip connections are implemented through additional ResNet blocks, allowing for the transfer of high-frequency content before filtering them out with the sinc filter as the input progresses deeper into the encoder. This design choice enables the high-frequency content to be not only recreated with the activation function but also modified through the intermediate networks. As a result, CNO establishes a multiscale operator learning architecture, capable of capturing and processing features at various spatial scales efficiently.

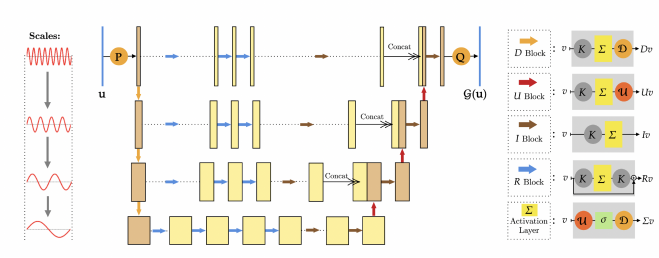

Figure 5: CNO model architecture [19].

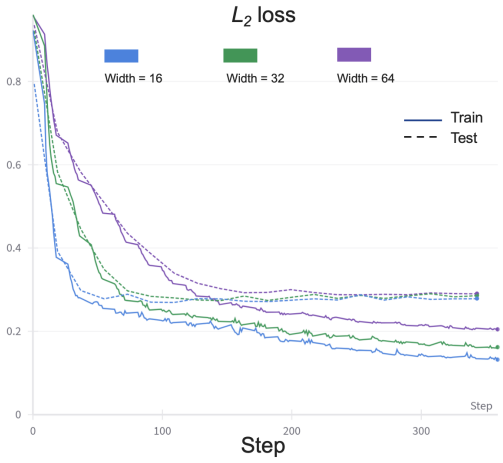

Figure 6: Effects of hidden dimensions on training/testing curves. Width means hidden dimensions used in fourier blocks.

Table 1: Model Performance Metrics with U-FNO Across Different Hidden Dimensions

Metric	**16**	**32**	**64**
Train Loss	0.2047	0.162	0.1321
Test Loss	0.2883	0.2744	0.2699
Best Gas Sat. Error	0.04553	0.04287	0.04183
Training time (s)	744.0	833.0	1038.0
Model parameters	486,945	1,941,313	7,753,089

6. Experiments

In this section, we provide training and implementation details for three different neural operators, namely the baseline model U-FNO, FNO, and CNO. Trained and tested on the dataset for CO_2-water flow system, we have compared the performances between three neural operators, where the predicted spatial-temporal fields, mesh-invariance testing, and computational efficiencies are all demonstrated.

6.1. Training and Implementation of Neural Operators

To find a good baseline U-FNO model, we focus on tuning different hidden dimensions (16, 32, 64) and block numbers (2, 3, 4, 5) of fourier blocks. We have identified hidden dimension size has a prominent impact while block number is found to be 4. Figure 6 shows that the effects of hidden dimensions on the training and testing curves with 400 training cases. We find that hidden dimension of 32 strike a good balance between model parameters and performances (see Table 1). The detailed model architecture of the baseline U-FNO is given in Table 5.

We also find that using a proper normalization layer ensures the stable and convergent behavior during the training, as the recurrent model often suffers from gradient exploding issues. Batchnorm is used for U-FNO model in this project. We have trained all models with 200 epochs with a training batch size of 128 to speed up the training. The dataset sizes for training and testing are 400 and 100, respectively. Adam optimizer with a learning rate of η is used to minimize the l_2-loss function. We also use a weight decay of magnitude w and a step learning rate scheduler and reduce the learning rate of each parameter group by a factor γ. All the training parameters are listed in Table 2. Note that we use the same set of training parameters for training all neural operator models.

6.2. Predicting Spatial-Temporal Distributions with Neural Operators

In this section, we compare U-FNO against the original FNO and CNO in terms of prediciting gas saturation sequences. The model details of U-FNO is given in appendix 8.2.

Table 2: Hyperparameters used for training.

Parameter name	value
Loss function	l_2 Relative Loss
Number of epochs	200
Training Batch size	128
Learning rate, η	0.0001
Weight decay, γ	$5e-4$
Optimizer	Adam
Optimizer scheduler	step
Learning reduction factor, γ	50

6.2.1 U-FNO Performance

We consider unseen mesh 450 from the test set as an examples to first demonstrate the accuracy of the baseline model, U-FNO. Figure 7 shows the predicted s_g maps at 50, 300 and 550 days. Overall, our recurrent model has learned the tempo-spatial dynamics of CO_2 plume migration in a faulted system. The shapes of plumes at different timesteps resemble that of ground truth. However, the prediction around the plume edge still manifest large errors, as highlighted in the last column of Figure 7.

6.2.2 FNO and CNO Performances

Figure 8 compares the model predictions of three operators for gas saturation with unseen mesh 400 at 550 days, which demonstrates that FNO and UFNO perform similarly and can capture the complex CO_2 shapes. However, CNO prediction is very blurried and fails to track the shape of CO_2 plumes.

Table 3: Performance Metrics Comparison between U-FNO, FNO, and CNO. Sg error denotes the gas plume errors, defined in Equ. 5. Test loss denotes relative l_2 loss. The training resolution is 40, where other resolutions are only seen during testing.

Metric	**FNO**	**U-FNO**	**CNO**
Sg Error 120	**0.04572**	0.1092	0.05023
Sg Error 24	0.05175	0.07083	**0.04962**
Sg Error 40	**0.045**	**0.04503**	0.04857
Sg Error 56	**0.04656**	0.0619	0.04839
Test Loss 120	0.6296	1.173	**0.3385**
Test Loss 24	0.4938	0.503	**0.3602**
Test Loss 40	**0.2795**	0.2862	0.3141
Test Loss 56	0.4003	0.4405	**0.3257**
Runtime (seconds)	**647**	833	11771

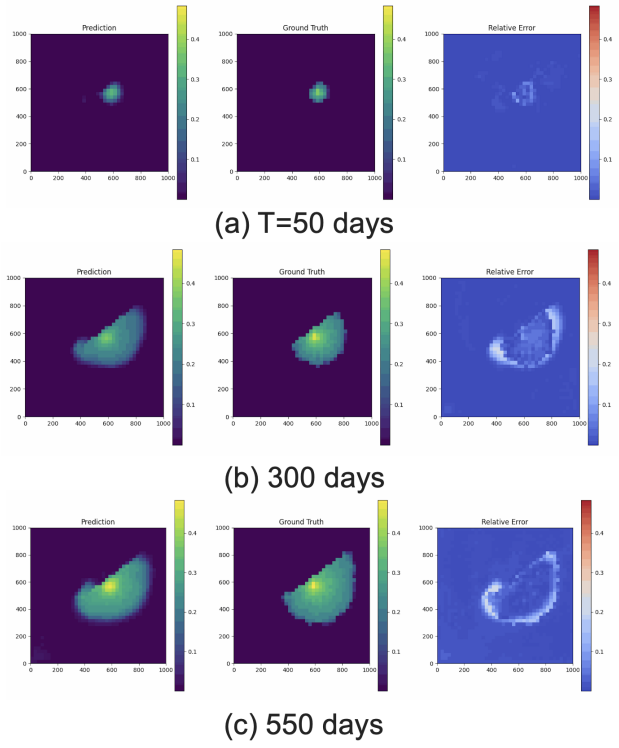

Figure 7: Prediction accuracy for gas saturation after rolling out for (a) 50, (b) 300, and (c) 550 days. Prediction column is the model inference, whereas the last two columns are ground truth and relative error. Testing is done with mesh 450.

6.3. Mesh-Invariance Testing Between Neural Operators

An important feature of an operator learning model is its mesh-invarience property, namely maintaining relatively similar test errors when evaluated on various resolutions or discretizations [10]. In this section, we compared the mesh invariance of three neural operators by computing the testing errors with different resolutions. Specifically, all the models are trained with simulation data of baseline resolution, namely 40x40, and then tested on three unseen resolutions, ranging from 24x24 to 120x120. The test cases with new resolutions are generated from interpolating from the base-resolution simulation data, whose gas saturation maps are demonstrated in Figure 9.

During testing, FNO and U-FNO can take input with different sizes, whereas CNO requires the input with a fixed size. Particularly, for CNO, we need to first transform input maps with unseen resolutions into the base resolution, which is fed into the trained CNO for generating predictions at the base resolution. Then, the prediction sequence is transformed back to the testing resolutions. Note that all

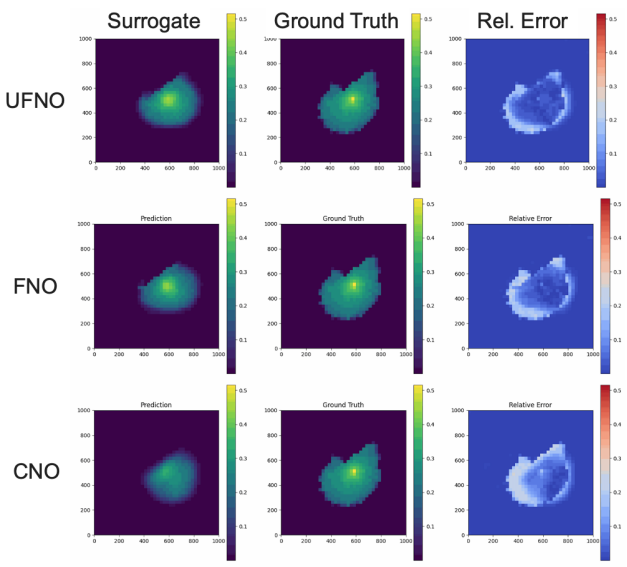

Figure 8: Prediction accuracy for gas saturation after rolling out for 550 days with U-FNO, FNO and CNO. The first column is the model output, whereas the last two columns are ground truth and relative error.

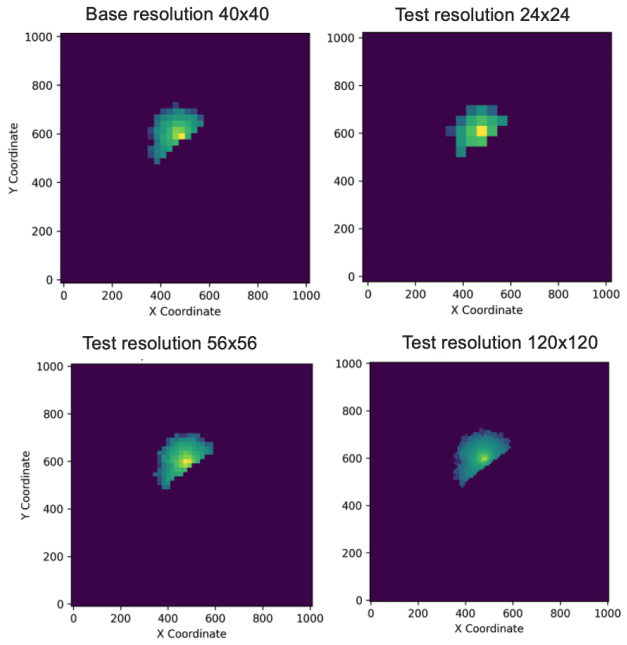

Figure 9: Ground truths of gas saturations with mesh 499 after 550 days. Testing cases with new resolutions are interpolated.

transformations involved need to follow the sampling theorm, whose details are given in the original paper [19].

Table 3 lists the test losses and s_g errors across different resolutions for the multiphase flow benchmark. The gas predictions with the resolution of 120 is also plotted to visualize the locations of prediction mismatches. The CNO model exhibits the greatest stability in response to resolution variations, maintaining an approximately constant error rate of around 0.049, demonstrating its invariance to resolution changes. Particularly, with the resolution of 120, CNO is the only model among others, which can predict the gas plume size without introducing patch-like predictions, as UFNO and FNO does (see Figure 10). The FNO also displays a certain degree of mesh-invariance, with downsampling cases (20x20) typically yielding the highest errors, while upsampling cases seem to preserve mesh invariance effectively. In contrast, the U-FNO's test error increases markedly with varying resolutions and shows a maximum increase by a factor of 2, indicating that U-FNO does not maintain mesh invariance.

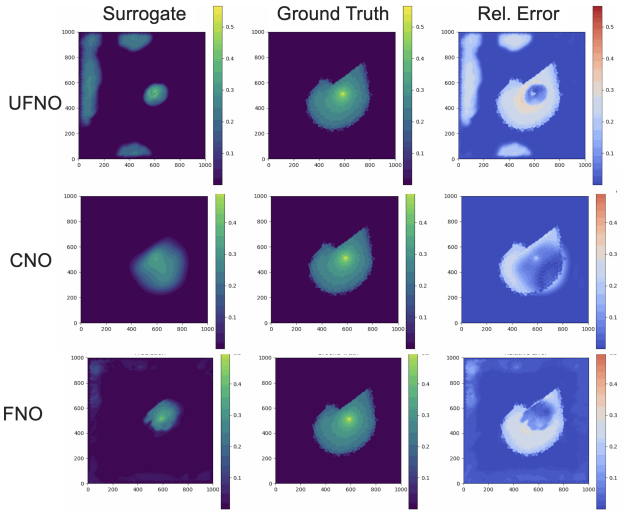

Figure 10: gas predictions at a resolution of 120 for UFNO, CNO, and FNO. For each model, the first column depicts the surrogate prediction, the second column shows the ground truth, and the third column visualizes the relative error between the prediction and ground truth.

6.4. Learning Efficiency Comparisons Between Neural Operators

This section focuses on comparing the learning efficiency of different neural operators. The computational efficiency is measured by the parameter size of models while achieving the same level of testing errors. To this end, we plot the model size against the testing erros, as shown in Figure 11, for our multiphase flow problem. Clearly, for the similar model size, FNO models lead to significantly smaller test losses and gas errors in a consitent manner.

Another important factor affecting the efficiency of neu-

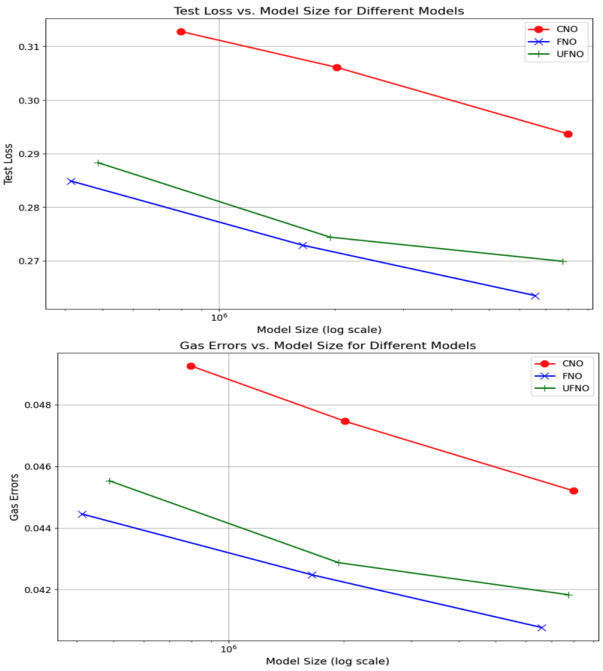

Figure 11: Test Losses/Gas errors versus Model size.

ral operators is the speed of training time. This is a critical metric for a good surrogate model. We also highlight the training time used for achieving certain testing errors, as shown in Table 3, where FNO-based models greatly outpeform CNO by 18 folds. This result contradicts the original CNO paper's claim that CNO surpasses other neural operators. The discrepancy may be due to the simplistic implementation of the CDE-preserving activation function using the torch interpolation function.

6.5. Inference Efficiency Comparisons Between Surrogate Models and Numerical Simulators

The most prominent advantage of using DL-based models for building surrogate models is its computational efficiency over traditional simulators. To this end, we compare the inference times of FNO, U-FNO, CNO against the numerical simulator, GEOS, as demonstrated in Table 4. Utilizing the same dataset for multiphase flow problems, FNO requires an average of 0.0247 seconds for a 11-step rollout on an NVIDIA Tesla A100 GPU to process a single batch. Moreover, FNO outperforms both U-FNO and CNO.

Comparing these surrogate models with GEOS demonstrates a significant performance gain. Specifically, on the same dataset, FNO exhibits a nearly 900-fold reduction in execution time in comparison to GEOS, which operates on a CPU Intel(R) Xeon(R) E5-2680 v4 2.10GHz. We expect that this performance gain will be even more prominent with a larger mesh size.

Table 4: Average inference times for UFNO, FNO, and CNO with 11-step rollouts over 550 days, compared with GEOS run times.

Model	Inference Time (s)	
	11-step Rollout[a]	GEOS Run Time[b]
FNO	**0.0247**	22.12
U-FNO	0.04827	22.12
CNO	0.1133	22.12

[a] On an NVIDIA Tesla A100 GPU, single-batch inference run.
[b] On an Intel Xeon E5–2695 v4, single-core serial run.

7. Conclusion and Future Work

In this course project, we have successfully applied three state-of-art neural operators, including UFNO, FNO, and CNO to a challenging multiphase flow benchmark. The neural operators are employed in autoregressive manner to predict a long sequence of CO_2 plumes with high fidelity and some models are demonstrated to achieve mesh invariance.

We systematically compared model performances in terms of testing errors and computational efficiency. We find that FNO, without enhancement, achieved the best testing results and was more sample-efficient than UFNO and CNO. CNO maintained the best mesh-invariance but produced blurred predictions that failed to capture sharp edges. FNO outperformed other operators computationally, achieving a 900-fold speedup over traditional simulators. These findings demonstrate the effectiveness of FNO-based models for learning subsurface flow physics compared to CNO, guiding surrogate model choice when data is scarce, which is often encountered in geoscience applications.

Although we have demonstrated the effictiveness of neural-operator-based surrogate models for predicting CO_2 plume, there are several limitations and possible future directions to further improve the applicability of the neural surrogates. The current model assumes the underlying simulation data are grid based, which is a big limitation as the unstructured mesh data with complex geometry is often of great interest to CCS community. One possible way to mitigate this limitation is to use a graph-based encoder to first map the geometrical data into a grid-based data, which later can be processed by the neural operators. Another limitation of our study is that we did not explicitly impose the physical laws, such as mass/energy balances, into the loss fucntion to ensure the trained surroage honor these important inductive bias. Correspondingly, incorporating a physics-informed loss terms during the training would be a natural choice to inject stronger inductive bias to achieve even higher sample efficiency.

8. Appendix

8.1. Governing Equations of CO_2-Brine Flow

In this work, we consider miscible two-phase (gas and aqueous) two-component (H_2O and CO_2) flow in a compressible porous medium. The H_2O component is only present in the aqueous phase, while the CO_2 component can be present in both the aqueous and the gas phases. We denote the aqueous and the gas phases using the subscripts a and g, respectively. The mass conservation of each component reads:

$$\begin{aligned}\frac{\partial}{\partial t}\left(\phi\sum_{\ell=1}^{2}x_{c\ell}\rho_\ell s_\ell\right)+\nabla\cdot\left(\sum_{\ell=1}^{2}x_{c\ell}\rho_\ell \boldsymbol{v}_\ell\right)\\ +\sum_{\ell=1}^{2}x_{c\ell}\rho_\ell q_\ell=0,\qquad c=\{\mathrm{H_2O},\mathrm{CO_2}\},\end{aligned}\tag{7}$$

where ϕ is the porosity, $x_{c\ell}$ is the mass fraction of component c in phase ℓ, ρ_ℓ is the density of phase ℓ, s_ℓ is the saturation of phase ℓ, $\boldsymbol{v}_\ell$ is the Darcy velocity of phase ℓ, and q_ℓ is the source flux for phase ℓ. Using the multiphase extension of Darcy's law, we write that the Darcy velocity is proportional to the gradient of the pressure:

$$\boldsymbol{v}_\ell=-\frac{k_{r\ell}}{\mu_\ell}\bar{\mathbf{k}}\cdot\nabla p_\ell,\qquad \ell=\{a,g\},\tag{8}$$

where $k_{r\ell}$ is the relative permeability of phase ℓ, μ_ℓ is the viscosity of phase ℓ, $\bar{\mathbf{k}}$ is the permeability tensor, and p_ℓ is the phase pressure. In this work, we assume that the permeability tensor is diagonal with an equal value for each entry. The system is closed with the following constraints

$$s_g+s_a=1,\tag{9}$$

$$p_g-p_a=p_c(s_g),\tag{10}$$

$$x_{\mathrm{H_2O},\ell}+x_{\mathrm{CO_2},\ell}=1,\qquad \ell\in\{a,g\},\tag{11}$$

as well as standard thermodynamics constraints on fugacities. A well injects pure supercritical CO_2 at a rate of 0.058 kg/s for 950 days, assuming a storage reservoir with unit meter thickness.

8.2. U-FNO and FNO Model Architectures

For U-FNO, see Table 5. For FNO, see Table 6.

Table 5: The U-FNO model runs simulations at base resolutions. Padding handles non-periodic boundaries. Linear layers elevate and project input dimensions back. Fourier2d applies a 2D Fourier transform, and Conv1d adds a bias term. UNet2d is a two-stage 2D U-Net. The Add operation combines layer outputs, while ReLU introduces nonlinearity. BatchNorm standardizes outputs, improving convergence and training stability. The model has 1,941,313 parameters.

Part	Layer	Output Shape
Input	–	(40, 40, 1)
Padding	Padding	(48, 48, 1)
Lifting	Linear	(48, 48, 32)
Fourier 1	Fourier2d/Conv1d/Add/ BatchNorm/ReLu	(48, 48, 32)
Fourier 2	Fourier2d/Conv1d/Add/ BatchNorm/ReLu	(48, 48, 32)
U-Fourier 1	Fourier2d/Conv1d/UNet2d/Add/ BatchNorm/ReLu	(48, 48, 32)
U-Fourier 2	Fourier2d/Conv1d/UNet2d/Add/ BatchNorm/ReLu	(48, 48, 32)
Projection 1	Linear	(48, 48, 32)
Projection 2	Linear	(48, 48, 1)
De-padding	–	(40, 40, 1)

Table 6: The FNO model runs simulations at base resolutions. Padding handles non-periodic boundaries. Linear layers elevate and project input dimensions back. Fourier2d applies a 2D Fourier transform, and Conv1d adds a bias term. The Add operation combines layer outputs, while ReLU introduces non-linearity. BatchNorm standardizes outputs, enhancing convergence and training stability. The model has 1,656,706 parameters.

Part	Layer	Output Shape
Input	–	(40, 40, 1)
Padding	Padding	(48, 48, 1)
Lifting	Linear	(48, 48, 32)
Fourier 1	Fourier2d/Conv1d/Add/ BatchNorm/ReLu	(48, 48, 32)
Fourier 2	Fourier2d/Conv1d/Add/ BatchNorm/ReLu	(48, 48, 32)
Fourier 3	Fourier2d/Conv1d/Add/ BatchNorm/ReLu	(48, 48, 32)
Fourier 4	Fourier2d/Conv1d/Add/ BatchNorm/ReLu	(48, 48, 32)
Projection 1	Linear	(48, 48, 32)
Projection 2	Linear	(48, 48, 1)
De-padding	–	(40, 40, 1)

9. Contributions and Acknowledgements

Isaac Ju: Conceptualization; Methodology; Software; Visualization; Writing - original draft. **Alaa Alhemed**: Conceptualization; Methodology; Writing - original draft.

We thank Ishikaa Lunawat, our assigned mentor, for her help with reviewing our milestones and fruitful discussions. We also acknowledge the Stanford Center for Computational Earth & Environmental Science (CEES) for providing the computational resources used in this work.

References

[1] B. Amaziane, M. El Ossmani, and M. Jurak. Numerical simulation of gas migration through engineered and geological barriers for a deep repository for radioactive waste. *Computing and Visualization in Science*, 15(1):3–20, 2012.

[2] J. Bear and A. H.-D. Cheng. *Modeling groundwater flow and contaminant transport*, volume 23. Springer, 2010.

[3] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart. Model reduction and neural networks for parametric pdes. *The SMAI journal of computational mathematics*, 7:121–157, 2021.

[4] R. Bischof and M. Kraus. Multi-objective loss balancing for physics-informed deep learning. *arXiv preprint arXiv:2110.09813*, 2021.

[5] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

[6] H. Gross and A. Mazuyer. Geosx: A multiphysics, multilevel simulator designed for exascale computing. In *SPE Reservoir Simulation Conference*. OnePetro, 2021.

[7] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

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

[9] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Multipole graph neural operator for parametric partial differential equations. *arXiv preprint arXiv:2006.09535*, 2020.

[10] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[11] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3):1–27, 2024.

[12] P. C. Lichtner and S. Karra. Modeling multiscale-multiphase-multicomponent reactive flows in porous media: Application to co2 sequestration and enhanced geothermal energy using pflotran. *Computational Models for CO2 Geosequestration & Compressed Air Energy Storage*, pages 81–136, 2014.

[13] M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. *arXiv preprint arXiv:2402.16845*, 2024.

[14] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[15] S. Mo, Y. Zhu, N. Zabaras, X. Shi, and J. Wu. Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. *Water Resources Research*, 55(1):703–728, 2019.

[16] F. M. Orr et al. *Theory of gas injection processes*, volume 5. Tie-Line Publications Copenhagen, 2007.

[17] R. K. Pachauri, M. R. Allen, V. R. Barros, J. Broome, W. Cramer, R. Christ, J. A. Church, L. Clarke, Q. Dahe, P. Dasgupta, et al. *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*. Ipcc, 2014.

[18] B. Page, G. Turan, A. Zapantis, J. Burrows, C. Consoli, J. Erikson, I. Havercroft, D. Kearns, H. Liu, D. Rassool, et al. The global status of ccs 2020: Vital to achieve net zero, 2020.

[19] B. Raonic, R. Molinaro, T. De Ryck, T. Rohner, F. Bartolucci, R. Alaifari, S. Mishra, and E. de Bézenac. Convolutional neural operators for robust and accurate learning of pdes. *Advances in Neural Information Processing Systems*, 36, 2024.

[20] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[21] H. Tang, P. Fu, C. S. Sherman, J. Zhang, X. Ju, F. Hamon, N. A. Azzolina, M. Burton-Kelly, and J. P. Morris. A deep learning-accelerated data assimilation and forecasting workflow for commercial-scale geologic carbon storage. *International Journal of Greenhouse Gas Control*, 112:103488, 2021.

[22] M. Tang, X. Ju, and L. J. Durlofsky. Deep-learning-based coupled flow-geomechanics surrogate model for co2 sequestration. *International Journal of Greenhouse Gas Control*, 118:103692, 2022.

[23] M. Tang, Y. Liu, and L. J. Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *Journal of Computational Physics*, 413:109456, 2020.

[24] M. Unser. Sampling-50 years after shannon. *Proceedings of the IEEE*, 88(4):569–587, 2000.

[25] G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

[26] G. Wen, Z. Li, Q. Long, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson. Real-time high-resolution co 2 geological storage prediction using nested fourier neural operators. *Energy & Environmental Science*, 16(4):1732–1741, 2023.

[27] G. Wen, M. Tang, and S. M. Benson. Towards a predictor for co_2 plume migration using deep neural networks. *International Journal of Greenhouse Gas Control*, 105:103223, 2021.

[28] Y. Zhu and N. Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.